MI11

Systèmes temps réel critique

Jérôme De Miras

Poste : 59 02
e-Mail : demiras@hds.utc.fr

1

Modélisation de systèmes temps-réel

2

Modélisation

- Concevoir un modèle conceptuel
- Mettre en évidence ce qui est pertinent et ce qui ne l'est pas

3

Abstraction appropriée

MI11 utc 4

4

Abstraction

- Modèle : représentation de la réalité qui permet de maitriser le problème
- Modèle conceptuel : ensemble de concepts bien définis et de leurs relations
- Un modèle plus simple est plus facile à utiliser
  - Attention à la simplification excessive
- Deux hypothèses pour modéliser un système TR
  - Hyp de charge : un temps de réponse ne peut être garanti si on dépasse la charge
  - Hyp de faute : on ne peut être tolérant qu'à ce qui a été prévu

MI11 utc Printemps 2017 5

5

Ce qui est pertinent

- Notion de temps physique
  - Éléments temporels définis en fonction de la base de temps
  - Horloges synchronisées avec une précision Π
  - Granularité suffisante pour l'application
- Durée des actions
  - L'exécution d'une instruction (ou groupe) constitue une ***action***
  - 4 quantités pour décrire une action ***a***
    - Durée actuelle : $d_{act}(a,x)$ (x : entrées)
    - Durée minimale : $d_{min}(a)$ (pour toute entrées possibles)
    - Temps d'exécution au pire cas $d_{wcet}(a)$ (pour toute entrées, charge, faute)
    - Gigue : $d_{wcet}(a) - d_{min}(a)$
- Fréquence d'activation : nombre d'appel de *a* par unité de temps
  - Ressources limitées ⇒ contrôle stricte de la fréquence et de la distribution temporelle des activations

MI11 utc Printemps 2017 6

6

Ce qui n'est pas pertinent

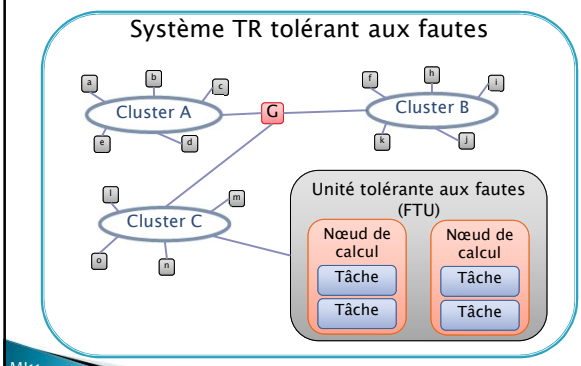
- Représentation
  - Focus sur les propriétés temporelles et le sens des variables, pas leur représentation
  - Une interface abstraite donne une valeur dans une unité commune à toutes les quantités de même type
    ⇨ interface de transformation sur passerelles
- Détails sur la transformation des données
  - Un algorithme a les caractéristiques suivantes
    - Données d'entrées
    - État interne
    - Résultats attendus
    - Modification de l'état interne
    - Ressources nécessaires
  - Du point de vue temporel, seul le pire cas d'exécution est pertinent
  - Logique interne et résultats intermédiaires sont inutiles au modèle conceptuel

7

Eléments structuraux

8

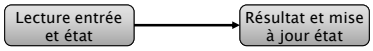
Structure globale d'un système TR



9

Tâches

- Programme séquentiel exécuté

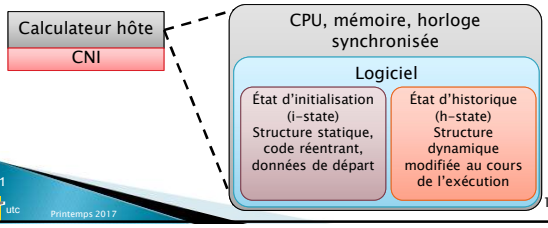


  - Le système d'exploitation déclenche la tâche
  - La tâche a une durée $d_{act}(task,x)$ sur une cible donnée
    - Stateless task : pas d'état interne au point d'invocation
    - Task with state : état interne existant
- Tâche simple (S-task)
  - Pas de point de synchronisation
  - Temps d'exécution non dépendant des autres tâches
- Tâche complexe (C-task)
  - Contient des instructions bloquantes de synchronisation
  - Le pire cas dépend du contexte global

10

Nœud – Composant

- Élément de calcul auto contenu (avec son propre matériel et logiciel)
- Élément d'abstraction le plus important
  - Une unité avec un comportement dans les domaines des valeurs et temporel
  - Ses messages le caractérisent pour le réseau
- Plus petite unité remplaçable (SRU)
  - Spécifier précisément les interfaces (temps et valeurs) base du diagnostique des fautes



11

Unité tolérante aux fautes (FTU)

- Abstraction introduite pour implémenter la tolérance aux fautes par duplication active
- Nœuds redondants avec messages répliqués déterministes
  - Mêmes résultats
  - Mêmes points dans le temps
  - Si un résultat est faux : mécanisme de vote
- Un FTU est un nœud unique du point de vue logique et temporel

12

Cluster de calcul

- Ensemble de FTU produisant un service tolérant aux fautes
- Service fourni à l'environnement du cluster
  - Opérateur
  - Objet contrôlé
  - Autres clusters
- Relié aux autres clusters par des nœuds passerelles

13

Interfaces

14

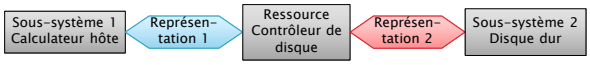
Design d'interfaces

- Fournir une abstraction utilisable par les partenaires (sans détails inutiles)
  - Propriétés de contrôle : signaux de contrôle qui traversent l'interface
    - Quelle tâche activer sur quel événement
    - Contraintes à satisfaire par les signaux de contrôle et les données qui traversent
  - Fonction attendues
    - Spécification de ce que les partenaires attendent
  - Propriétés des données
  - Structure et sémantique des éléments qui traversent l'interface

15

Exemples

- Extracteur de fumée
  - Fonction attendue : extraire la fumée en respectant le standard *lambda*
  - Si le standard change, le code peut changer mais la fonction attendue reste la même
- Interface disque : dans certains cas les partenaires utilisent des codages différents
  - L'interface transforme le codage : interface intelligente ou contrôleur de ressource



16

Composants d'interfaces

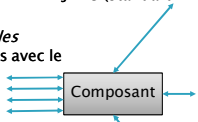
- Fin de phase de design : obtention d'un modèle indépendant de la plateforme (PIM)
  - PIM : modèle exécutable qui partitionne le système en clusters et composants
    - Contient la spécification précise des l'interfaces
    - (Domaine temporel et valeurs)
    - Pas d'hypothèse sur l'implémentation des composants
  - PSM : Après transformation pour une exécution sur la plateforme finale, on obtient modèle spécifique à la plateforme
  - Les caractéristiques d'interfaces d'un PIM et d'un PSM sont les mêmes

17

Les 4 interfaces messages

TDI : *Technology Dependent Interface*
(pour diagnostique interne)
Ex : JTAG (Standard IEEE 1149.1)

IL : *Interfaces locales*
Ex : entrées/sorties avec le monde physique. Mécanismes non spécifiés au niveau d'abstraction utilisé pour le cluster



LIF : *Linking Interface*
(pour la composition des composants)
Services spécifiques du composant indépendants de la réalisation du composant

TII : *Technology Independent Interface*
(Configuration composant et management des ressources)
Ex : reconfiguration d'un composant FPGA
Interface orthogonale au LIF (pas d'accès à la couche application)

18

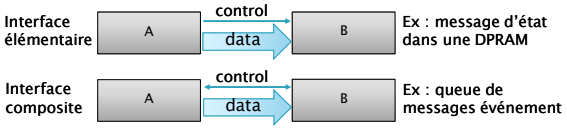
Caractérisations des interfaces : Push, Pull

- PUSH : le système de communication provoque une interruption pour forcer la prise en compte du message immédiatement.
  Comportement temporel subit par le composant
- PULL : stockage du message dans une variable partagée.
  Vérification périodique par le composant pour récupérer le message.
  - Sur un RTS toujours préférer le PULL.
  - Réserver le PUSH au cas où un cycle d'attente n'est pas envisageable avant la prise en compte du message. Penser à la protection contre les interruptions dues à une faute extérieure

19

Caractérisations des interfaces : Elémentaire, Composite

- Elémentaire: situation où un lien simple unidirectionnel est suffisant pour le flot de contrôle et le flot de donnée
- Composite : scénario avec un flot de donnée unidirectionnel et un flot de control bidirectionnel

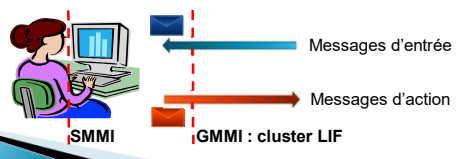


- Une interface élémentaire est plus simple à intégrer dans un RTS car le comportement du récepteur n'a pas à être pris en compte pour vérifier l'intégrité de l'émetteur

20

Interfaces passerelles (Gateway)

- Système dédié à 2 interfaces (*Composant ouvert*)
  - EX : Interface homme machine
    - Interface spécifique entre l'opérateur et la machine (SMMI)
    - Interface généralisée entre le système HM et le reste du système distribué (GMMI)
  - Ce qui est pertinent correspond aux propriétés temporelles et données au niveau GMMI



21

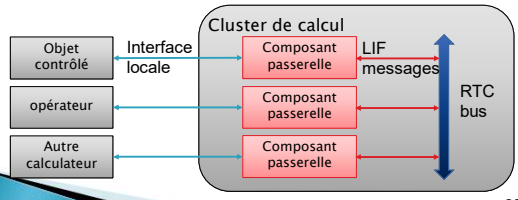
Interfaces : caractéristiques

caractéristique	Interface locale	Interface LIF
Représentation de l'information	Unique, déterminée par le périphérique	Uniforme dans l'ensemble du cluster
Couplage	Serré, déterminé par le protocole I/O spécifique du périphérique	Plus faible, déterminée par le protocole de communication du message
Codage	Analogique ou numérique unique	Numérique, codes uniformes
Base de temps	Dense	Éventuellement clairsemé
Modèle d'interconnexion	Un pour un	Un pour plusieurs
Liberté de design	Déterminée par le format et le timing des périphériques physiques	Déterminée par le standard uniforme de l'architecture

22

Interfaces uniformes

- Il est important que les sous-systèmes ne soient pas reliés par des interfaces cachées
  - Peut arriver dans le système contrôlé
- La représentation doit être uniforme dans un cluster
  - Contrôleur de ressources



23

Interfaces messages standardisées

- Existent pour simplifier la connexion de périphériques I/O
- Exemples
  - MAP : Manufacturing Message Specification
  - Norme SAE J1587
    - Véhicules lourds :
      - Nom de message
      - Nom de paramètres
      - Taille des variables
      - Fréquences
      - …

24

Spécification Interfaces

- Le comportement d'une interface (d'un composant) est déterminée par les messages qui la traverse
- Spécifications des messages :
  - Spécifications de transport (formelles)
  - Spécifications opérationnelle (formelles)
  - Méta-spécifications (naturelles)

25

Spécifications de transport

- Propriétés nécessaires pour amener le message d'un émetteur à un(des) récepteur(s) sur le système de communication
  - Pas de définition du contenu du champ de donnée (champ de bits non structuré)
  - Mais des attributs pour :
    - Sens et adresse du port utilisé
    - Taille du champ de donnée
    - Cycle pour les messages TT
    - Taille de la queue pour les messages ET ou le streaming

26

Spécifications opérationnelles (1)

- Définir la nature du champ de bits correspondant aux données
  - Définition des variables message (partie fixe et partie variable)
  - La structuration est définit dans un MSD (Message-structure déclaration)
    - contient les noms des variables
    - Définit la position de la valeur dans le champ de bits
    - Des assertions sur la validité des valeurs (ex : intervalle) (en entrée et en sortie)

27

Spécifications opérationnelles (2)

- Généralement la structure des messages est statique
  Le MSD est stocké dans le système de communication
- Association :
  - Nom MSD = nom du port d'entrée
    un seul type de message par port
  - Nom MSD inclus dans le message
    différents messages peuvent être reçus sur le même port (ex : CAN)
  - Nom MSD assigné à un instant du cycle temporel TT
    différents messages peuvent être reçus sur le même port sans envoi du nom MSD (ex : TTP)
  - Nom MSD stocké sur un serveur accessible par un récepteur (ex : CORBA)
  - Le MSD est inclus dans le message ; flexible mais couteux en bande passante (ex : Architectures Orientées Services)

28

Meta-spécifications

- Définition de la signification des variables message : interopérabilité sémantique
- Modèle de service LIF : définit les concepts associés avec les noms des variables message
  - Composant fermé : peut être formalisé car pas de comportement imprévisible dû à l'environnement
  - Composant ouvert :
    - non définissable sans le contexte d'utilisation
    - Description à faire dans la sphère de compréhension de l'utilisateur
      - Utilisateur orientée : concept dans le savoir commun d'un utilisateur du domaine
      - Objectif orientée : quel solution va apporter le composant
      - Vue système : quel effet sur le système à travers le composant

29

Obligations Temporelles clients et serveurs

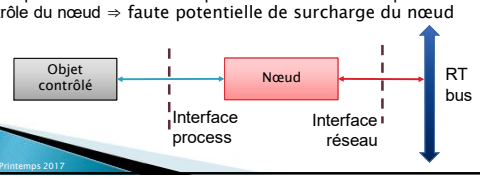
- Réponse :
  - Modification d'état
  - Message de réponse
- Paramètres temporels
  - Temps de réponse maximum (RESP)
    attendu par le client et prévue dans les spécifications
  - Le pire cas d'exécution (WCET)
    - Sphère de contrôle serveur
  - Le temps minimum entre deux requêtes successives (MINT)
    - Sphère de contrôle client

30

OT clients et serveurs : conditions

- Temps réel : WCET < RESP
  sous la condition que MINT soit suffisant
- Si WCET ≪ RESP ➡ surdimensionnement
  rare en embarqué
- Si WCET ≃ RESP ➡ nécessité d'une analyse pointue

Si les opérations sur l'interface process sont hors de la sphère de contrôle du nœud ⇒ faute potentielle de surcharge du nœud

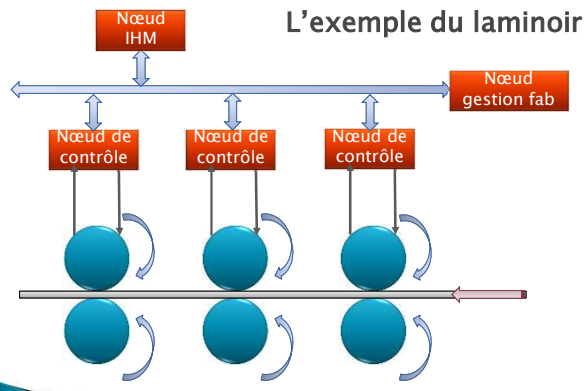


31

Contrôle temporel contrôle logique

32

L'exemple du laminoir



33

Retour sur l'exemple du laminoir

- 3 pressions mesurées indépendamment
- Spécification de l'alarme sur le MMI :

```
when (p1 < p2) ∧ (p2 < p3)
     then OK
     else ALARME
```

- Implémentation :
  - 4 S-Tasks
  - 4 nœuds

34

Retour sur l'exemple du laminoir

- Questions :
  - Quel est le temps tolérable entre l'arrivée du défaut et sa détection ?
  - Quel est le temps tolérable entre les mesures des trois pressions ?
    - Si non respect : fausse alarme ou défauts perdus
  - Quand déclencher les tâches dans les nœuds de contrôle ?
  - Quand activer le tâche de monitoring du défaut dans le MMI ?
- Si pas de réponse, les spécifications sont incomplètes dans le domaine temporel

35

Cause de spécifications incomplètes

- Le « when » englobe deux interprétations
  - Le point dans le temps où l'alarme doit être prise en compte
  - La condition dans le domaine des valeurs à surveiller
- Contrôle logique : flot à l'**intérieur** d'une tâche
  - Structure du programme
  - Données d'entrée
- Contrôle temporel : points dans le temps où la tâche doit être activée ou bloquée
  - Dus à des conditions **extérieures**

36

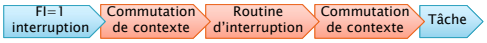
Activation ET ou TT

- Le signal d'activation d'une tâche peut avoir deux sources :
  - Changement d'état significatif (événement) dans l'environnement ou dans le système réparti
    - Boutons, commutation sur limites, arrivé d'un message, fin d'une tâche
  - Progression du temps : atteinte d'une valeur préprogrammée dans une table (horloge)
  - Exemple : bouton d'ascenseur → Interruption à chaque fois ; Mise en mémoire et échantillonnage
- Systèmes ET ⇒ flexible
- Systèmes TT ⇒ prévisible

37

Interruptions et activation

- Systèmes ET ⇒ mécanisme d'interruption
  - Requête supportée par un mécanisme matériel asynchrone déclenché par un événement extérieur au nœud
    - Les exceptions déclenchée par le flot de contrôle de la tâche n'entrent pas dans cette catégorie

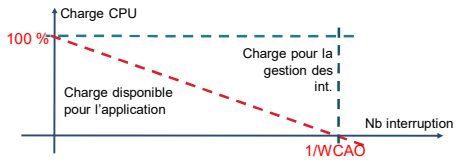


  - Interruption prise en compte qu'en des points particuliers de l'exécution processeur + commutation de contexte

⇒Pire cas supplément d'administration
  - (worst case administrative overhead : WCAO)

38

Interruptions et activation



- Il est essentiel de limiter le nombre d'interruption
- Difficulté : la source est hors de la sphère de contrôle du nœud
- De plus des interruptions peuvent être inutiles

39

Tâches trigguée TT

- Système TT :
  - Contrôle interne au systèmes
  - Capture périodique de l'état pour vérifier les changements
- Réalisation : une tâche périodique
  - Pb : ne perçoit que les états plus long que la période d'activation
  - ➡ mise en mémoire des changements dans l'interface
  - Engendre une charge processeur fonction de la période

Période < laxité d'une opération RT due à un événement de l'environnement

Laxité : différence entre deadline et temps d'exécution

40

Activation

Caractéristique	Interruption	Tâche trigguée TT
Facteur influençant la latence	Réponse à l'interruption	Période d'échantillonnage
Source du contrôle	Externe au nœud	interne
Charge CPU par répétition	aucune	WCET/laxité
Charge CPU d'administration	variable	constant
Préemption	À tout moment	contrôlée
Élément mémoire	interne	externe
Condition de trigg	simple	complexe
prévisibilité	basse	haute

41

Pire cas d'exécution

42

Pire cas

- Une deadline pour effectuer une opération TR peut être garantie uniquement si les WCET de toutes les tâches de l'application entrant dans l'opération sont connus à priori
- En plus du WCET, il faut connaitre le WCAO
  - WCAO : délai qui affecte l'application mais hors du contrôle de la tâche

43

WCET d'une S-Task

- Tâche la plus simple
  - Séquentielle
  - Matériel dédié
  - Pas de préemption
  - Pas de recours aux services système d'exploitation
- WCET dépend de :
  - Code source de la tâche
  - Propriétés du code objet généré par le compilateur
  - Caractéristique du matériel cible

44

Analyse du code source

- Objectif : connaître le temps d'exécution d'un programme écrit dans un langage de haut niveau sachant le temps de chaque instruction
- A priori non soluble

```
S: while(exp)
   do loop;
```

Combien d'itérations ?

- Restrictions pour obtenir un problème traitable :
  - (Puschner and Koza : 1989)
  - Absence de déclaration de contrôle sans borne au début d'une boucle
  - Absence de fonction récursives
  - Absence de structure de données dynamiques

45

Borne WCET code source

- Borne WCET pour chaque construction de base
- Borne WCET pour chaque construction
- Caractéristique temporelle d'un programme

```
S: if(exp)
   then S1
   else S2
```

T(S) = max(T(exp)+T(S1), T(exp)+T(S2))

WCET de chacune des structures

46

Borne WCET code source

- Les formule de ce type pour raisonner sur le comportement temporel sont appelé ***schéma temporel***
- Le plus long chemin est appelé ***chemin critique***
- Le nombre de chemins augmente exponentiellement avec la taille du programme
  - La recherche peut devenir intraitable
  - Recherche guidée par réduction de l'espace de recherche (exclusion des chemins impossibles)
- Recherche de chemin peut être transformée en problème de programmation linéaire sous contraintes
  - Contraintes de la structure programme dérivées du flot de contrôle
  - Contraintes fonctionnelles données par le programmeur
    - Ex : annotations interprétables par un outil d'analyse

47

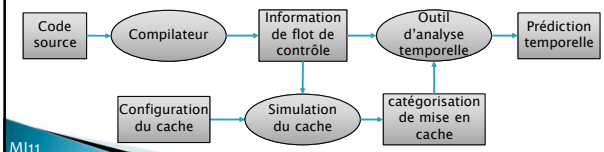
Analyse du compilateur

- Déterminer le temps maximum d'exécution des constructions de base sachant le temps maximum du langage machine
- Construction d'un arbre temporel au cours de la compilation en utilisant la relation
  code source-code objet
- Les processus de compilation (optimisations) doivent être analysés
  - On peut les interdire pour assurer la traçabilité

48

Timing de l'architecture matérielle

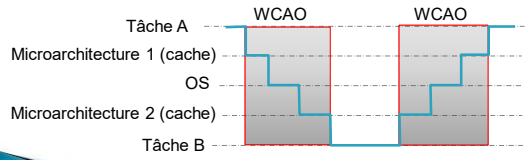
- Pire cas d'exécution de chaque instruction assembleur
  - Si temps fixe d'exécution : table de correspondance
  - Si architecture moderne (pipe, cache) : introduction d'imprévisibilité
    - Méthodes par simulation



49

S-Task préemptive

- Une tâche simple préemptée par une tâche indépendante voit son temps augmenté par :
  - Le WCET de la tâche qui interrompt
  - Le WCET du système d'exploitation
  - Le temps nécessaire au rechargement du cache



50

WCET d'une tâche complexe

- Le WCET d'une C-Task préemptive avec accès à des ressources :
  - Dépend de la tâche elle-même
  - Dépend des autres tâches et de l'OS
- Problème global au sein d'un nœud
  - Délais dus à l'interaction avec le reste du système
    - Exclusion mutuelle
    - Précédence

➡ Travaux en ordonnancement

51

Etat de la pratique

- Borne d'une S-Task :
  - restriction sur le code
  - Annotation du code source par le programmeur
  - Cartographie entre code source et annotations afin d'analyser le résultat
- Tout est encore à faire progresser
  - Outils permettent l'analyse WCET de codes (Ex : RAPITA Systems)
  - Outil graphique de spécification
  - Compilateurs certifiés
  - génération automatique de code

52

Eléments d'action

- Mesurer sur une implémentation pour recueillir des données expérimentales
- Minimiser les interactions entre tâches pour simplifier l'analyse à priori
- Analyser les sous-problèmes (temps d'exécution des programmes sources) pour générer un ensemble de test qui peuvent conduire au pire cas
- Tests en grands nombre sur la réalisation complète

53

Etat d'historique

54

H-state

- Structure de données dynamique dont le contenu évolue au cours de la progression des calculs (sur un nœud)
- Concept essentiel pour la restauration du contexte sur faute

- Ex : calculatrice

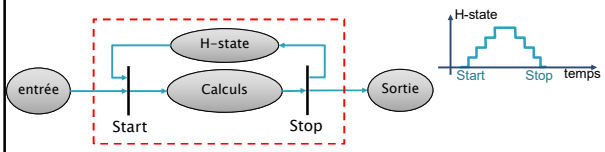
  Nombre en entrée → calcul d'un sinus → sortie du résultat

  Si cette opération est atomique et qu'on observe le système avant ou après, le h-state est vide

55

H-state : calculatrice

- Si on regarde entre « start » et « stop », on va avoir toutes les étapes du calcul du sinus



- En un point du temps le compteur de programme et tout le contenu de la mémoire forme le h-state
- A la fin du calcul les données sont non pertinentes et vidées

56

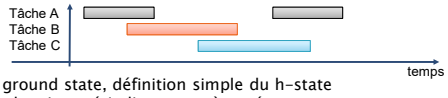
H-state et restauration

- Correspond au résultats intermédiaire du calcul
- Image telle qu'elle devrait être chargée dans un dispositif vierge pour reprendre une opération au point d'interruption
- La taille dépend du niveau d'abstraction et du moment d'observation
- En gérant la granularité de l'observation on peut minimiser la taille du h-state
  - ➡ Définition simplifiée d'un point de restauration après faute

57

Ground state

- Etat :
  - sans tâche active
  - Tous les canaux de communication sont vides
- Prenons la situation suivante
  - S-tasks concurrentes échangeant des messages
  - S-tasks atomiques
- Si toutes les tâches sont asynchrones, il se peut que le ground state ne soit jamais atteint



- Sinon au ground state, définition simple du h-state
- Si un nœud revient périodiquement à cet état :

Point de redémarrage

58

Relations temporelles

59

Objectifs

- Définir les relations temporelles entre les différentes parties d'un système cybernétique
  - Entité RT
  - Image RT
    - Validité temporelle
    - Permanence d'une observation
  - Déterminisme

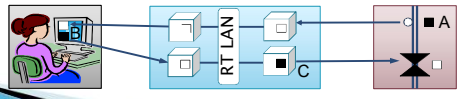
60

Entités, images et objets RT

61

Entités Temps Réel

- Variable d'état d'intérêt pour un objectif donné
  - Ex : débit d'un liquide, consigne de fonctionnement, position d'une vanne, …
  - Attributs statiques (nom, type, domaine de valeur,…)
  - Attributs dynamiques qui changent avec le temps
- Chaque entité appartient à une sphère de contrôle

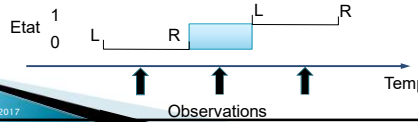


62

Observations

- Entités RT a un ensemble de valeurs
  - continu

  ou
  - discret
- Entités discrètes
  - Constante sur un intervalle entre un *left event* et un *right event*
  - Entre un *R_event* et un *L_event* la valeur de l'entité est indéfinie



63

Observations

- L'information sur une entité RT à un instant donnée est une observation
  - Structure de donnée atomique

    $observation = < Nom, t_{obs}, Valeur >$

  - Pour une entité discrète, l'observation doit se faire quand elle a du sens
  - Un nœud intelligent est relié au capteur pour fournir la date et le format numérique correct
  - Une observation = un seul message
    - Le concept de message fournit l'atomicité de l'information

64

Observations non datées

- Sans temps global une date n'est interprétable que dans la sphère de validité de l'horloge qui a datée
- Si on utilise une date d'arrivée hors de cette sphère comme t_{obs}, la datation est imprécise due à la latence et à la gigue du réseau
  - Réduction de la qualité de l'observation

65

Observations indirectes

- Il n'est parfois pas possible d'observer directement une entité RT
- Possibilité d'avoir des observations indirectes de l'entité
- Nécessité d'utiliser un modèle mathématique pour reconstruire la valeur de l'entité visée

66

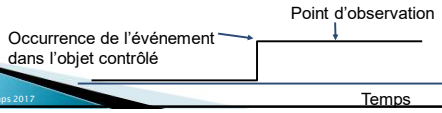
Observations d'état et d'événements

- Observation d'état :
  - L'observation contient l'état de l'entité RT échantillonné à un instant donné
  - Control : échantillonnage équidistant de l'état
  - Bonne correspondante observation d'état et sémantique de message d'état

67

Observations d'événement

- Un événement est localisé sur un instant
  - Correspond à un changement d'état
  - Observation = événement
    => impossible d'observer un événement dans l'objet contrôlé mais seulement son effet
  - Une observation d'événement contient l'évolution de valeur entre l'ancien et le nouvel état
    - Meilleur estimée de l'instant d'arrivé de l'événement
      - Comment obtenir l'instant précis d'occurrence ? ET ou TT ?
      - La perte ou la duplication d'une EO provoque une perte de synchronisation entre émetteur et récepteur (fiabilité ?)
      - EO envoyée uniquement sur un changement ; la latence de détection d'une faute ne peut pas être bornée, un récepteur suppose une non évolution si aucun message n'arrive



68

Images Temps Réel

- Image courante d'une entité RT
- Une image est valide à un instant donné si c'est une représentation exacte de l'entité correspondante
- une observation décrit un fait qui reste valide à jamais (une valeur à un temps donné)
  ≠
  la validité d'une image RT est dépendante du temps
- Construction à partir :
  - D'une observation d'état
  - D'une observation d'événement
  - D'une estimation d'état

69

Objets Temps Réel

- Conteneur à l'intérieur d'un nœud d'un système distribué pour une image RT ou une entité RT
- Une horloge est associée à chaque objet RT
  - Tick d'horloge => déclenchement d'une procédure associée à l'objet
- Un objet RT peut être distribué
  - Réplication dans plusieurs nœud afin de fournir un service spécifique localement
  - Contraintes de consistance pour la qualité de service
    - Ex : temps global est un objet RT distribué qui assure une précision Π
    - Ex : service d'appartenance ; le temps de prise en compte d'une évolution de l'état d'un membre par les autres est un critère important

70

Précision temporelle

71

Définition

- Définie en utilisant l'historique récent des observations de l'entité RT
  - Historique récent

$$RH_i = \{t_i, t_{i-1}, t_{i-2}, \ldots, t_{i-k}\}$$

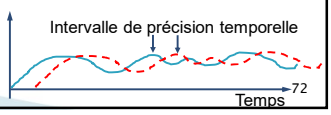
  - Intervalle de précision temporelle

$$d_{acc} = z(t_i) - z(t_{i-k})$$

  - Une image RT est précise temporellement si

$$\exists t_j \in RH_i : \quad Valeur(\ image\ RT\ à\ t_j\ ) = Valeur(\ entité\ RT\ à\ t_j\ )$$

  - La transmission d'une image RT se fait avec une latence qui doit être inférieure à d_{acc}



72

Intervalle de précision temporelle (1)

- La taille de d_{acc} est déterminée par la dynamique de l'entité RT
  - Le délai de transmission provoque une erreur

$$erreur\ (t) = \frac{dv(t)}{dt}\left(z(t_{use}) - z(t_{obs})\right)$$

  - Le pire cas est donné par

$$erreur\ = \left(\max_{\forall t} \frac{dv(t)}{dt} d_{acc}\right)$$

  - Doit être du même ordre que l'erreur de mesure faite sur l'entité RT
  - Plus une entité RT change rapidement de valeur, plus d_{acc} doit être petit

73

Intervalle de précision temporelle (2)

- Soit t_{use} l'instant d'utilisation du résultat d'un calcul faisant intervenir une image RT :

$$z(t_{obs}) \le z(t_{use}) \le \left(z(t_{obs}) + d_{acc}\right)$$

- ou encore

$$z(t_{use}) - z(t_{obs}) \le d_{acc}$$

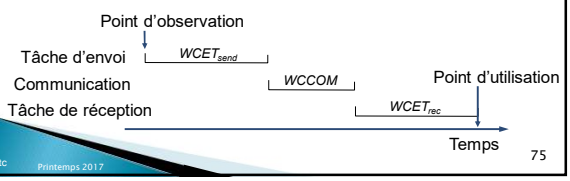
74

Transaction alignée en phase

- Acquisition, transfert, utilisation, chacune avec un pire cas de réalisation
  - Pire cas de la chaine (tâches synchronisées)

$$\left(t_{use} - t_{obs}\right) = WCET_{send} + WCCOM + WCET_{rec}$$

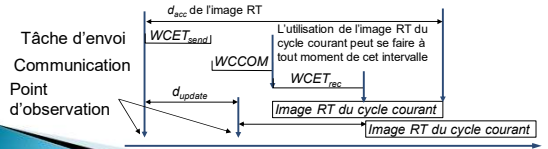
  - Si d_{acc} requis par la dynamique est plus petit que cela, nécessité de faire de **l'estimation d'état**



75

Classification des images RT

- Image paramétrique ou insensible à la phase
  - Image RT mise à jour périodiquement (d_{update})
  - Transaction alignée avec l'émetteur
  - Condition : $d_{acc} > (d_{update} + WCET_{send} + WCCOM + WCET_{rec})$
  - Un récepteur peut accéder à cette image sans considération de la relation de phase
  - En cas de réplication, les récepteurs doivent accéder à la même version de l'image RT pour assurer le déterminisme



76

Classification des images RT

- Image sensible à la phase (PSI)
  - Transaction alignée sur l'émetteur
  - Condition :

et $d_{acc} \leq (d_{update} + WCET_{send} + WCCOM + WCET_{rec})$

$d_{acc} > (WCET_{send} + WCCOM + WCET_{rec})$

  - Impose des contraintes supplémentaires sur l'ordonnancement des tâches qui utilisent ce type d'image RT
  - Bonne pratique : limiter l'utilisation des PSI
    - Réduire d_{update} (augmentation de la charge communication)
    - Modèle d'estimation d'état (augmentation de la charge processeur)

77

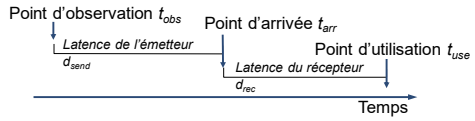
Estimation d'état

- Construction d'un modèle de l'entité RT à l'intérieur d'un objet RT
  - Prédiction de l'état futur probable et mise à jour de l'image RT
  - Exécution périodique pilotée par l'horloge associée à l'objet RT
  - Pour la prédiction on se base sur t_{use}
  - Nécessité de posséder un modèle de comportement (pas de modèle piloté par le hasard)
  - Un paramètre d'implémentation est l'intervalle $[t_{obs}, t_{use}]$ perçu par des nœuds différents
    => nécessité d'un système de communication avec une gigue minimum et un temps global précis
  - Une approximation d'ordre 1 peut être suffisante mais ce n'est pas toujours le cas

78

Composabilité

- L'intervalle $[t_{obs}, t_{use}]$ se compose comme suit



  - Dans une architecture TT tous ces intervalles sont connus à priori
- Estimation d'état dans le récepteur
  - Une modification du délai de l'émetteur modifie le traitement d'estimation et donc nécessite aussi une modification logicielle du récepteur
  - Pour réduire le couplage, découper l'estimation en 2 :
    - Prédire t_{arr} dans l'émetteur
    - Faire passer t_{arr} dans le récepteur pour t_{obs} pour le récepteur

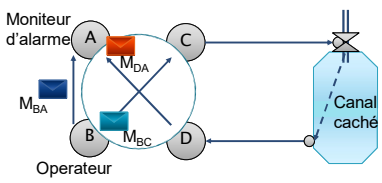
79

Permanence et idempotence

80

Permanence (1)

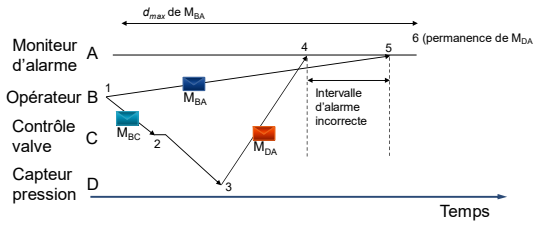
- Un message est permanent sur un nœud donné si tous les messages antérieur qui s'y rapportent sont arrivés ou n'arriveront jamais



81

Permanence (2)

- Protocole avec un temps minimum d'exécution d_{min} et maximum d_{max}, une gigue $d_{jit} = d_{max} - d_{max}$
- Le temps de réaction du canal caché inférieur à d_{min}



- Pour éviter toute fausse alarme, le nœud A doit retarder toute action jusqu'à ce que MBA devienne permanent

82

Permanence (3)

- Délai d'action
  - Pour un message donné, le temps entre le début de transmission et le point où ce message devient permanent sur le récepteur est appelé délai d'action
  - Un récepteur doit retarder toute action sur ce message avant qu'il ne devienne permanent
- Action irrévocable
  - Il s'agit d'une action dont l'effet ne peut être défait
  - Son action est durable sur l'environnement
  - Important de ne déclencher une action irrévocable qu'une fois que tous les délais associés à son déclenchement sont passés

83

Durée du délai d'action

- Dépendant
  - De la gigue du système de communication
  - De l'attention temporelle du récepteur
- Avec temps global
  - Transmission de la date dans le message
  - Le récepteur peut inférer que le message sera permanent à $t_{permanent} = t_{send} + d_{\max} + 2g_{GT}$
- Sans temps global
  - Attente de $d_{\max} - d_{\min}$ après l'arrivée du message
  - Permanence à $t_{permanent} = t_{send} + 2d_{\max} - d_{\min} + 2g_l$

84

Précision contre Délai d'action

- Une image RT ne peut être utilisée que si le message qui la transporte est permanent et si elle est temporellement précise
- Possible sans estimateur uniquement sur la fenêtre $[t_{permanent}, t_{obs} + d_{acc}]$
  - d_{acc} dépend de la dynamique de l'application de contrôle
  - $t_{permanent} - t_{obs}$ dépend de l'implémentation
- Si toutes les exigences ne peuvent être atteintes simultanément, l'estimateur reste la seule alternative

85

Idempotence

- Un ensemble de messages répliqués est idempotent pour un récepteur donné si la réception de plusieurs de ces messages provoque le même effet que la réception d'un seul.
  - Dans un système où des messages non datés sont envoyés
    - un message d'état est idempotent
    - Un message d'événement relatant une variation d'un l'état ne l'est pas
      (description de l'incrément de la variable, si prise en compte plusieurs fois, erreur permanente sur la valeur)

86

Déterminisme

87

Définition

- Si dans un système qui suit le principe de causalité on a des implications à la fois logiques et temporelles, on parle de déterminisme
- Un système physique est déterministe si en donnant son état initial à un instant *t* et un ensemble d'entrées futures, on peut prédire les états et sorties futurs
  - Définition supposant un temps dense (lois de la physique)

88

Base de temps clairsemée

- Dans un système numérique, la base de temps est clairsemée
- Hypothèse que les événements sont clairsemés
  - Spécification des propriétés temporelles comme la simultanéité
  - Nécessité de protocoles d'accord pour passer du temps dense au temps clairsemé aux interfaces avec le monde
  - Réduction de la fidélité du modèle numérique

89

L-déterminisme

- Un système est L-déterministe si en donnant un état initial et un ensemble ordonné d'entrées, on peut calculer les états et sorties suivantes
  - Pas de notion de futur
- Concept insuffisant dans un contexte temps réel
  - En plus d'assurer que les actions seront bien celles prévues, il faut assurer une borne temporelle de l'action
  - Ex : système de freinage

90

Pourquoi du déterminisme

- Une relation par implication entre état initial et état et sortie futures simplifie la compréhension du comportement d'un composant
- Deux composants répliqué qui démarre avec le même état initial et recevant les mêmes informations produiront les mêmes résultats aux mêmes moments
  - Essentiel pour masquer les fautes par un vote
- La testabilité d'un composant est simplifiée, tous les cas de test sont reproductibles
  - pas d'apparition d'erreur *Heisenbugs*

91

Propriété désirée

- Le déterminisme est une propriété désirée de comportement
- L'implémentation peut atteindre cette propriété avec une probabilité estimée
- Raison d'échecs
  - L'état initial n'est pas précisément connu
  - Le matériel tombe en panne sur un défaut physique imprévu
  - La notion de temps est obscure
  - Le logiciel contient des erreurs de design ou des constructions non déterministes (NDDC)

92

Déterminisme sur réplication

- L'état initial doit être consistent pour toute les répliques
  - Nécessité d'une base de temps clairsemée commune
    - datation consistante des événements
    - Définition de la simultanéité pour éviter un ordre temporel inconsistant (perte du déterminisme)
- Datation sur la base de temps clairsemée
  - par le système par génération d'événements
  - par un protocole d'accord pour assigner un événement à une date particulière de la base de temps
- Le système de communication est prévisible
  - Instants de livraison bornés
  - Conservation de l'ordre d'envoi sur tous les canaux
- Existence d'une notion précise du « temps réel »
- Les résultats de calculs sont certains
  - pas de NDDCs
  - obtention avant la fin d'une *fenêtre d'acceptation* connue

93

Etat initial consistant

- Existe si une séparation consistante entre événements passés et futurs peut être faites
  - Une base de temps globale clairsemée permet cette séparation
  - Sans cela, l'établissement d'un état initial dans des composants répliqués est difficile
- Un capteur peut faillir => redondance
  - Aucun capteur parfait : erreur finie de mesure
  - Les valeurs sont numérisées : erreur de discrétisation
  - ⇨Déviation dans la redondance de l'observation
- Protocoles d'accord pour attribuer à une mesure (distribuée à plusieurs répliques)
  - La même valeur consistante pour la mesure redondante
  - La même date sur la base de temps du système

Constructions non déterministes

- Partant d'un état initial correct,
- Les effets d'un NDDC peut être dans le domaine des valeurs ou du temps
  - Ne respecte pas l'hypothèse d'indépendance des fautes

Constructions ND domaine des valeurs

- Générateur de nombres aléatoires
- Éléments de langages non déterministes
  - Choix laissé à l'implémentation
- Points de décision majeures mal définis (sur un timeout par exemple)
- Ordonnancement préemptif
  - Prise en compte d'une interruption en des points différents du code de chaque réplique
- Ordre de messages inconsistant entre les différents canaux

Constructions ND domaine temporel

- Les constructions précédentes
- Préemption et blocage de tâches
  - Peut retarder un résultat au-delà de la fenêtre d'acceptation
- Mécanismes de ré-essai
  - Ajout d'un délai à la prise en compte d'un résultat correct
- Accès concurrents
  - Mécanisme conduisant à l'octroi d'un sémaphore
  - Accès à un média par priorité (CAN)

97

Récupération du déterminisme

- Dans un système L-déterministe, une perte de déterminisme peut être évitée en étendant la fenêtre d'acceptation
  - Réduction du risque d'un délai non respecté
  - Technique utilisée à un niveau macroscopique alors que le déterminisme a été perdu à un niveau microscopique
- Récupération du déterminisme au niveau externe
  - Par rapport aux services fournis par le composant
  - (voir le modèle des 4 univers d'un système informatique)

98

Sureté de fonctionnement

99

Préambule

- Dans un paradis technologique, aucun acte divin ne peut se produire et tout se déroule selon les plans

  Hannes Alfen, prix Nobel

- Le monde réel n'est pas un paradis technologique

100

Base

- Si on dispose d'un temps global clairsemé, tout phénomène adverse au niveau logique et au dessus peut être identifié
  - Configuration binaire spécifique au niveau valeur
  - Instant d'arrivée sur la base de temps

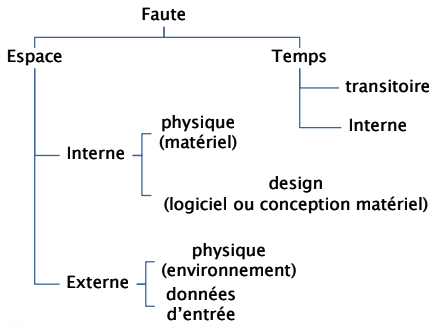
101

Fautes

- Un système est un assemblage de composants
- Unité de confinement de faute (FCU)
  - une faute unique influence un seul composant
  - Dans un ensemble de FCU, les défaillances doivent être indépendantes
- Le confinement de faute correspond aux efforts de conception pour assurer que les conséquences immédiates d'une faute affecte un seul FCU
- Les modèles de fiabilité courants utilisent l'hypothèse d'indépendance

102

Classification des Fautes



103

Espace temps d'erreurs

- Les fautes physiques sont transitoires ou permanentes, les fautes dues au design sont toujours permanentes
- Une faute transitoire (transient) apparait pour un court intervalle de temps
  - Aucune action explicite n'est nécessaire pour réparer
  - Peut provoquer une erreur mais laisse le matériel intact
  - Une faute externe est transitoire (transitory)
    - Ex : impact d'une particule cosmique
    - Taux d'arrivée généralement constant
  - Une faute interne est intermittente
    - Ex : défaut mécanique non encore permanent (oxydation)
    - Fréquence augmente avec le temps (indicateur pour maintenance préventive)
- Une faute permanente nécessite une action de réparation

104

Erreurs

- Conséquence d'une faute : état incorrect dans le composant
  - Donnée fausse : mémoire, registres, bascules
- Etat d'erreur
  - Activée : un calcul accède à l'erreur et la propage
    - Potentiellement loin dans le temps
    - Erreurs logicielles : Bohrbugs et Heisenbugs
  - Détectée : le calcul perçoit une variation par rapport à ce qui est attendu (ex : vérification de parité)
    - Latence de détection d'erreur (temps de détection)
    - Couverture de détection d'erreur (probabilité de détection)
    - Le test est essentiel pour détecter les erreurs de design
  - Anéantie : écrasée avant d'être détectée ou activée
  - Sinon une erreur est latente (corruption silencieuse de donnée)

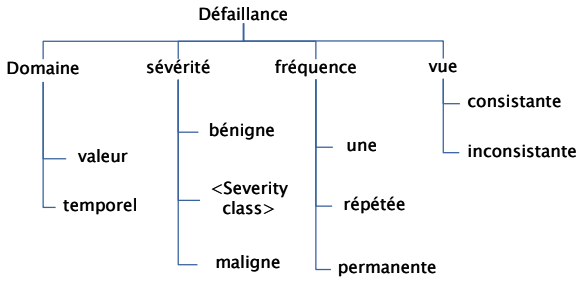
105

Défaillances

- Evénement qui marque une déviation par rapport au comportement attendu (service)
- Dans un système réparti : génération d'un message inattendu
- Message perçu par un utilisateur du service du composant

106

Classification des Défaillances



107

Défaillances

- Domaine temporel
  - Défaillances précoces et tardives (LF)
    - Un composant qui ne présente que des LF ou *défaillances par omission* (OF) (suppression des autres) est appelé *fail-silent component*
    - Si arrêt sur la première OF : *fail-stop component*
- Sévérité
  - Possibilité de définir des classes en fonction de l'impact de la défaillance
  - Une défaillance maligne dépasse la perte du simple service du composant (catastrophe)
- Vue
  - Tous les utilisateurs constate le même comportement défaillant (consistance)
  - Les utilisateurs perçoivent des comportement différents (comportement malicieux – byzantin)
    - difficulté de détection supérieure

108

Propagation

- Si une erreur est activée et propagée hors du composant on parle de *propagation d'erreur*
- Interaction = message => message incorrect
  - Domaine de valeur
    - Détection de la responsabilité des récepteurs
  - Domaine temporel
    - Détection par le système de communication
  - gardien indépendant (safety bag)
- Système périodique : existence d'un g-state
  - Erreur dans le g-State : pollution des calculs du cycle suivant (*érosion d'état*)
    - G-State vide : aucune possibilité de propagation
    - Sinon surveillance de l'intégrité par une tâche de détection d'un composant indépendant