Département Informatique INSA de Lyon

Ordonnancement temps réel

Jean-Philippe Babau

Département Informatique, INSA Lyon

jean-philippe.pbabau@insa-lyon.fr

Département Informatique INSA de Lyon

jean-philippe.pbabau@insa-lyon.fr

Plan

- Introduction
- Ordonnancement et temps de réponse
  - Tâches indépendantes
  - Serveurs, ressources partagées
  - Autres contraintes
- Exemple
  - Suivi de capteurs en temps réel
- Estimation de la pire durée d'exécution

jean-philippe.pbabau@insa-lyon.fr

Problématique du temps réel

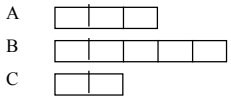
Lorsqu'un événement survient, on doit effectuer une séquence d'actions dans un délai borné, appelé échéance

- Paramètres temporels
  - Fréquence d'arrivée de l'événement (périodique, sporadique, …)
  - Enchaînement des actions
  - Durée d'utilisation du processeur pour traiter les actions
    - Pire durée : WCET (Worst Case Execution Time)
- Temps de réponse
  - Intervalle de temps séparant la fin de l'exécution des actions de l'arrivé de l'événement déclenchant
  - Durée d'exécution **+** tous les temps d'attente (blocages, préemptions, …)
  - Pire temps de réponse : WRT (Worst Response Time)
- Ordonnancement
  - Choix de l'action à exécuter à un instant donné
    - calcul des priorités, mise en place d'une séquence
- Analyse d'ordonnançabilité
  - WRT < échéance
  - borne supérieure(WRT) < échéance

jean-philippe.pbabau@insa-lyon.fr

Ordonnancement circulaire

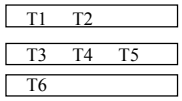
• quantum de temps To :
  – durées < To Trép = N x To + durée(tâche)
  – durées > To Trép = (N+1) x durée(tâche)



To = 5 RT(C) =

To = 1 RT(C) =

• quantum de temps et file multiples :

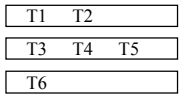


4 To

2 To

To

•quantum de temps et file à priorité :



High (To)

Medium (To)

Low (To)

jean-philippe.pbabau@insa-lyon.fr

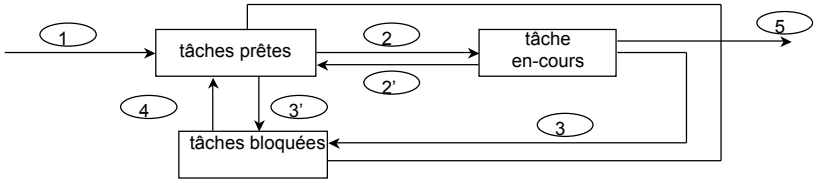
Définitions

- Priorité
  – élément utilisé par l'ordonnanceur pour élire une tâche
  – Priorité statique / priorité dynamique
- Algorithme d'ordonnancement
  – algorithme de calcul des priorités des tâches
- Ordonnancement préemptif
  – L'action la plus prioritaire est toujours élue
- Ordonnancement non préemptif
  – L'élection a lieu uniquement lorsque une action est terminée
- Utilisation des algorithmes
  – en ligne (on-line)
  – hors ligne (off line, pre run time scheduling)

jean-philippe.pbabau@insa-lyon.fr

Ordonnanceur

- Rôle
  - choix de la tâche à exécuter parmi les tâches prêtes



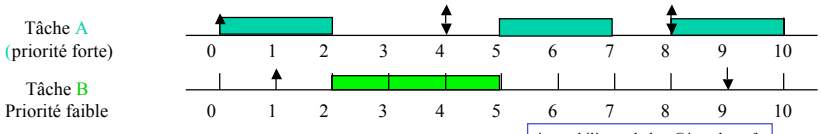
- 1 : création de tâche (create)
- 2 : élection
  - système préemptif : 1, 3, 4, 5
  - système non préemptif : 1(si libre), 3, 5
- 2' : préemption
- 3 : demande de ressource occupée, attente événement non présent, auto-suspension
- 3' : suspension (suspend)
- 4 : ressource libérée, événement arrivé, réactivation
- 5 : fin (delete)

jean-philippe.pbabau@insa-lyon.fr

Séquence d'exécution

- Séquence
  - trace temporelle de l 'exécution d 'un ensemble de tâche (diagramme de Gantt)
  - Tornado : WindView
- Séquence valide
  - une séquence est valide lorsque chaque tâches est exécutée dans le respect de ses contraintes de temps (activation, délai)

Tâche	ri date de départ	Ci durée d ' exécution	Ri échéance	Ti période	rti Temps de réponse	wrti Pire temps de réponse
A	0	2	4	4	2, 3	3
B	1	3	8	-	4	4



jean-philippe.pbabau@insa-lyon.fr

Département Informatique INSA de Lyon

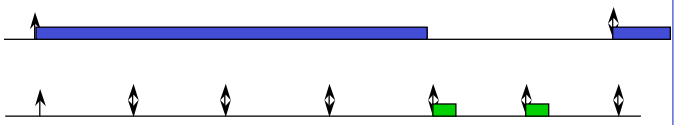
Pourquoi ordonnancer ?

"soit une application de contrôle composée d'une activité de contrôle moteur et d'une activité de monitoring implémentées par deux taches périodiques (tâches *control* et *display*). L'architecture s'appuie sur un micro contrôler (40 Mhz)"

- Première intuition : le contrôle est plus important plus prioritaire

Tâche *Control*
Période= 60 ms
Durée = 40 ms

Tâche *Display*
Période = 10 ms
Durée = 2 ms



➔ l'information clignote à l'écran

jean-philippe.pbabau@insa-lyon.fr

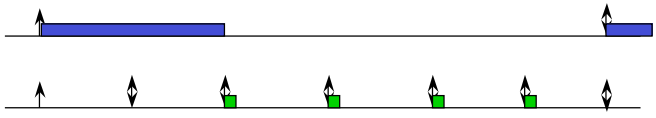
Département Informatique INSA de Lyon

Pourquoi ordonnancer ?

- Processeur plus rapide
  - 80 Mhz

Tâche *Control*

Tâche *Display*

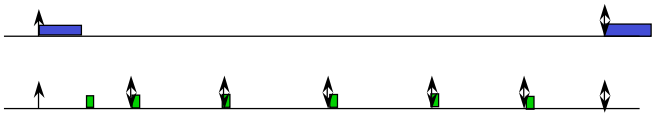


➔ l'information clignote encore à l'écran

- Processeur trè rapide
  - 320 Mhz !!!

Tâche *Control*

Tâche *Display*

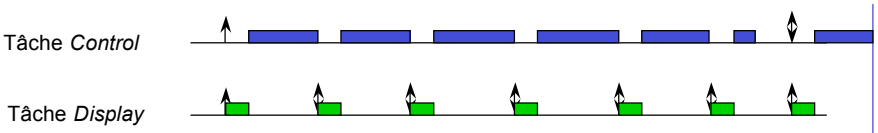


➔ Solution couteuse

jean-philippe.pbabau@insa-lyon.fr

Pourquoi ordonnancer ?

- Solution temps réel
  - Meilleur ordonnancement
  - La priorité est liée à l'urgence temporelle
  - Processeur bas coût



jean-philippe.pbabau@insa-lyon.fr

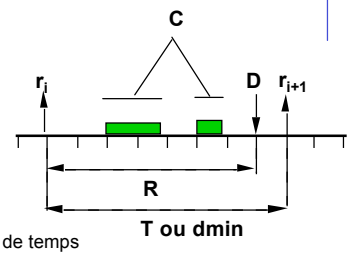
Plan

- Introduction
- Ordonnancement et temps de réponse
  - Tâches indépendantes
  - Serveurs, ressources partagées
  - Autres contraintes
- Exemple
  - Suivi de capteurs en temps réel
- Estimation de la pire durée d'exécution

jean-philippe.pbabau@insa-lyon.fr

Modèle des tâches

- Paramètres des tâches : r_0, C, R, T or dmin
  - date d'activation r_i
    - tâche périodique, période **T**
      - $r_i = r_0 + i\,T$
    - tâche sporadique, intervalle minimal **dmin**
      - $r_{i+1} - r_i > dmin$
    - tâche aperiodique
      - Nombre maximum d'activation par intervalle de temps
      - Probabilité d'arrivée
  - pire durée d'exécution (WCET) **C**
  - échéance relative **R**, échéance absolue **D**
  - Contraintes : $0 < C < R \leq T$



• tâche périodique

• tâche sporadique

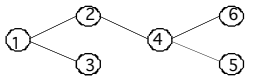
dmin : intervalle miminum entre deux activations

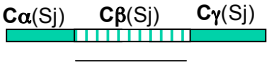
jean-philippe.pbabau@insa-lyon.fr

Département Informatique INSA de Lyon

Modèles des tâches

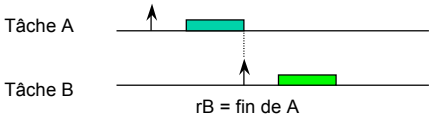
- Communications
  - synchronisation : graphe de précédence
  - ressources partagées
    - pour chaque tâche, par ressource S, $C\alpha(S)$ $C\beta(S)$ $C\gamma(S)$
    - $C\alpha(S) + C\beta(S) + C\gamma(S) = C$





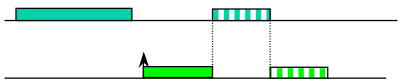
Section critique

• Relations de précédence



rB = fin de A

• Exclusion mutuelle



Section critique

jean-philippe.pbabau@insa-lyon.fr

Evaluation du temps de réponse

- Temps de réponse : Trép = durée + Σ (retard)
  - Evaluable dans des cas simples
- Tâches périodiques préemptibles indépendantes synchrones (r_0 identique), priorités fixes
  - Durée : Ci, Période : Ti, délai : Ri = Ti, Tâche j : tâche plus prioritaire
  - RT : solution (si valide) de Wi = Ci + Σ (⌈ Wi/Tj ⌉)Cj
  - RT: borné (si valide) par RTMax = Ci + Σ (⌈ Ti/Tj ⌉)Cj
- Tâches périodiques préemptibles indépendantes non synchrones, priorités fixes
  - RT : borné par la solution (si valide) de Wi = Ci + Σ (⌈ Wi/Tj ⌉)Cj
  - RT : borné (si valide) par Trépmax = Ci + Σ (⌈ Ti/Tj ⌉)Cj

jean-philippe.pbabau@insa-lyon.fr

RMA

Rate Monotonic Analysis «A practionner's Handbook for Real-Time Analysis»
M. Klein & all, Kluwer Academic Publishers, 1993.

- Pour chaque événement
  - type (hard/soft)
  - caractéristiques temporelles (période, intervalle dmin)
  - l'enchaînement des actions associées
- Pour chaque action
  - le temps d'exécution
  - la priorité
  - les ressources utilisées
  - la préemptibilité
  - la gigue
- Pour chaque ressource
  - la politique d'accès
  - le temps d'accès
- Résultat
  - Pire temps de réponse, validité, charge, séquence, ...

jean-philippe.pbabau@insa-lyon.fr

Evaluation du temps de réponse

- Analyse RMA
  - Durée : Ci, Période : Ti, délai : Ri = Ti

$$Borne(WRTi) = Ci + \Sigma (\lceil Ti/Tj \rceil) Cj + \Sigma Bi + Max(Ck) + n\Sigma (\lceil Ti/dmin \rceil) (Ij) + \Sigma (\lceil Ti/Tsys \rceil) (Csys)$$

WCET — tâches plus prioritaires — ressources partagées — tâches moins prioritaires non preemptibles — interruptions — exécutif

- Problème de réalisme
  - Borne des pires durées, borne des pires temps de réponse
  - Charge processeur réelle de 30 %

jean-philippe.pbabau@insa-lyon.fr

Algorithmes d'ordonnancement

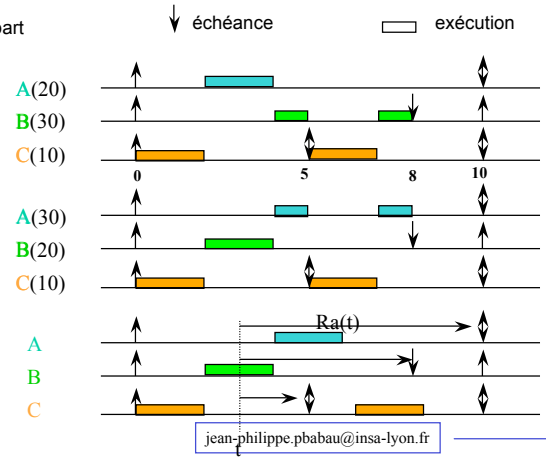
- Principes
  mise en place des priorités
  construction de la séquence d'exécution

Exemple — ↑ date de départ — ↓ échéance — exécution

Rate Monotonic
Ti<Tj => Pi>Pj

Deadline Monotonic
Ri<Rj => Pi>Pj

Earliest Deadline
Ri(t)<Rj(t) => Pi>Pj



jean-philippe.pbabau@insa-lyon.fr

Temps de réponse

- Simulation
  RT(A) =
  RT(B) =
  RT(C) =
- Analyse réaliste (solution de $W_i = C_i + \Sigma (\lceil W_i/T_j \rceil) C_j)_j$
  RT(A) =
  RT(B) =
  RT(C) =
- Analyse RMA
  RT(A) =
  RT(B) =
  RT(C) =

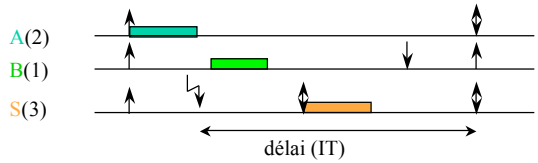
jean-philippe.pbabau@insa-lyon.fr

Requête non-périodique

- Traitement en-ligne
  - service immédiat
    - traitement d'urgence
  - service dans les temps creux
    - traitement sans contraintes
- Serveur
  - mémorisation de la requête
  - traitement différé
    - tâches périodiques : polling server, deferrable server, Exchange Priority Server
    - tâches sporadiques : sporadic server
    - TBS server : serveur apériodique avec ED

RM

serveur à scrutation
(polling server)



jean-philippe.pbabau@insa-lyon.fr

Temps de réponse

- Temps de réponse de la tâche serveur S
  - Idem tâche
- Temps de réponse à un événement
  - Temps de prise en compte de l'événement par le serveur **+** RT(Serveur)

    RT(event) ≤ T_S + RT(Serveur) ≤ T_S + WRT(Serveur)
- Paramètres du serveur

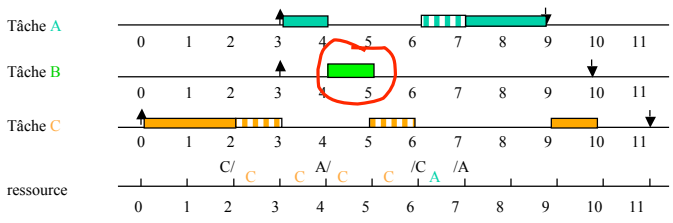
  T_S + WRT(Serveur) ≤ T_S + R_S ≤ délai

  par exemple : T_S = R_S = 1/2 délai
- Analyse RMA pour une IT =

jean-philippe.pbabau@insa-lyon.fr

Prise en compte des ressources critiques inversion de priorité

• Notations
- demande d 'une ressource (C demande la ressource : C/)
- attribution de la ressource (ressource attribuée à D : D)
- libération d 'une ressource (D libère la ressource : /D)



Priorité(A) > priorité(B) > priorité(C)

Section critique

• Blocage sur ressource
- ressource déjà allouée
- tâche moins prioritaire (inversion de priorité)

jean-philippe.pbabau@insa-lyon.fr

Protocole à héritage de priorité

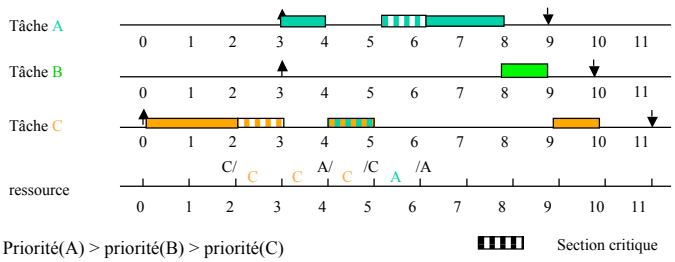
- Priority Inheritance Protocol
- Principe
  la tâche en section critique hérite de la plus haute priorité parmi les tâches bloquées
- Héritage simple
  l 'héritage se fait par file d 'attente d 'accès aux ressources critiques
- Résultat
  réduction du temps d 'attente en section critique
- VxWorks

```
/*creation d un semaphore mutex pour avec héritage de priorité */

LOCAL SEM_ID       semId;

semId = semMCreate(SEM_Q_PRIORITY | SEM_INVERSION_SAFE);
```

jean-philippe.pbabau@insa-lyon.fr

Protocole à héritage de priorité

- Notations
  - demande d 'une ressource (C demande la ressource : C/)
  - attribution de la ressource (ressource attribuée à D : D)
  - libération d 'une ressource (D libère la ressource : /D)



- Blocage sur ressource
  - ressource déjà allouée

jean-philippe.pbabau@insa-lyon.fr

Mars pathfinder 1997, robot Sojourner

- Mission
  - Robot d'exploration
  - Transmission d'images sur la terre
- Architecture
  - OS VxWorks
  - Un bus partagé en exclusion mutuelle
  - Accès au bus
    - Protégé par mutex
    - Une tâche « information thread » en lecture/écriture sur le bus
      - forte priorité
      - fréquemment exécutée
    - Une tâche « meteorological thread » en écriture sur le bus
      - à faible priorité
      - durée faible
      - peu utilisée
    - Une tâche de priorité intermédiaire
    - Watchdog d'accès au bus : si dépassé reset général
- Bug
  - Inversion de priorité

jean-philippe.pbabau@insa-lyon.fr

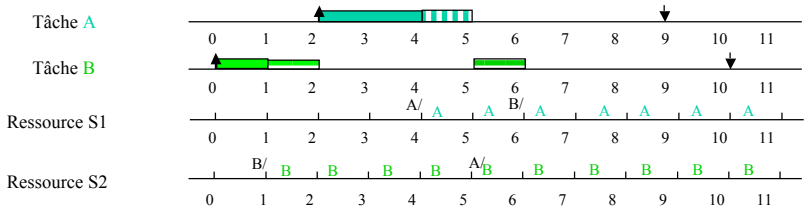
Mars pathfinder 1997, robot Sojourner

- Analyse
  - Exécution en mode trace (nombreuses heures) pour retrouver l'erreur
  - Test en « boite blanche »
  - Erreur détectée lors des tests, à terre, une deux fois dans les mois précédant la mission => la conclusion était alors d'incriminer l'électronique (contacts, …)
  - Partie non critique du robot donc non totalement validée
- Correction
  - Le code étant embarqué et statique, estimation de l'adresse mémoire du Semaphore Control Block (SCB) responsable
  - L'interpréteur C étant accessible, téléchargement d'un programme C de modification du paramètre du sémaphore via son SCB)
- Conclusion
  - Mauvaise politique de gestion des priorités
  - Pas d'analyse a priori
  - Pas de politique de reprise sur faute
  - Code modifiable

jean-philippe.pbabau@insa-lyon.fr

Département Informatique INSA de Lyon

Interblocage (deadlock), simulation



Priorité(A) > priorité(B)

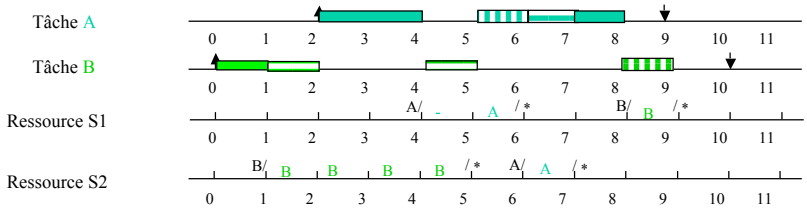
Sections critiques

jean-philippe.pbabau@insa-lyon.fr

Département Informatique INSA de Lyon

Protocole à priorité plafond

- principes
  - priorité plafond : maximum des priorités des tâches pouvant accéder à la ressource
  - entrée en section critique si : **priorité (tâche) > priorité plafond (ressource occupée)**
  - temps de blocage : maximum des durées des sections critiques de tâches moins prioritaires
  - gestion dynamique plus difficile (versions dégradées de PCP)



Priorité(A) > priorité(B)

Sections critiques

Priorité plafond (S1) = Priorité plafond (S2) = Priorité(A)

jean-philippe.pbabau@insa-lyon.fr

Prévention de l'interblocage

- Analyse de code
- Politique d'ordonnancement
  - Protocole à priorité plafond (Ada 95)
  - Protocole à super priorité (passage en mode non préemptif vis-à-vis de l'application)
  - Highest Locker Protocol (HLP), Stack Resource Protocol (SRP)
- Limitations des schémas de programme
  - prise/libération atomique des ressources
    - début … fin de chaque tâche
    - P(R1,R2, ...) … V(R1,R2)
  - prise/libération dans l'ordre des ressources
- Timeout (!)

jean-philippe.pbabau@insa-lyon.fr

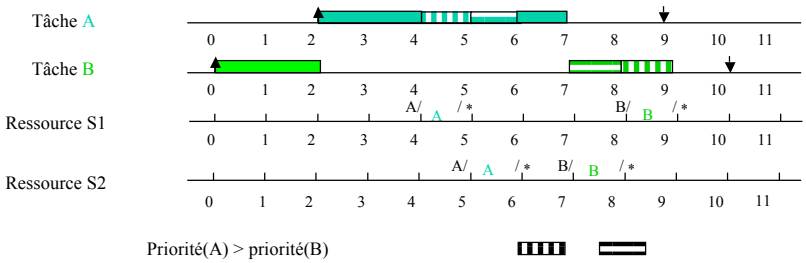
Temps de réponse

- Retard maximal en FIFO (sans deadlock !)
  - Durées des sections critiques des tâches
  - + préemption par toutes les tâches de priorité intermédiaire
- Retard maximal avec PHP (sans deadlock !)
  - Durées des sections critiques des tâches
- Retard maximal avec PCP pour une tâche Ti de priorité Pt
  - Construire l'ensemble des tâches Tj pouvant retarder la tâche
    - Priorité plafond des ressources prises par Tj >= Pt
  - Plus longue section critique des tâches Tj
- Exemple
  - Algorithme DM, A et C, partage une ressource pendant toute leur exécution
  - Analyse RMA pour C (gestion FIFO) :
  - Analyse RMA pour C (gestion PHP) :
  - Analyse RMA pour C (gestion PCP) :

jean-philippe.pbabau@insa-lyon.fr

Simulation de l'exemple avec interblocage (deadlock)

avec durée(B) supérieure



Priorité(A) > priorité(B)

➔ Pas d'interblocage !!

Validation temps réel en présence de ressources partagées

- Exécution cyclique
  - Simulation avec WCET -> séquence
  - Exécution selon la séquence (attente si nécessaire)
  - Validation : vérification de la séquence
- Ordonnancement en-ligne
  - une simulation, même avec le WCET, n'est pas une preuve
  - Nécessité d'évaluer le temps de blocage noté B
  - PCP
    - Pire durée de section critique d'une tâche de priorité inférieure

Prise en compte des ressources critiques

- Validation
  - Analyse du temps de réponse
    - via la simulation **: uniquement pour un séquenceur**
    - ordonnancement en ligne : estimation des pire temps de blocage
- Gestion des ressources et des tâches
  - Attribution de la ressource : libre ou priorité plafond
  - Gestion des tâches bloquées : FIFO, ordonnée selon la priorité
  - Priorité de la tâche en section critique : priorité simple, héritage simple
- RTOS
  - VxWorks : PHP

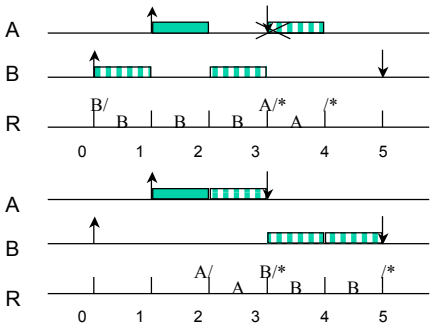
semId = semMCreate(SEM_Q_FIFO ou SEM_Q_PRIORITY |SEM_INVERSION_SAFE | SEM_DELETION_SAFE)

  - XX : PCP

jean-philippe.pbabau@insa-lyon.fr

Limites des algorithmes

- Ordonnancement avec ressources critiques NP complet
- RM,DM,ED non optimaux avec les ressources

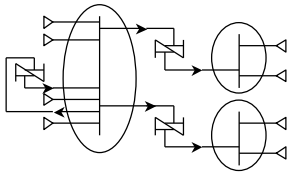


ri = 1, Ci = 2 (1,1,0), Ri = 2

ri = 0, Ci = 2 (0,2,0), Ri = 5

ri = 1, Ci = 2 (1,1,0), Ri = 2

ri = 0, Ci = 2 (0,2,0), Ri = 5

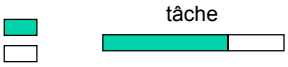


- Schémas de programmes limités
- Pas d'optimalité en cas de surcharge

jean-philippe.pbabau@insa-lyon.fr

Politiques de tolérance aux fautes

- Pas d'algorithme optimal en cas de surcharge
- Parties obligatoire/optionnelle d'une tâche
  - Ordonnancement des parties obligatoires
  - Ordonnancement des parties optionnelles, si possible



- Niveau d'importance
  - En cas de surcharge : suppression des tâches selon le niveau d'importance et les contraintes de précédence
  - Niveau d'importance dynamique
    - Suite à une suppression, augmentation du niveau d'importance
- Niveau de qualité
  - une requête : deux tâches
    - une «courte» de qualité «faible» durée connue : WCET
    - une «longue» de qualité «correcte» durée estimée : OCET
  - on tente la «longue» et on arme une alarme
  - si échec (alarme), on lance la courte
  - alarme ajustée afin d'être sur de pouvoir exécuter la «courte»

jean-philippe.pbabau@insa-lyon.fr

Autres contraintes

- Contraintes de précédence
- Contraintes de gigue
  - Contrôle de procédé
  - Traitement du signal
- Consommation
  - Processeur à fréquence variable

jean-philippe.pbabau@insa-lyon.fr

Ordonnancement en distribué

- Architecture multiprocesseur
  - Nouveau modèle de tâches
    - Migrante / non migrante
  - Techniques d'ordonnnancement étendues ou spécifiques
    - ED non optimal, Least Laxity, ...
- Ordonnancement des messages
  - Techniques d'ordonnnancement étendues ou spécifiques
    - Échéance
    - Protocoles prédictible en temps
- Allocation des tâches

jean-philippe.pbabau@insa-lyon.fr

Dérivation des contraintes

- Contraintes bout en bout « end-to-end deadline »
- Dérivation des délais - multi processeur
  di* = Min(dj*succ-Cjsucc-Ctransm)
- Intérêt
  - réaction aux fautes temporelles
  - homogénéité des délais

jean-philippe.pbabau@insa-lyon.fr

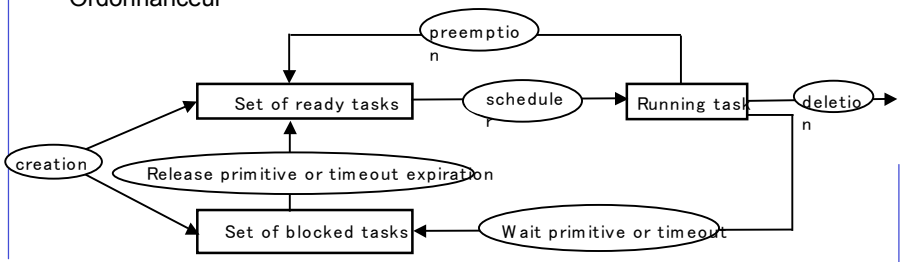
Approche par recherche d'une séquence optimale

- Principes
  - modélisation des contraintes (précédence, exclusion mutuelle, délais)
  - évaluation de toutes les séquences possibles correctes
  - optimisation de critères
    - minimum du nombre de changements de contexte
    - taux de réaction minimum
    - temps creux répartis uniformément
- Avantages
  - plus efficace que les algorithmes non optimaux
  - permet d' «ajuster» la séquence
  - Prédictible
- Inconvénients
  - algorithmes de recherche parfois complexes
  - ordonnancement statique

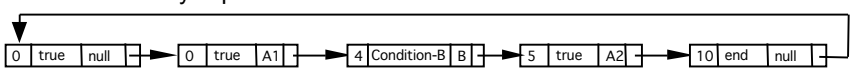
jean-philippe.pbabau@insa-lyon.fr

en-ligne Vs hors-ligne

Ordonnanceur



Exécution cyclique



jean-philippe.pbabau@insa-lyon.fr

en-ligne Vs hors-ligne

- hors-ligne
  - Une tâche n'est qu'un concept de conception
  - Le RTOS n'est pas nécessaire
  - Prédictible
  - Domaines spécifiques (certification avionique, traitement du signal, automatique)
  - Facile à implémenter
- en-ligne
  - reconfiguration
    - une tâche peut être activée/ supprimée
    - différent modes d'exécution
  - Temps réel
    - priorités dynamique
  - Réaction aux fautes
- en-ligne / hors-ligne
  - L'ordonnanceur gère un ensemble de séquences
  - L'ordonnanceur gère un ensemble de dates d'activation (OSEKTime, TTA)

jean-philippe.pbabau@insa-lyon.fr

Simulation exhaustive / RMA

- Simulation exhaustive
  - Description et simulation de l'ensemble des chemins possibles d'exécution
  - Adapté aux comportements dynamiques
  - Réalisme dans l'analyse
- RMA
  - Pas d'explosion combinatoire
  - Preuve dans le cas de ressources partagées
- Approches mixtes
  - Extraction des paramètres d'un modèle exhaustif pour une analyse RMA
  - Ajout du retard Bi dans un modèle exhaustif

jean-philippe.pbabau@insa-lyon.fr

Analyse temps réel

- Connaissance des paramètres temporels
  - pire durée d'exécution
- Analyse par mode de fonctionnement / d'exécution
- Pour chaque tâche
  - respect des délais : RT ≤ R
    - Sinon pourcentage de non respect, taux de dépassement
  - taux de réaction (min, moyen, max) : moyenne (RT)
  - temps de retard admissible : R – WRT
    - Prise en compte de tâches apériodiques
- Pour chaque ressource
  - taux d'utilisation
- Pour la séquence
  - charge, charge dynamique : somme (C / R)
  - répartition des temps creux
  - taux de réaction moyen : moyenne(RT)
  - temps de retard admissible minimum : minimum(R – WRT)

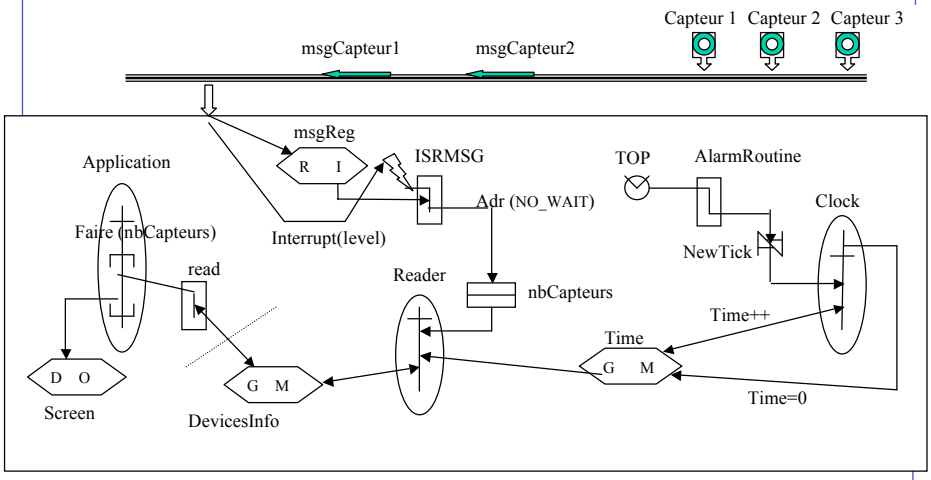
jean-philippe.pbabau@insa-lyon.fr

Plan

- Introduction
- Ordonnancement et temps de réponse
  - Tâches indépendantes
  - Serveurs, ressources partagées
  - Autres contraintes
- Exemple
  - Suivi de capteurs en temps réel
- Estimation de la pire durée d'exécution

jean-philippe.pbabau@insa-lyon.fr

Exemple : suivi de capteurs en temps réel



nbCapteurs = 16

jean-philippe.pbabau@insa-lyon.fr

Exemple : paramètres temporels

- Fréquences d'activation
  - msgCapteurs : 1 ms
  - msgCapteurs / capteur : 100 ms
  - TOP : 10 ms
- Pire durées
  - C1 pour Clock, C2 pour Reader et C3 pour Application
- Pire durée d'utilisation des ressources
  - Clock / Time = C11, Reader /Time = C21, Reader / DevicesInfo = C22
  - Application (par appel de la fonction read) / DevicesInfo = C32.
- Contraintes temps réel
  - Pas de perte de message
  - Datation précise à 20 ms
  - Toutes les informations sont affichées en 100ms (en 5 ms ?)

jean-philippe.pbabau@insa-lyon.fr

Exemple : inéquations

- Échéances
  - ISRMSG :
  - AlarmRoutine :
  - Reader :
  - Clock :
  - Application :
- Priorités
  - RM :
  - DM :
- Analyse des temps de réponse
  - ISRMSG :
  - AlarmRoutine :
  - Reader :
  - Clock :
  - Application :

Plan

- Introduction
- Ordonnancement et temps de réponse
  - Tâches indépendantes
  - Serveurs, ressources partagées
  - Autres contraintes
- Exemple
  - Suivi de capteurs en temps réel
- Estimation de la pire durée d'exécution

Pire durée d'exécution

- Objectifs
  - estimer la pire durée d'exécution
    - Worst Case Execution Time WCET
  - estimer une durée moyenne ou optimiste (OCET)
  - estimer le WCET en fonction du mode de fonctionnement
- Définitions
  - déterminisme (temporel) : le comportement d'un système ou d'un sous-système est dit déterministe si ses performances temporelles sont toujours identiques indépendamment de son contexte d'exécution.
  - prédictibilité (temporelle) : un système (ou sous-système) est dit prédictible si il est déterministe et qu'il est possible, a priori, de déterminer ses performances temporelles.

jean-philippe.pbabau@insa-lyon.fr

Département Informatique INSA de Lyon

Pire durée d'exécution

- Restrictions sur le matériel
  - durées d'exécution des instructions fixes ou bornées
  - pas de cache
  - pas de pagination
  - pas de swapping
  - pas de pipeline
- Restrictions sur le logiciel
  - pas de récursivité
  - pas de déclarations dynamiques
  - pas de goto, break, etc.
  - pas de float, long
  - appels explicites des procédures ou fonctions
  - limitation des boucles
    - durée bornée
    - nombre d'itérations borné

jean-philippe.pbabau@insa-lyon.fr

Pire durée d'exécution

- Principes (majorant de Ci)
  - (1) instruction : T (I) = T_I
  - (2) séquence : T (A;B) = T(A) + T(B)
  - (3) alternative : T (A ou B) = Max (T(A), T(B))
  - (4) répétition : T (maxi N fois A) = N X T(A)
- Principes réalistes (Ci)
  - (1') : T (I) ≤ T_I addition des réels, mémoire cache
  - (2') : T (A; B) ≤ T(A) + T(B)
  - (3) : T (A ou B) = Max (T(A), T(B))
  - (4') : T (maxi N fois A) ≤ N X T(A)
- (2') et (4')
  - chemins inaccessible
  - comportement du procédé
  - code mort
  - analyse fine des chemins réalistes

jean-philippe.pbabau@insa-lyon.fr

Extensions

- Analyse réaliste
  - Intégration d'informations sur l'application
    - Limitations du nombre des comportements possibles
- Logicielles
  - Langages de haut niveau (C)
    - Prise en compte du compilateur
  - Allocation dynamique : gestion prédictible de la mémoire
    - Gestion de la mémoire par blocs
    - Allocateur prédictible : activation périodique
- Matérielles
  - Mise en place de composants spécialisés
    - Cache partitionné
  - Analyse fine des composants matériels (pipe-line, cache)

jean-philippe.pbabau@insa-lyon.fr

Conclusion

- Politique d'ordonnancement
  - Affectation des priorités
  - Prise en compte de contraintes diverses (activation, synchronisation, distribution, fautes et surcharges)
- Modèle de tâches
  - Période, dmin, précédence, exclusion mutuelle, WCET
- Ordonnancement
  - Priorités, en-ligne / hors-ligne
- Analyse d'ordonnancement
  - Pire temps de réponse
    - Pire durée d'exécution + retard dû au partage de ressources
- Prédictibilité
  - Durées d'exécution, blocage sur ressources

➔ Impact sur la conception et l'implémentation

jean-philippe.pbabau@insa-lyon.fr