

MI11

Systemes temps reel critique

Jérôme De Miras

Poste : 49 53  
e-Mail : demiras@hds.utc.fr

MI11

utc

1

1

---

---

---

---

---

---

---

---

Bibliographie

« Real-Time Systems : design principles for distributed embedded applications

Hermann Kopetz

Kluwer Academic Publishers

MI11

utc

mars 21

2

2

---

---

---

---

---

---

---

---

Panorama general

MI11

utc

3

3

---

---

---

---

---

---

---

---

Définitions

MI11  
utc

4

4

---

---

---

---

---

---

---

Définitions de base

❑ Un système informatique temps réel doit être évalué non seulement sur la validité des résultats logiques qu'il produit mais aussi par rapport au moment où il est capable de les délivrer

❑ Un système temps réel doit répondre aux stimuli de son environnement dans un temps donné : l'échéance

❑ En fonction des contraintes sur les échéances, un système informatique temps réel peut être dur (hard) ou mou (soft)

MI11  
utc mars 21

5

5

---

---

---

---

---

---

---

Clusters

❑ Un système informatique temps réel n'est qu'une partie d'un système plus large qui dépend du temps et qui se compose de clusters.

Opérateur  
(cluster)

Système informatique temps réel  
(cluster)

Objet Contrôlé  
(cluster)

Interface homme-machine

Interface d'instrumentation

MI11  
utc mars 21

6

6

---

---

---

---

---

---

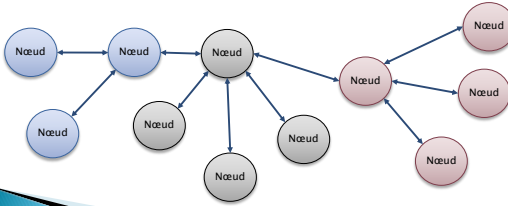
---

2

## Systèmes distribués

- Un système temps réel peut être distribué :

- Nœuds
- Lien de communication temps réel



7

---

---

---

---

---

---

---

---

## Exigences

8

---

---

---

---

---

---

---

---

## Exigences

- A : Exigences fonctionnelles : les fonctions que doit remplir un système informatique temps réel
- B : Exigences temporelles : les contraintes de temps à respecter
- C : Exigences de fiabilité (dependability)

9

---

---

---

---

---

---

---

---

A

Exigences fonctionnelles

❑

Les fonctions que doit remplir un système informatique temps réel peuvent se grouper en trois classes :

❑

collecte de données

❑

pilotage de l'environnement

❑

interactions homme machine

MI11

utc

mars 21

10

---

---

---

---

---

---

---

10

A

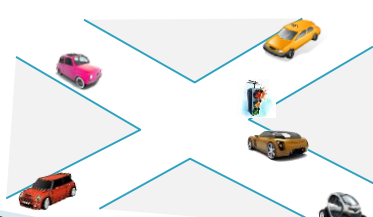
Collecte de données

❑

Un système change d'état en fonction du temps

❑

Si on arrête le temps, l'état peut être décrit par un certain nombre de valeurs  
(position, vitesse, couleur du feu, ...)



MI11

utc

mars 21

11

---

---

---

---

---

---

---

11

A

Entités TR  
Collecte de données

❑

Seul un sous ensemble des variables sont pertinentes pour l'application :  
**Entités Temps Réelles**

❑

Chaque entité TR appartient à une sphère de contrôle où elle peut être modifiée

❑

En dehors de cette sphère, entité TR uniquement observable

MI11

utc

mars 21

12

---

---

---

---

---

---

---

12

A

Images TR

Collecte de données

❑ Observer et collecter ces entités TR est l'exigence N°1 : **Images Temps Réelles**

❑ Validité temporelle limitée des images TR

❑ L'ensemble de images TR valides est appelé Base de données Temps Réelle

- Mise à jour à chaque évolution des entités TR
  - ❑ Observation « Time Triggered »
  - ❑ Observation « Event Triggered »

MI11

utc

mars 21

13

13

---

---

---

---

---

---

---

---

A

Données agréées

Collecte de données

Données

Traitements

BDTR

❑ Conditionnement

❑ Système de détection de faute

➢ Image TR correcte = données agréée

➢ Gestion d'alarmes (importance du temps pour aider l'opérateur à trouver l'origine du défaut)

MI11

utc

mars 21

14

14

---

---

---

---

---

---

---

---

A

Relation au temps

Collecte de données

❑ Toute propriété d'une entité TR qui reste valide pendant un intervalle est un **attribut d'état**

❑ Un changement d'état est un **événement**

❑ Une observation est un événement qui permet l'acquisition de l'état d'une entité TR à un instant donné

❑ L'horloge découpe la ligne de temps en portion égales appelé **granules de temps de l'horloge**

❑ Les granules de temps sont séparés par des événements : les **ticks d'horloge**

❑ Un trigger est un événement qui déclenche une action

MI11

utc

mars 21

15

15

---

---

---

---

---

---

---

---

5

A

Contrôle numérique

❑

Calculer le point de fonctionnement pour un actionneur en fonction de la BDTR

❑

Tâche répétitive et régulière

⇒ problème de l'automatique

⇒ exigences temporelles

MI11

utc

marc 21

16

16

---

---

---

---

---

---

---

A

Interactions homme-machine

❑

Un système temps réel doit informer l'opérateur de l'état du système contrôlé

⇒ Système critique majeur

❑

Une bonne partie des accidents sur des applications critiques ont eu lieu à cause de l'IHM

MI11

utc

marc 21

17

17

---

---

---

---

---

---

---

B

Exigences temporelles

❑

Elle proviennent en grande partie des besoins de performances des boucles de régulation

❑

Exemple : échangeur de chaleur

Point de fonctionnement

Ordinateur de contrôle

Capteur température

Vanne

Capteur de débit

Echangeur de chaleur

MI11

utc

marc 21

18

18

---

---

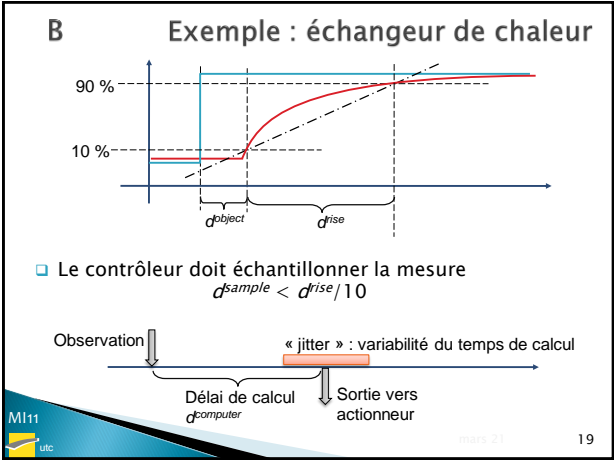
---

---

---

---

---



19

---

---

---

---

---

---

---

---

**B Résumé exigences temporelles**

Variables	Sphère de contrôle	Relations
$d^{object}$	Délai de l'objet contrôlé	Process physique
$d^{rise}$	Temps de réponse	Process physique
$d^{sample}$	Période d'échantillonnage	$d^{sample} \ll d^{rise}$
$d^{computer}$	Temps de réponse calculateur	$d^{computer} < d^{sample}$
$\Delta d^{computer}$	Gigue sur le calcul	$\Delta d^{computer} \ll d^{computer}$
$d^{headtime}$	Temps mort	$d^{computer} + d^{object}$

❑ Critères :

- ❑ On suppose la gigue minimal car en contrôle on sait compenser les délais fixes
- ❑ La gigue apporte du bruit sur les mesures
- ❑ La détection d'une erreur doit se faire dans un temps compatible avec la période d'échantillonnage (minimiser la latence de détection qui dépend de la gigue)

⇒ minimiser les giges (jitter)

MI11  
utc  
mars 21  
20

20

---

---

---

---

---

---

---

---

**C Exigences de fiabilité**

❑ Exigences de fiabilité (dependability)

- ❑ fiabilité (reliability)
- ❑ sureté (safety)
- ❑ maintenabilité (maintainability)
- ❑ disponibilité (availability)
- ❑ Sécurité (security)

MI11  
utc  
mars 21  
21

21

---

---

---

---

---

---

---

---

C

Fiabilité (reliability)

❑  $R(t)$  : probabilité que le système fournisse le service au temps  $t$  sachant son état au temps  $t_0$

$$R(t) = \exp(-\lambda(t - t_0))$$

$\lambda$  : taux de pannes  
1 /  $\lambda$  : MTTF (mean time to failure)

MI11

UTC

marc 21

22

22

---

---

---

---

---

---

---

---

C

Sûreté (safety)

❑ Modes critiques des pannes

❑ Une panne critique est dite maligne

❑ Un système « safety critical » doit avoir un taux de panne extrêmement bas ( $10^{-9}$  panne/h)

❑ Certification du système :

❑ Sous-système critique protégés par des interfaces stables qui empêchent la propagation d'une erreur

❑ Tous les scénarios sont gérables sans argument probabiliste  $\Rightarrow$  design des ressources

❑ L'architecture est compatible avec un processus de certification incrémental

MI11

UTC

marc 21

23

23

---

---

---

---

---

---

---

---

C

Sûreté (2)

❑ Propriétés requises :

❑ Un modèle de sûreté peut être produit  
ou  
tous les paramètres non déduits analytiquement  
sont mesurables en un temps raisonnable

❑ Le modèle de fiabilité n'inclut pas de transition d'état correspondant à un défaut de conception  
 $\Rightarrow$  à prouver analytiquement

❑ Choisir un design qui minimise le modèle

MI11

UTC

marc 21

24

24

---

---

---

---

---

---

---

---



C

Maintenabilité (maintainability)

□ Temps moyen de réparation après une panne

Probabilité  $M(d)$

$d$  : intervalle de temps pour la réparation

MTTR : mean time to repair

Conflit entre fiabilité et maintenabilité

↙

Systèmes monolithiques

↘

SRU : smallest replaceable unit

Pb : connectique

MI11

utc

mars 21

25

---

---

---

---

---

---

---

---

25

C

Sécurité (security)

□ Un système ne doit pas permettre l'intrusion

□ Difficile à quantifier et à spécifier

□ Intégrer des systèmes de cryptage

MI11

utc

mars 21

26

---

---

---

---

---

---

---

---

26

Classification

des systèmes temps réel

MI11

utc

27

---

---

---

---

---

---

---

---

27

« Hard real time » contre  
« Soft real time »

Caractéristique	Hard real time	Soft real time
Temps de réponse	sans compromis	violation possible
Performance sur pic de charge	prédictible	dégradation
Vitesse de conduite	environnement	calculateur
sûreté	souvent critique	non critique
Taille des fichier de données	petit / moyen	large
Type de redondance	active	points de restauration
Intégrité des données	court terme	long terme
Détection des erreurs	autonome	assistée

28

---

---

---

---

---

---

---

---

« fail safe »  
contre « fail-opérationnel »

- ❑ Fail-safe :
  - ❑ en cas de panne le système peut atteindre un état sûr
  - ❑ caractéristique de l'objet et non du système informatique
  - ❑ la détection d'erreur doit avoir une couverture très large ⇒ la gestion TR améliore la disponibilité
- ❑ Fail-opérationnel :
  - ❑ Fonctionner quoi qu'il arrive

29

---

---

---

---

---

---

---

---

« Réponse garantie » contre  
« meilleur effort possible »

- ❑ Le design exclu toute notion de réponse probable (fautes et pics de charge inclus)  
⇒ Garantie sur la réponse
- ❑ Sinon on fait au mieux  
⇒ Validation du design par des tests (économiquement plus attrayant)

30

---

---

---

---

---

---

---

---

« ressources adéquates » contre  
« ressources inadéquates »

□ Un système TR critique a besoin d'être dimensionné pour le pire cas

⇒ Économiquement peu enviable

MI11  
UTC

marc 21

31

31

---

---

---

---

---

---

---

« Event triggered » contre  
« time triggered »

□ Deux approches possibles en fonction du mode de déclenchement des communication et de l'activité de calcul

□ Event triggered : le mécanisme de l'interruption permet de répondre à l'événement  
⇒ ordonnancement dynamique

□ time triggered : seule la progression du temps fait évoluer le système  
⇒ disposer d'un temps global (granularité ?)

MI11  
UTC

marc 21

32

32

---

---

---

---

---

---

---

Systèmes temps réel embarqués

□ Éléments pertinents pour le design d'un système

□ Production de masse

□ Structure statique

□ Interface homme machine

□ Support de stockage du logiciel

□ Stratégie de maintenance

□ Capacité de communication

MI11  
UTC

marc 21

33

33

---

---

---

---

---

---

---

Trois exemples

Régulateur de débit

Moteur thermique

laminoir

MI11

utc

marc 21

34

34

---

---

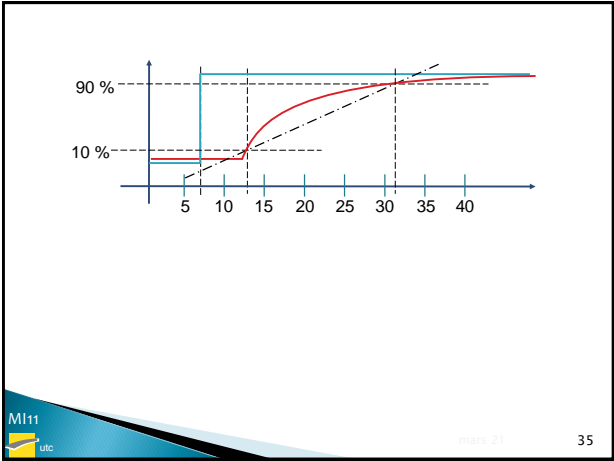
---

---

---

---

---



35

---

---

---

---

---

---

---

Quelle architecture

Pour les systèmes temps réel ?

MI11

utc

36

36

---

---

---

---

---

---

---

## La fonction et la forme

- ❑ Un principe architectural bien établie précise que la fonction d'un objet doit déterminer sa forme [Vitruvius 1960]
- ❑ Dans un système distribué on peut appliquer ce principe à la fonction logique et à son support matériel :  
⇒ **une fonction = un nœud**
- ❑ On peut utiliser une abstraction du nœud physique qui capture « l'essence » de ses fonctionnalités logiques et de ses propriétés temporelles

MI11  
UTC

37

37

---

---

---

---

---

---

---

## Fautes et nœuds

- ❑ S'il n'y a pas équivalence fonction - support, les modes de défaut sont difficiles à interpréter
- ❑ Sous forme d'abstraction, le nœud est parfaitement adapté à la prise en compte des fautes  
$$1 \text{ nœud} = 1 \text{ défaut}$$
- ❑ Très utile pour la validation de grands systèmes

MI11  
UTC

38

38

---

---

---

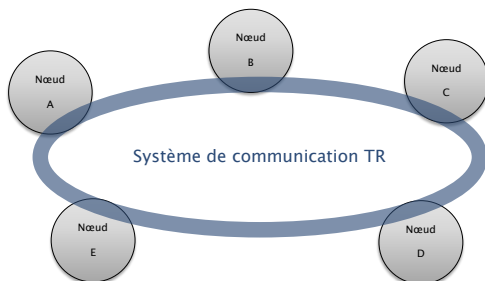
---

---

---

---

## Structure matériel d'un cluster de calcul



- ❑ Un nœud = 2 sous systèmes
  - ❑ Calculateur hôte
  - ❑ Contrôleur local de communication

MI11  
UTC

39

39

---

---

---

---

---

---

---

## Sémantique des messages

- ❑ Deux types à distinguer :
  - ❑ Message relatant un événement
  - ❑ Message relatant une valeur d'état
- ❑ Événement : ne peut être perdu (désynchronisation)
  - ⇒ mise en attente (queue) en respectant un ordre temporel (≠FIFO)
- ❑ Information d'état : peut être perdue
  - ❑ généralement besoin de la dernière
  - ❑ Plus d'un récepteur peut être intéressé par le message
  - ❑ Fréquence plus importante que les événements



40

40

---

---

---

---

---

---

---

## Stratégie de contrôle

- ❑ Sphère de contrôle
  - ❑ Soit dans le calculateur hôte
    - ❑ Situation courante (SEND depuis l'application)
    - ❑ Le message est envoyé à la demande
    - ❑ Interruption à la réception
      - ⇒ des signaux de contrôle traversent le CNI
  - ❑ Soit dans le système de communication
    - ❑ Contrôle autonome (décision interne au RTCS)
    - ❑ Time triggered : table temporelle d'envoi
    - ❑ Ordonnancement transmission / réception
      - ⇒ aucun signal de contrôle ne traverse le CNI (sorte de mémoire partagée)



41

41

---

---

---

---

---

---

---

## « Event message »

- ❑ Combinaison
  - ❑ Sémantique d'événement
  - +
  - ❑ Contrôle externe
- ❑ Mise en file d'attente
- ❑ Nécessite une synchronisation « un pour un »  
sinon risque de débordement



42

42

---

---

---

---

---

---

---

### « State message »

- ❑ Combinaison
  - ❑ Sémantique de valeur d'état
  - +
  - ❑ Contrôle automatique
- ❑ Très proche d'une variable globale mais :
  - ❑ Le système de communication assure l'atomicité d'une lecture/écriture
  - ❑ Un seul processus d'écriture (émetteur)
- ❑ Pas de synchronisation « un pour un », un récepteur peut lire plusieurs informations de suite



43

43

---

---

---

---

---

---

---

### La fonction et la forme

- ❑ Un principe architectural bien établie précise que la fonction d'un objet doit déterminer sa forme [Vitruvius 1960]
- ❑ Dans un système distribué on peut appliquer ce principe à la fonction logique et à son support matériel :  
⇒ **une fonction = un nœud**
- ❑ On peut utiliser une abstraction du nœud physique qui capture « l'essence » de ses fonctionnalités logiques et de ses propriétés temporelles



44

44

---

---

---

---

---

---

---

### Fautes et nœuds

- ❑ S'il n'y a pas équivalence fonction – support, les modes de défaut sont difficiles à interpréter
- ❑ Sous forme d'abstraction, le nœud est parfaitement adapté à la prise en compte des fautes  
$$1 \text{ nœud} = 1 \text{ défaut}$$
- ❑ Très utile pour la validation de grands systèmes



45

45

---

---

---

---

---

---

---

### Structure matériel d'un cluster de calcul

- Un nœud = 2 sous systèmes
  - Calculateur hôte
  - Contrôleur local de communication

MI11  
UTC

46

46

---

---

---

---

---

---

---

---

### Sémantique des messages

- Deux types à distinguer :
  - Message relatant un événement
  - Message relatant une valeur d'état
- Événement : ne peut être perdu (désynchronisation)
  - ⇒ mise en attente (queue) en respectant un ordre temporel (≠FIFO)
- Information d'état : peut être perdue
  - généralement besoin de la dernière
  - Plus d'un récepteur peut être intéressé par le message
  - Fréquence plus importante que les événements

MI11  
UTC

47

47

---

---

---

---

---

---

---

---

### Stratégie de contrôle

- Sphère de contrôle
  - Soit dans le calculateur hôte
    - Situation courante (SEND depuis l'application)
    - Le message est envoyé à la demande
    - Interruption à la réception
      - ⇒ des signaux de contrôle traversent le CNI
  - Soit dans le système de communication
    - Contrôle autonome (décision interne au RTCS)
    - Time triggered : table temporelle d'envoi
    - Ordonnancement transmission / réception
      - ⇒ aucun signal de contrôle ne traverse le CNI (sorte de mémoire partagée)

MI11  
UTC

48

48

---

---

---

---

---

---

---

---



« Event message »

❑ Combinaison

❑ Sémantique d'événement


+

❑ Contrôle externe

❑ Mise en file d'attente

❑ Nécessite une synchronisation « un pour un » sinon risque de débordement

MI11

 utc

49

49

---

---

---

---

---

---

---

« State message »

❑ Combinaison

❑ Sémantique de valeur d'état

+

❑ Contrôle automatique


❑ Très proche d'une variable globale mais :

❑ Le système de communication assure l'atomicité d'une lecture/écriture

❑ Un seul processus d'écriture (émetteur)

❑ Pas de synchronisation « un pour un », un récepteur peut lire plusieurs informations de suite

MI11

 utc

50

50

---

---

---

---

---

---

---

Propriétés importantes

Composabilité

Evolutivité

Fiabilité

Réalisation physique

MI11

 utc

51

51

---

---

---

---

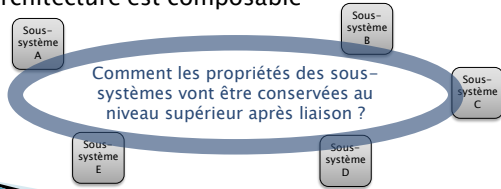
---

---

---

## Composition

- ❑ Un grand système se construit par intégration de sous-systèmes bien spécifiés et testés
- ❑ Leur connexion ne doit pas détruire ces propriétés
- ❑ Cette approche constructive est possible si l'architecture est composable




---

---

---

---

---

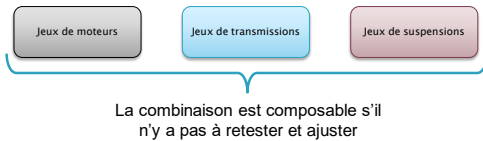
---

---

---

## Composition : définition

- ❑ Une architecture est composable par rapport à une propriété si l'intégration du sous-système n'invalide pas cette propriété si elle est établie pour le sous-système
- ❑ Héritage de la propriété




---

---

---

---

---

---

---

---

## Composition : event triggered systems

- ❑ Plusieurs nœuds communiquent vers un seul
  - ❑ Si plusieurs canaux  $\Rightarrow$  surcharge
  - ❑ Si un seul canal  $\Rightarrow$  conflit
- ❑ Résolu de différentes façons
  - ❑ Accès aléatoire (Ethernet)
  - ❑ Ordre prédéfini d'accès (token protocol)
  - ❑ Priorité (Control Area Network)
- ❑ Pb : le comportement temporel n'est pas défini par le système de communication, mais par les nœuds.
- ❑ De ce point de vue ce n'est pas composable

---

---

---

---

---

---

---

---

## Composition : Time triggered systems

- ❑ Gestion du temps indépendante des nœuds
  - ❑ Messages envoyés en fonction de table dans le CNI
  - ❑ Le contrôleur hôte n'a pas la capacité d'influencer ce comportement temporel
  - ❑ Le CNI n'influence pas le contrôleur hôte
    - ⇒ pare feu temporel, pas de propagation d'erreur
- ❑ Le comportement temporel de chaque CNI est parfaitement défini et testable indépendamment
  - ⇒ la composition ne change pas le comportement temporel
- ❑ Architecture TT composable du point de vue des échéances de communication



55

55

---

---

---

---

---

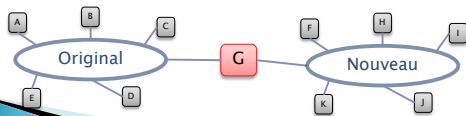
---

---

---

## Evolution

- ❑ Un système de grand taille doit pouvoir évoluer par l'ajout de nouvelles fonctions
- ❑ Capacité d'extension : ne pas avoir de goulot d'étranglement ni sur la capacité de calcul ni sur le système de communication
  - ⇒ système distribué
  - ❑ Des nœuds peuvent être ajoutés dans la limite du système de communication
  - ❑ Si capacité max ⇒ ajout d'un cluster par un passerelle



56

56

---

---

---

---

---

---

---

---

## Complexité

- ❑ Un système large ne peut être construit que si l'effort pour le comprendre reste contrôlable avec l'évolution ✓
- ❑ L'effort doit rester constant pour une fonction particulière
  - ⇒ système distribué avec une bonne maîtrise du système de communication



Pour analyser un cluster :

- ❑ comportement temporel de la passerelle
- ❑ données disponibles

La complexité est contenue dans le cluster



57

57

---

---

---

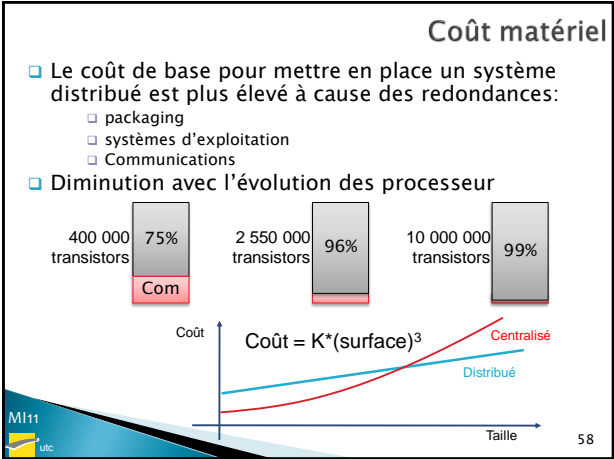
---

---

---

---

---



---

---

---

---

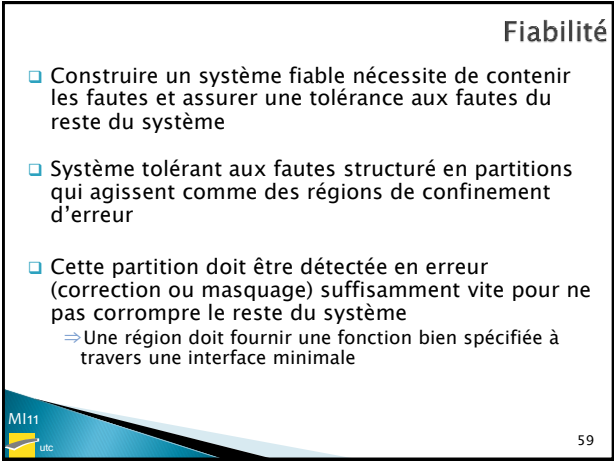
---

---

---

---

58



---

---

---

---

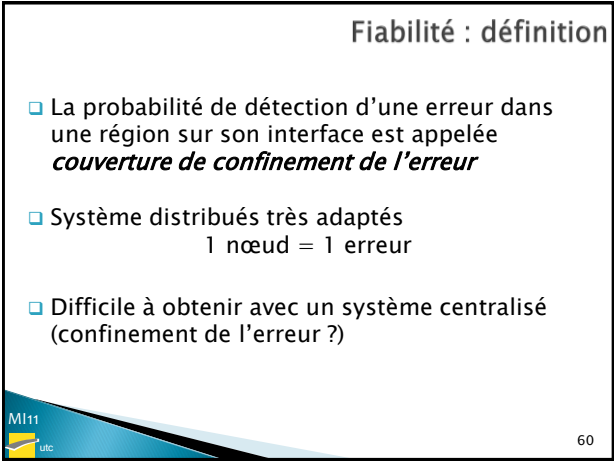
---

---

---

---

59



---

---

---

---

---

---

---

---

60

Fiabilité : réplication

- ❑ 1 nœud = 1 unité de défaut
- ❑ Si possible 1 mode de défaut : *silence sur faute*
- ❑ 💡 répliquer les nœuds
  - ❑ Importance d'assurer le déterminisme de la réplication (comportement identique)
    - ❑ Difficile dans un contexte d'ordonnancement asynchrone

61

---

---

---

---

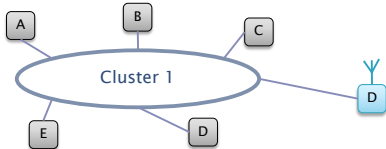
---

---

---

Fiabilité : certification

- ❑ Montrer qu'une erreur dans un système non critique ne peut traverser vers un système critique
- ❑ Faire des preuves des *safety cases* (proposées par le concepteur)



62

---

---

---

---

---

---

---