FROM RESEARCH TO INDUSTRY

cea tech

Méthodes de conception de systèmes temps réels critiques : cadencement par le temps ou des évènements

Mathieu.Jan@cea.fr

www.cea.fr

leti & list

Plan

- Concepts de base
  - Aperçu du fonctionnement d'une approche cadencée par le temps
  - Illustration de la programmation & exemple applicatifs
  - Extension pour adresser d'autres domaines applicatifs (contrôle moteur)
    - Cadencé par des évènements extérieurs temporellement maîtrisés
  - Aperçu d'une extension au distribué (topologie bus)

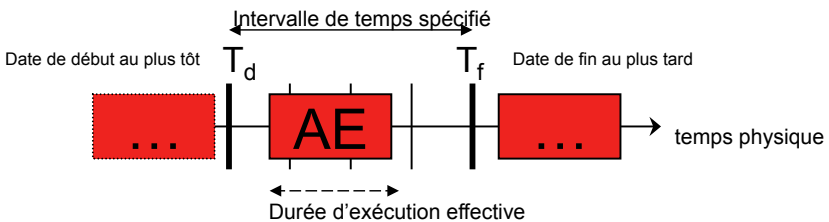
- Travaux sur la base d'une approche cadencée par le temps
  - Modèle de tâche
    - Application à la description du comportement temporel des tâches
    - Application à la description du comportement réseau
  - Mécanismes de migrations de tâches
  - Génération automatique depuis Scade : équivalence fonctionnelle
  - Ordonnancement temps réel multiprocesseur avec gestion de l'énergie

Méthode de conception

- Modèle de programmation : ΨC
  - Expression du parallélisme (définition des agents)
  - Expression de contraintes temporelles
  - Expression des communications
- Modèle d'exécution (multitâche communiquant)
  - Cadencé par le temps réel, multi-échelle
  - Cadencements déduits d'après les contraintes de temps exprimées et les communications
- Outils support permettant …
  - … la génération de code et l'implantation sur cible
  - … l'aide à l'analyse et la validation (simulation POSIX)
- Noyau orienté sûreté de fonctionnement et performance
  - Simple cœur (e.g. 68K, IA32, ARM), multicoeur hétérogène (e.g. S12XE), multicoeur SMP (e.g. IA32), architecture distribuée

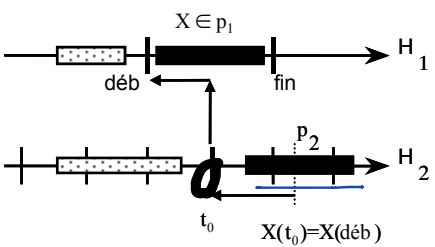
Modèle d'exécution

- Le système est cadencé par le temps réel (Time-Triggered)
  - tous les traitements prennent place entre deux instants du système
    - les exécutions sont asynchrones mais sont synchronisées (synchronisme) au début et à la fin des traitements : isochronisme réel indépendant du matériel
    - Cela définit une action élémentaire (AE)
  - tous les transferts de données entre « tâches » ont lieu à la transition entre deux traitements
  - Multi-échelle de temps



Modèle d'exécution

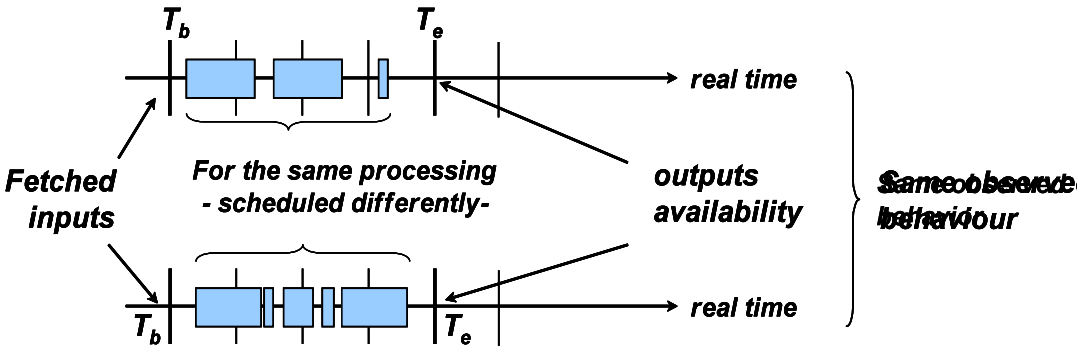
- Cohérence temporelle des échanges
  - encapsulation des données et des traitements (donnée protégée et un seul producteur)
  - les valeurs des données sur lesquelles travaille une AE sont figées à un instant temporel donné (date de début de l'AE)
  - Consommation et production de nouvelles données uniquement aux TSPs



- principe d'observabilité strict
- mise à jour implicite et automatique aux rythmes définis
- dimensionnement automatique et protection des tampons, optimaux et sûrs

Determinism property

- Data values are independent
  - From the effective execution duration
  - From the scheduling policies (including job preemptions)



- Determinism property
  - Dynamic behavior of the system predictable and unvarying (reproducibility)
  - Implementations have identical results (except for the sizing)
  - Test scenarios are predictable and their results reproducible

Exemple d'écriture d'un cadencement simple

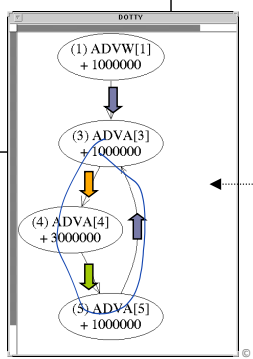
- Un agent avance dans le temps par sauts successifs le long d'une échelle de temps (horloge de base de la tâche) définie par le concepteur pour chaque agent
  - Les sauts peuvent être de longueurs différentes (comme ci-dessous 1-3-1-1-3-1-1-…)


```
clock HBASE= gtc1(0, 1); /* 1 ms */
application demo(inittime= 99) with
HBASE;
agent ag1tst(starttime= 1) with HBASE
{
  body start
  {
    /* proc 1 */ advance(1);
    /* proc 2 */ advance(3);
    /* proc 3 */ advance(1);
  }
}
```

Initialisation TT :

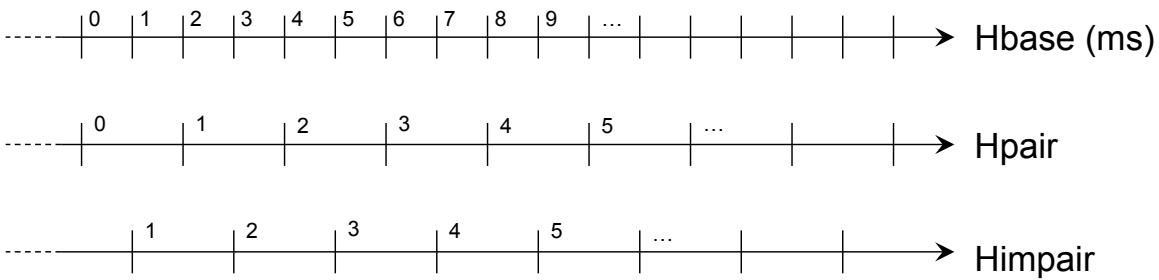
- La valeur initiale du temps TT de l'application est elle aussi prédéfinie par le concepteur
- Chaque tâche démarre à partir d'un instant spécifié de son horloge de base après le démarrage TT de l'application

Chaîne de compilation et de génération de code



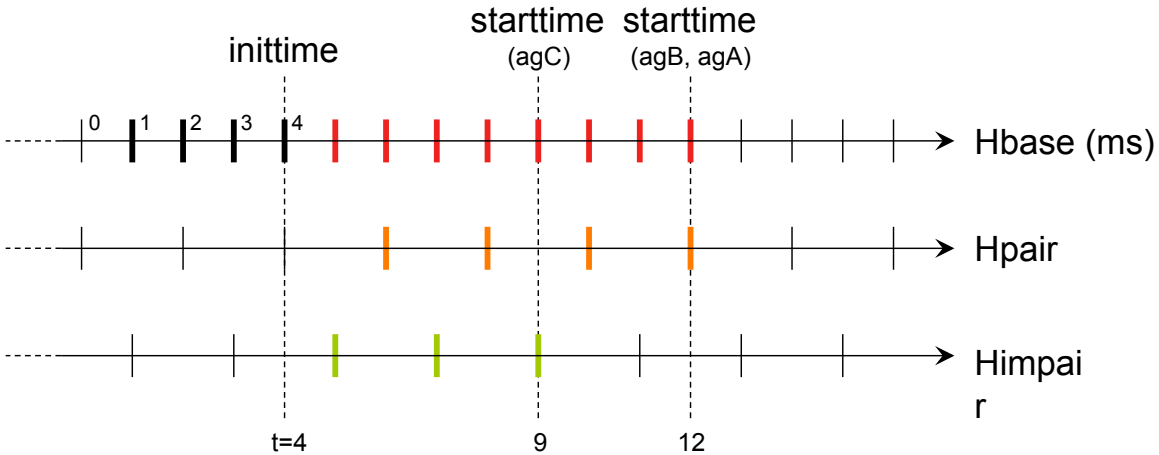
Graphe de contrôle et d'exécution de la tâche utilisé par le noyau

Les horloges : bases de temps



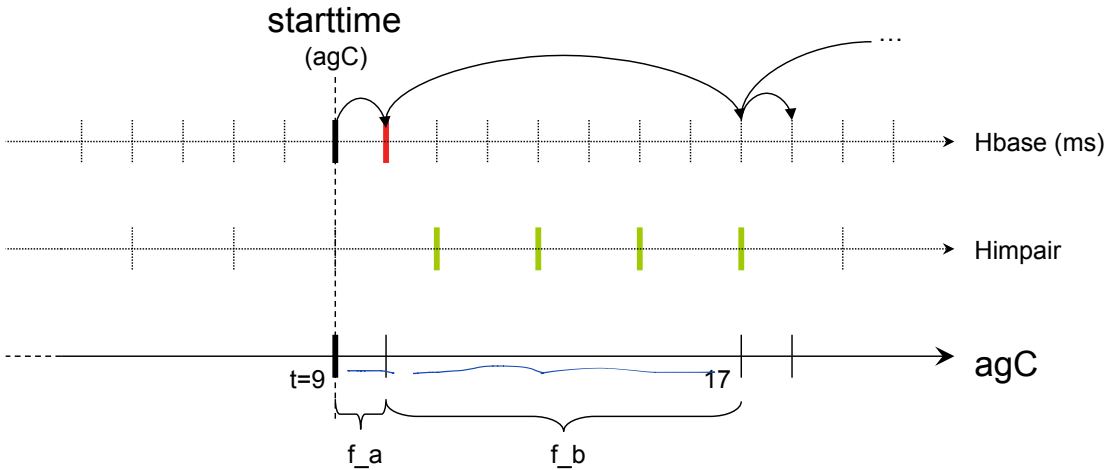

```
clock Hbase = gtc1 (0,1) ;
  ou clock Hbase = gtc2 (0,1,0) ;
  ou clock Hbase = gtc3 (0,1,0,0) ;

clock Hpair = 2*Hbase ;

clock Himpair = 2*Hbase + 1 ;
```

Les horloges : démarrage




```
application monAppli ( inittime = 4 ) with Hbase ;

agent agA ( starttime = 8 ) with Hbase
  { ... /* ensemble de blocs */ ... }
agent agB ( starttime = 4 with Hpair ) with Hbase
  { ... }
agent agC ( starttime = 3 with Himpair ) with Hbase
  { ... }
```

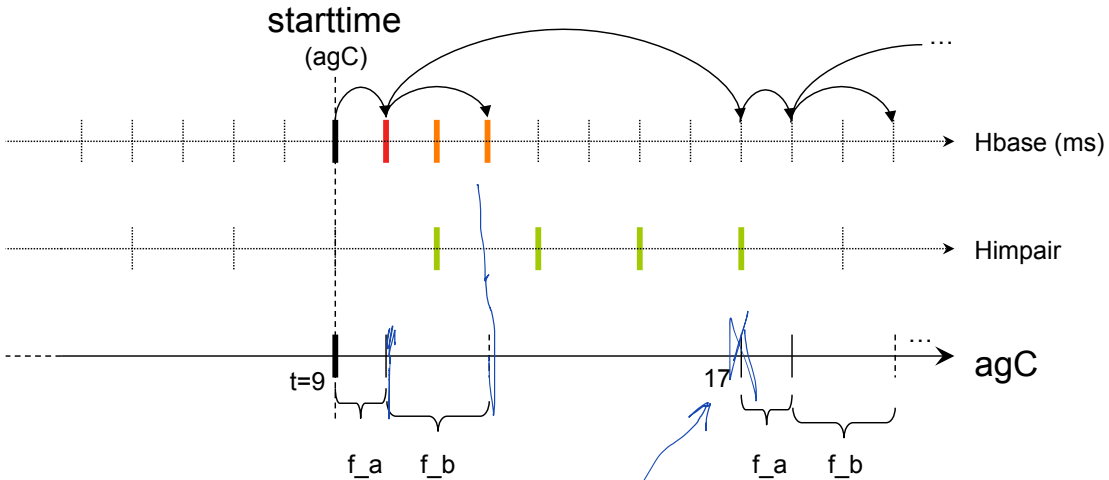
Fenêtres temporelles : advance



(rmq : code de l'agent agC exécuté après son démarrage)

```
body start
{
   /* traitement f_a */
   advance (1) ;
   /* traitement f_b */
   advance (4) with Himpair ;
}
```

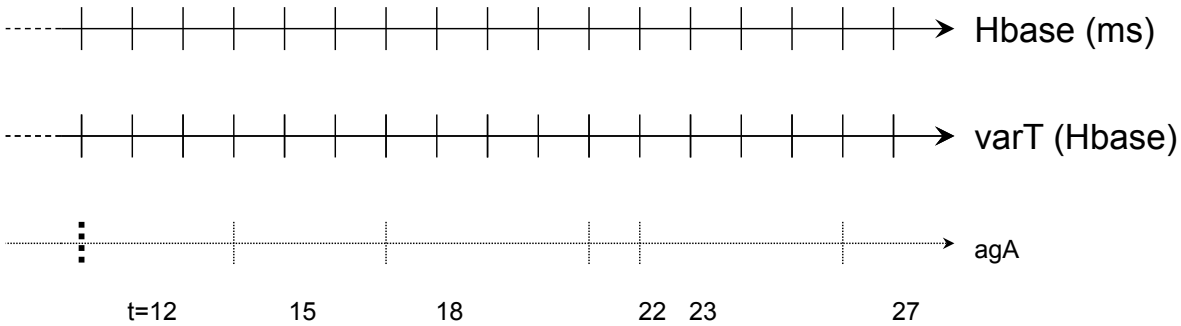
Fenêtres temporelles : before



Code de l'agent agC exécuté après son démarrage :

```
{
   /* traitement f_a */
   advance (1) ;
   /* traitement f_b */
   before (2) ;
   /* traitement f_c */
   advance (4) with Himpair ;
}
```

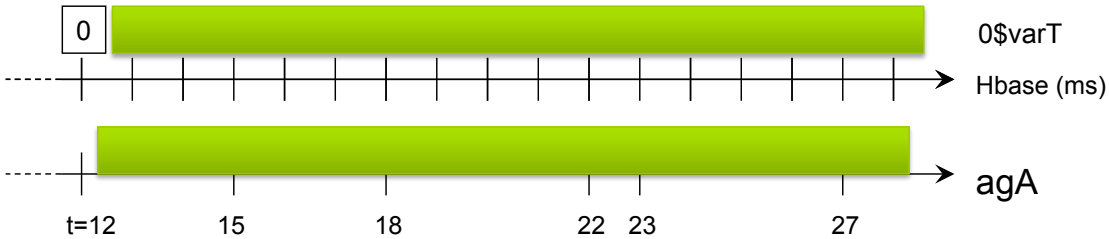
Variables temporelles : définition et droits




```
/* Blocs chez le propriétaire (agA) */
temporal
{
   unsigned long 1$varT=0 ;
}

display
{
   varT : agB ;
}
```

```
/* Bloc chez un consultant (agB) */
consult
{
   agA : 1$varT ;
}
```

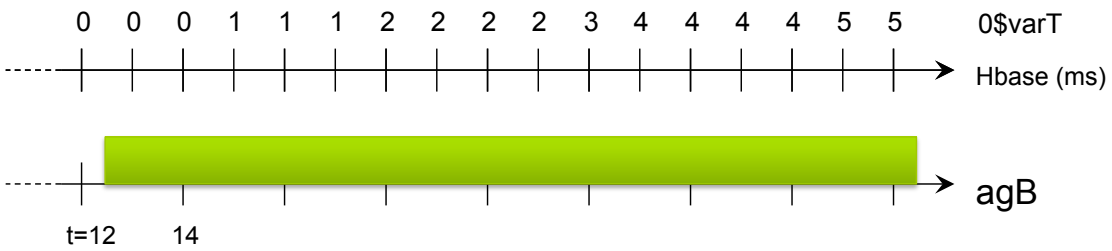
Variables temporelles : accès aux valeurs passées (1/2)




```
/* Code chez le propriétaire (agA) */
{ varT=1; affiche(`0$varT);
  advance(3);
  ++varT; affiche(`0$varT);
  advance(2) with Hpair;
  ++varT; affiche(`0$varT);
  advance(2) with Hpair;
  ++varT; affiche(`0$varT);
  advance(1);
  ++varT; affiche(`0$varT);
  advance(4); /* ... */        }
```

Q : dessinez chronogramme avec 'temporal with Hpair' et le même code ΨC

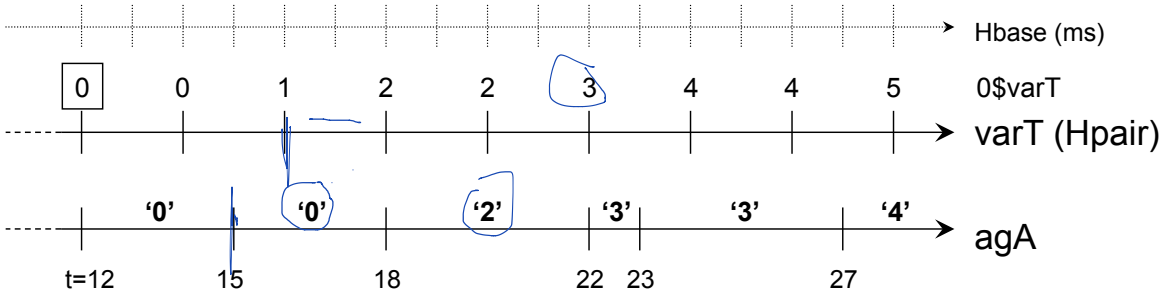
Variables temporelles : accès aux valeurs passées (2/2)




```
/* Code chez le consultant (agB) */
{ affiche(agA`0$varT);
  advance(1) with Hpair ;
}
```

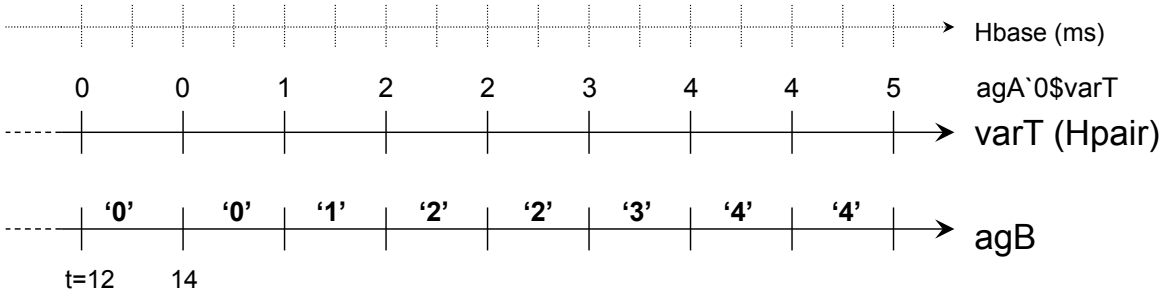
Q : redessinez ce chronogramme lorsque agB consulte les deux dernières valeurs passées de varT

Variables temporelles : réponse




```
/* Code chez le propriétaire (agA)  */
{ varT=1; affiche(`0$varT);
  advance(3);
  ++varT; affiche(`0$varT);
  advance(2) with Hpair;
  ++varT; affiche(`0$varT);
  advance(2) with Hpair;
  ++varT; affiche(`0$varT);
  advance(1);
  ++varT; affiche(`0$varT);
  advance(4); /* ... */          }
```

Variables temporelles : réponse




```
/* Code chez le consultant (agB) */
{ affiche(agA`0$varT);
  advance(1) with Hpair ;
}
```

Construction simple d'une « synchronisation multi-échelle »

emacs: exemple_FRA_2.psy

File Edit Apps Options Buffers Tools C Help

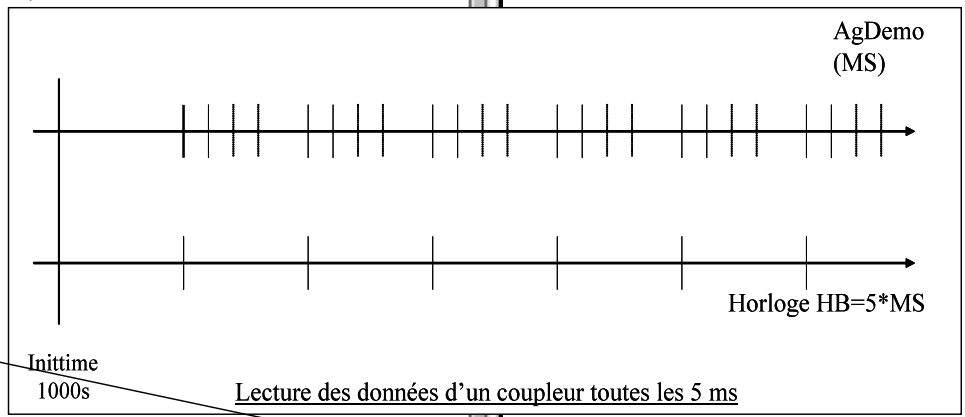
```
clock MS = gtc1( 0, 1 ); /* horloge de période 1 millisecondes */
clock HA =  1*MS;        /* horloge de période 1 millisecondes */
clock HB =  5*HA;        /* horloge de période 5 millisecondes */

/* déclaration de la date de début de l'application */
application demo (inittime=1000*1000) with MS;

#include "simu_coupleur.h"

/* code source de l'agent AgDemo */
agent AgDemo (starttime=1 with HB) with HA
{
  body start
    {
      long indParc=0;
      long maxParc;

      maxParc=lec_donn();

      do [3]  {
        advance(1);
        indParc++;
      } while (indParc < maxParc);

      advance(1) with HB;
    }
}
```

-----XEmacs: exemple_FRA_2.psy (C PenDel Font)----L1--C1--All----------

Fontifying exemple_FRA_2.psy... done.

Déclaration de plusieurs horloges (échelles de temps) :

1ms et 5 ms



Contrainte temporelle d'un tick de l'horloge à 1ms

Resynchronisation au prochain tick de l'horloge à 5ms quelque soit le nombre de tours de boucle

Exemple simple de gestion de la gigue

- Affichage d'un tiret toute les secondes avec une maîtrise de la gigue
  - Par construction le cadencement temporel implémenté garantit une gigue entre deux occurrences d'affichage de [750;1250ms] (si système ordonnançable)

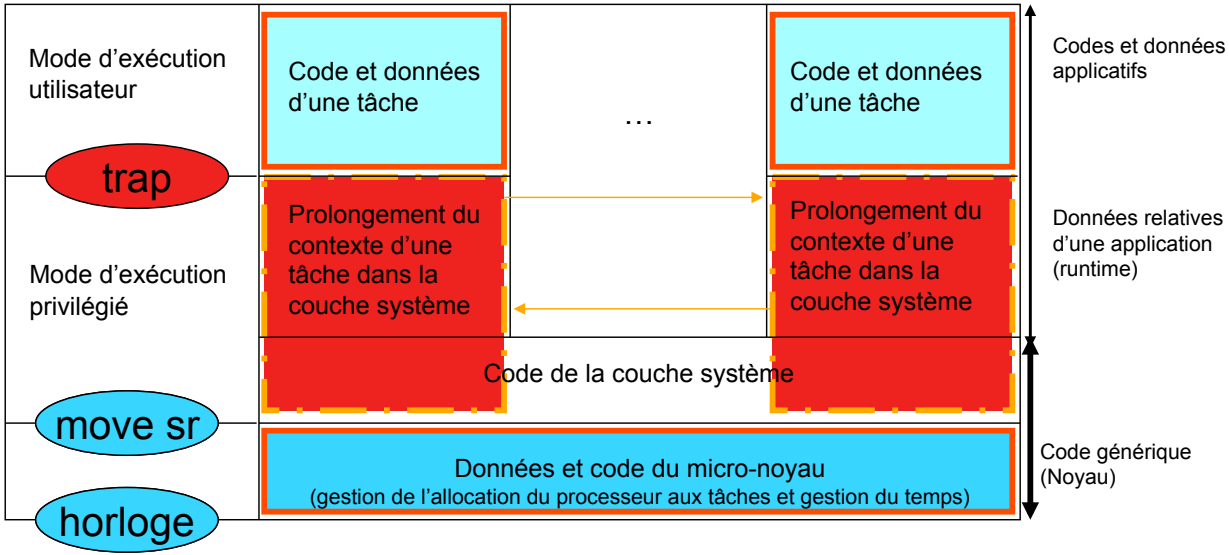
Anomalies, déterminisme et modularité

- Un système contient *toujours* des défauts résiduels

- Principe de déterminisme fort dans OASIS
  - Confiner les anomalies et maîtriser leurs conséquences pour conserver la propriété de déterminisme même en cas d'anomalies
  - il faut
    - détecter à l'exécution toutes les tentatives de violation de la séparation des tâches
    - contrôler le dimensionnement des ressources d'exécution (tampons, quotas CPU, etc.)
    - contrôler l'enchaînement séquentiel des actions, etc.

- Partitionnement spatial et temporel au sein d'OASIS
  - Unité d'exécution et de confinement d'anomalie : l'agent
  - Isolation spatiale (illustré sur le transparent suivant)
    - entre les différents agents
    - entre les différentes parties du noyau
    - entre le noyau et les agents
  - Isolation temporelle
    - Surveillance des budgets (WCET) associés aux actions élémentaires (AE) : ségrégation de la défaillance au sein de la tâche et maintien de disponibilité du système
      Impact limité à l'agent, possibilité de définir une stratégie d'exécution en mode dégradé
    - Surveillance des échéances des AE : détection d'une erreur de dimensionnement

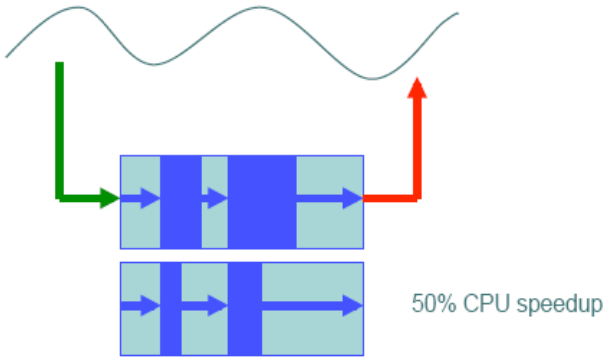
Organisation défensive des programmes exécutables



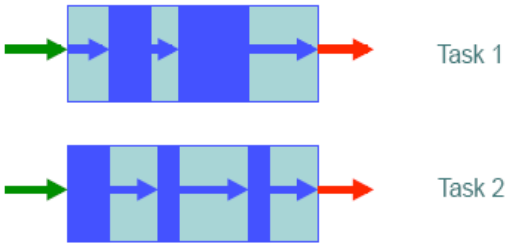
- contextes d'exécution distincts hors de CS et différents internes à CS, contexte μN indépendant
- segmentation particulière des exécutables et des données
  - conception orienté sûreté et exploitation spécifique des mécanismes de protection liés au H/W
- contrôle de conformité de l'exécution de chaque tâche
- contrôle toutes les hypothèses de dimensionnement (quotas, tampons, etc.)
- les préemptions de tâches sont toutes contrôlées par le μNoyau atomique
- CS 100% préemptible

Bénéfices

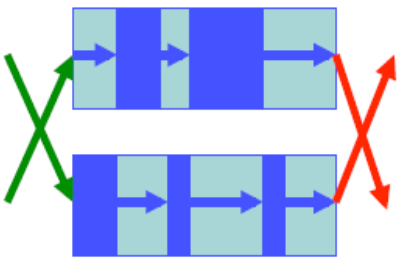
Portabilité



Composabilité

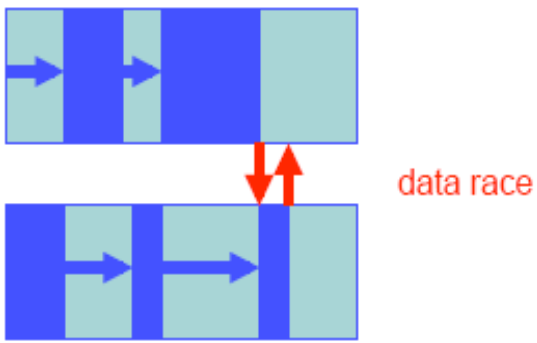


Déterminisme



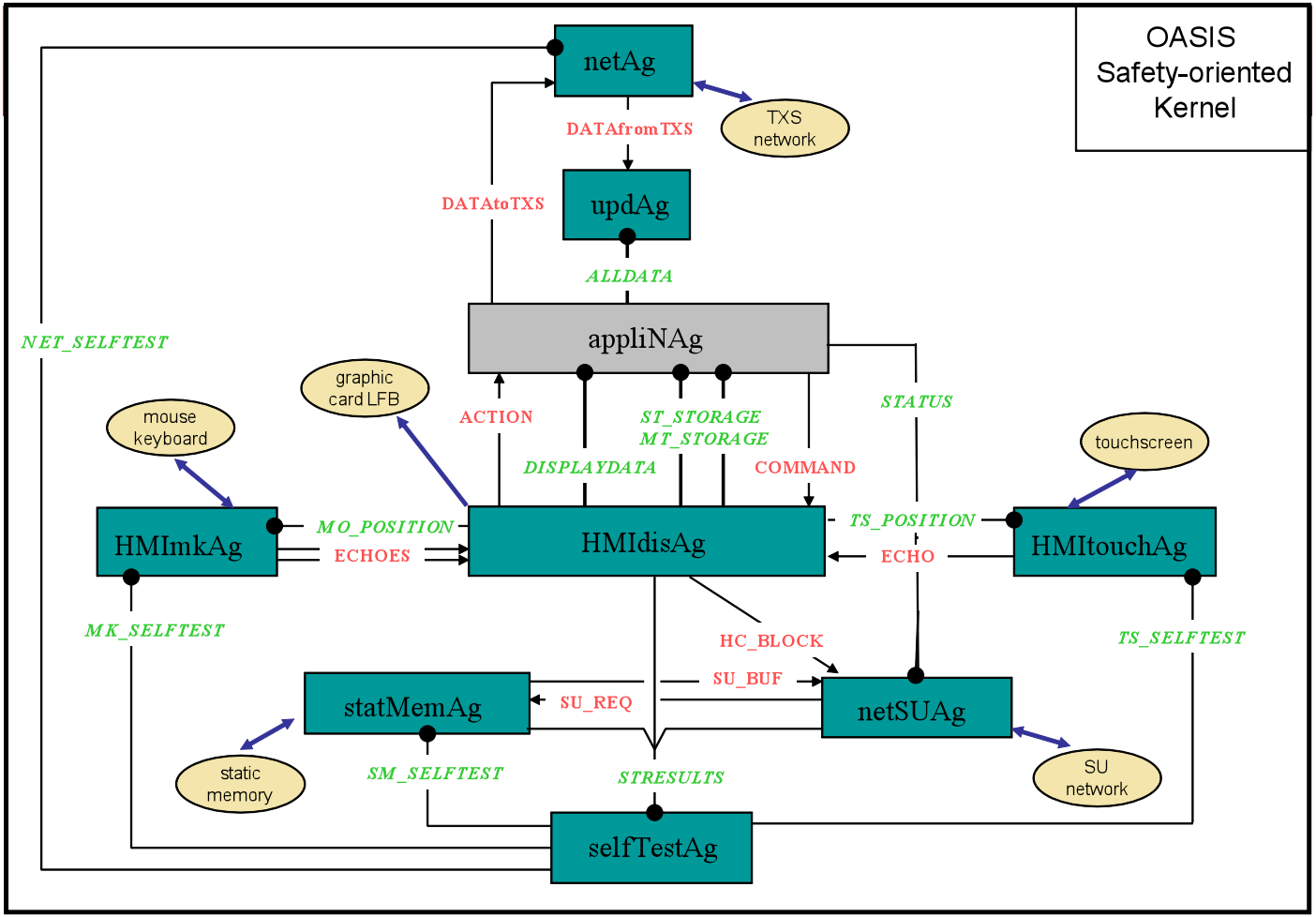
Timing predictability: minimal jitter
Function predictability: no race conditions

Différence avec les pratiques courantes

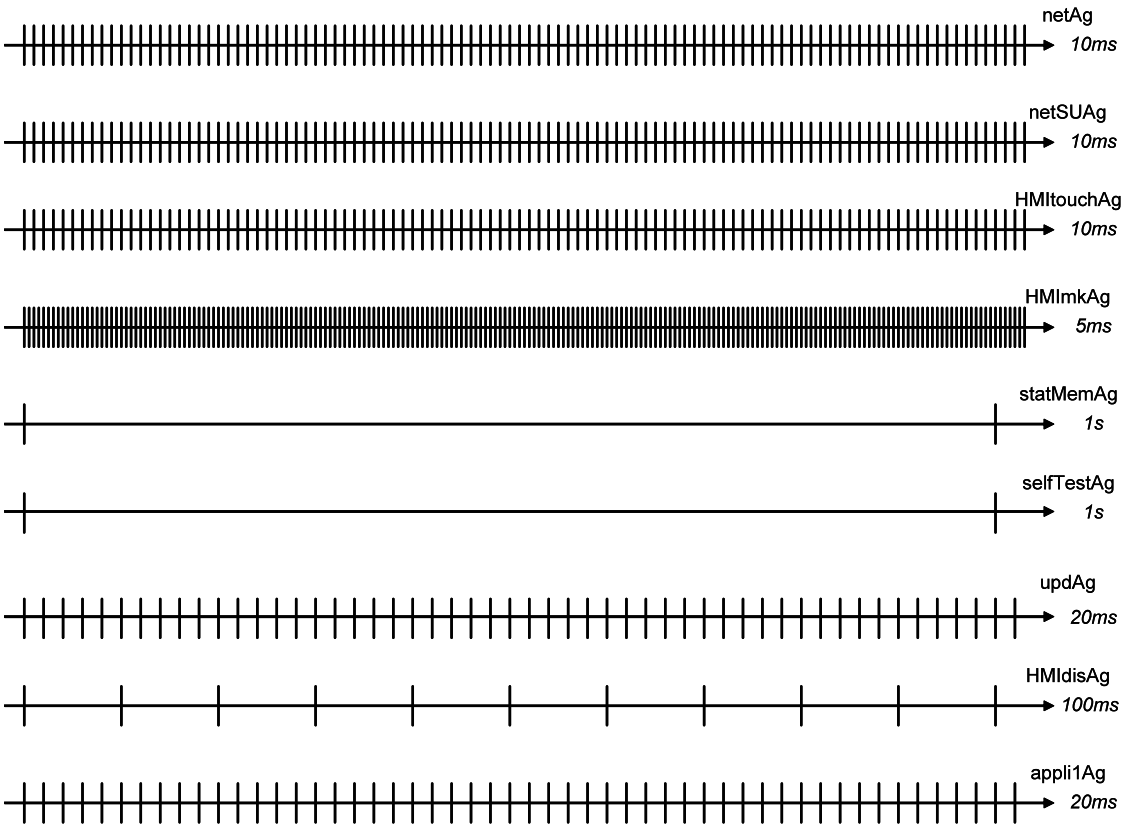


OASIS and the QDS plaform

- Industrial success in the nuclear industry with AREVA NP (2003-2006)
  - The Qualified Display System platform (compliant to requirements of safety classified system of category A, IEC 61513 & 60880)
    - Part of 1E safety-classified I&C system of the EPR (European pressurized reactor)
    - Single processor IA32 architecture
- QDS: display equipment with high dependability level for safety classified application
  - Based on the OASIS system software and off the shelf hardware components
  - Application: post-accident monitoring & control / supervision of the NPP starting phase and monitoring when operational
    - full control of a graphical HMI (10 images per second, 1024x768x16bpp)
    - data storage (24h) and calculations for 20 million measure points
    - network communication management (about 50000 data per second)
    - application downloading and monitoring
    - all hardware component configurations performed & checked
    - self-tests in regular time

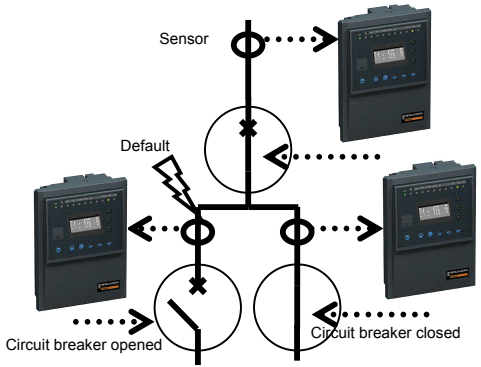


QDS: example of Real-Time behaviors



Medium voltage protection relays

- Safety-function of the software part of protection relays
  - Detects and isolates fault in the electrical network
  - Sepam line of products within Schneider Electric

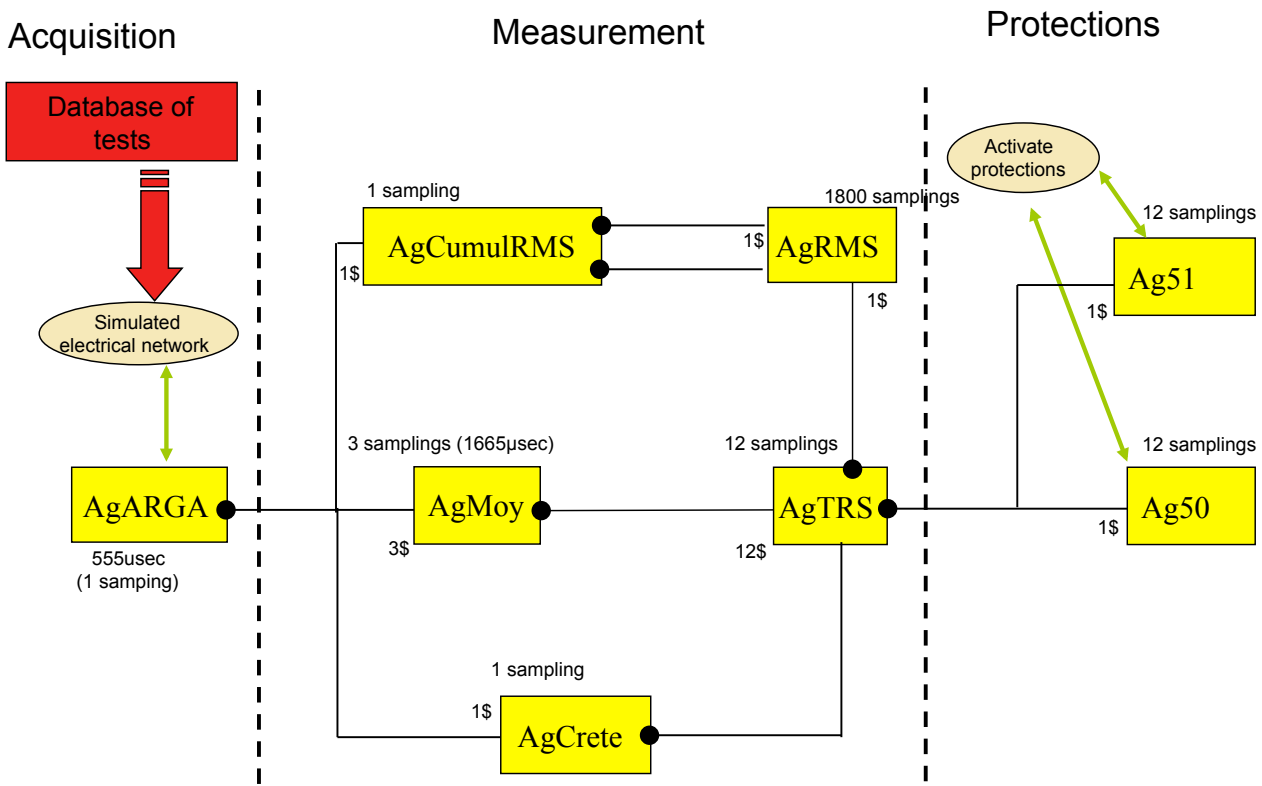


Sensor: transmits information current/ voltage

Relay: digitalizes the information and takes decisions according to user settings

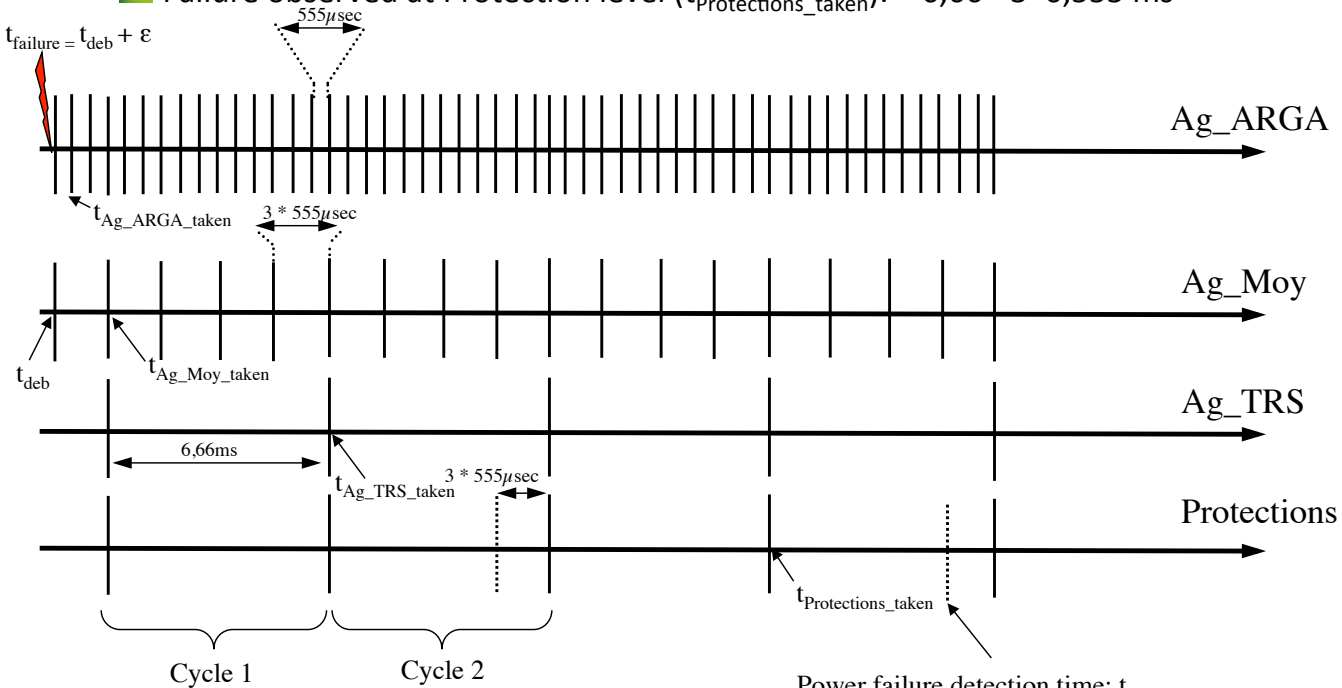
- Requirements for SIL2 or higher certification (IEC 61508)
  - Demonstration of the fulfillment of the detection delay ($t_{detection}$)
    - End-to-end temporal constraint between the detection of power faults and asking the tripping of circuit breakers
      - $t_{detection}$ <= 26,640 ms
  - Current practice: costly phase of tests and high margins are taken on the temporal behaviour of the software part of protection relays

Software architecture of OASISepam: a simplified Sepam 10 protection relay



Mastered temporal behavior within OASISepam

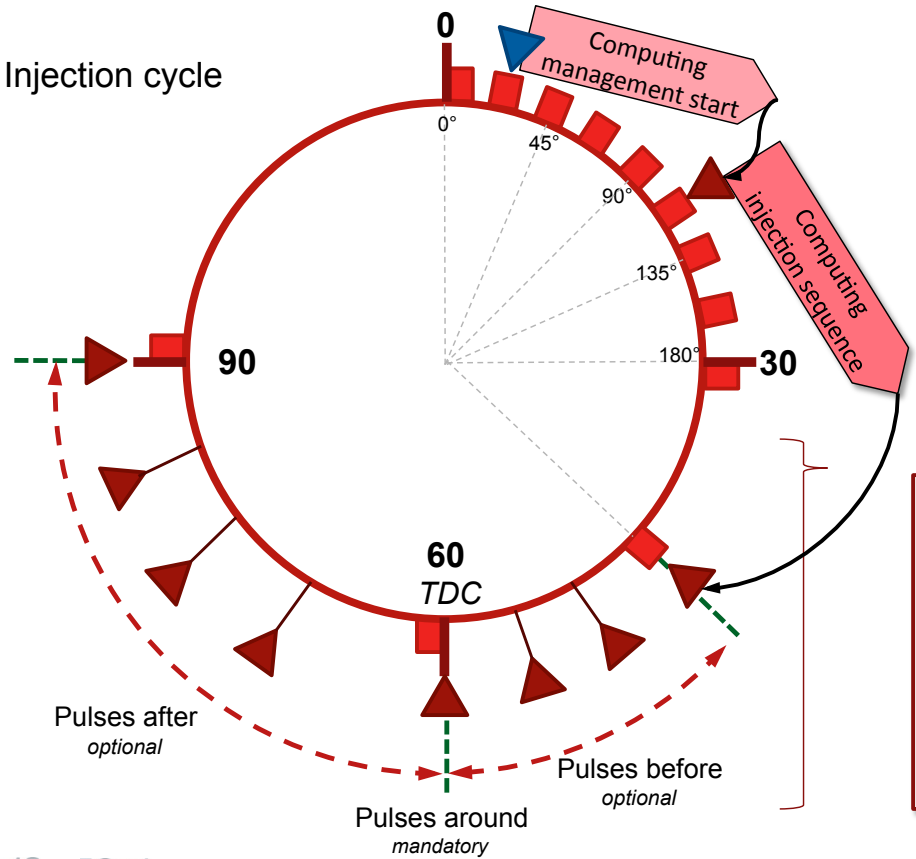
- Worst-case $t_{detection}$ = 26,64 ms
  - Failure observed at AgMoy level (t_{AgMoy_taken}): 3 * 0,555 ms
  - Failure observed at AgTRS level (t_{AgTRS_taken}): + 6,66 * 3 ms
  - Failure observed at Protection level ($t_{Protections\_taken}$): + 6,66 - 3*0,555 ms



Advanced features of PharOS: context

- Domain field: automotive powertrain
  - Move from single core to multi-core architectures in order to be compliant with next « euro » emission norms
  - Upcoming safety standard (ISO 26262) requires an increase in the confidence of the temporal behavior

Powertrain Usecase

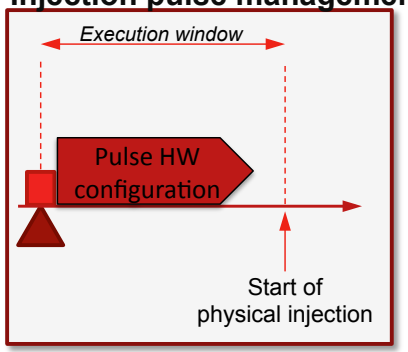


Static angle

Dynamic angle

SW processing

Injection pulse management

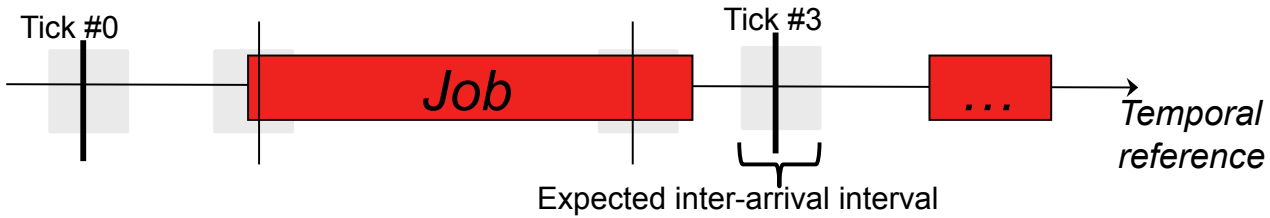


Advanced features of PharOS: problems

- Classical Time-Triggered (TT) paradigm [Giotto, etc.]
  - Specify task activation and deadlines using physical time
  - Abstract the underlying architecture
  - BUT inappropriate for interacting with the external world
    - Inefficiency of the looping approach
    - Express timing constraints in another temporal reference than physical time

Generalized TT paradigm: eXternal-Triggered (xT)

- Address sporadic interactions
  - Specify job release/deadline independently of the exact timing occurrence of events
    - Min. and max. inter-arrival time for: 1) system liveness and 2) schedulability demonstration



- Specificy timing constraints using an application specific temporal reference
  - Given for a set of range of values of an application-level variable
    - For example: [1000 – 3000 rpm] and [3000 – 5000 rpm]

Application-level filtering

- Executed at each occurrence of an xT tick
- To implement a specific protocol (receive/send)
- To give a semantic to xT ticks
  - What is the number of a tick ?
  - Should the tick be filtered?
    - Initialization sequence (identification of the tooth number)
    - Intermediate event for communicating through a peripheral

Extension à des bus de communication

- Continuité et intégration avec le modèle OASIS
  - intègre déjà de manière réaliste les contraintes temporelles
    - délais de communication jamais estimés comme nuls
    - diffusion avant l'échéance, approvisionnement après l'échéance
  - couches systèmes et matérielles transparentes pour le développeur
  - application conçue indépendamment de l'architecture

- Interface TDMA avec Ethernet standard (physical layer)
  - aucune collision par construction (une collision devient une erreur)
  - garantir par construction la ponctualité des communications
  - dimensionnement et ordonnancement statique du réseau
    - pour vérifier et valider le réseau hors-ligne
    - pour détecter un comportement fautif

=> Cadencement global des communications connu de tous

Concept de l'extension du modèle d'exécution

- L'échéance réseau
  - Prise en compte des délais de transmission
  - Surcontraindre l'échéance d'un traitement
  - Libération d'une fenêtre de transmission des données