ALASCA — Architecture logicielles avancées pour les systèmes cyber-physiques autonomiques

© Jacques Malenfant

Master informatique, spécialité STL – UFR 919 Ingénierie

Sorbonne Université
`Jacques.Malenfant@lip6.fr`

Cours 3

Modélisation des systèmes de contrôle cyber-physiques

Objectifs pédagogiques du cours 3

- Adopter une vision **théorie des systèmes** pour analyser les systèmes cyber-physiques autonomiques et en comprendre les apports dans leur conception et leur mise en œuvre.
- Comprendre la modélisation **quantitative** des systèmes traditionnellement utilisée par les spécialistes de ce domaine et les automaticiens ainsi que les relations qui existent avec les systèmes cyber-physiques.
- Comprendre les différents modèles de systèmes : **continus**, **discrets**, **hybrides**, leurs interrelations et leur utilisation pour **spécifier** et **modéliser** les systèmes auto-adaptables.
- Approfondir plus spécifiquement les **systèmes hybrides** comme modèles de comportement des systèmes cyber-physiques autonomiques, et leurs principales déclinaisons développées dans le domaine de l'informatique théorique sous le vocable **automates hybrides**.

“All models are wrong – but some models are useful.” *

— G.E.P. Box

G.E.P. Box, Dans *Robustness in the Strategy of Scientific Model Building*, R.L. Launer et G.N. Wilkinson, éditeurs, Academic Press, 1979.

* *« Tous les modèles sont faux, mais certains modèles sont utiles. »*

Plan

Modélisation par entrées/sorties des systèmes *continus*

- Considérons un ensemble de propriétés continues, modélisées par un ensemble de variables continues $v_1, \ldots, v_n$ et leurs domaines de valeurs (souvent les réels dans $\mathbb{R}$).
  Ex.: niveau de batterie, temps moyen de traitement des requêtes, . . .
- L'approche de modélisation par entrées/sorties demande d'abord de considérer une *partition* des $\{v_{i=1,\ldots,n}\}$ en :
  - variables *en entrée* $u_1, \ldots, u_p$ dont les valeurs sont le plus souvent *contrôlées*[1] (ex. : mémoire allouée, nombre de MV, ...), et
  - variables *en sortie* $y_1, \ldots, y_m$ dont les valeurs sont *observées* (ex. : temps moyen de traitement des requêtes).
- Prises dans un intervalle de temps $[t_0, t_f]$, chacune de ces variables évolue continûment en prenant différentes valeurs (contrôlées ou observées) qui vont ainsi définir des fonctions :

$$\{u_1(t), \ldots, u_p(t)\}, \{y_1(t), \ldots, y_m(t)\} \qquad t_0 \leq t \leq t_f$$

  Ce qu'on appelle également des *signaux*.

[1] Elles peuvent aussi inclure des phénomènes externes non-contrôlés appelées *perturbations*.

Comportement du système

- La modélisation formelle d'un système continu capture son évolution, son *comportement*, par l'évolution des variables de son modèle, décrite par des *fonctions mathématiques* de ses entrées dans ses sorties *i.e.*, un ensemble d'équations :

$$\begin{aligned} y_1(t) &= g_1(u_1(t), \ldots, u_p(t), t) \\ &\ldots \\ y_m(t) &= g_m(u_1(t), \ldots, u_p(t), t) \end{aligned}$$

  ou, dans une notation vectorielle

$$\mathbf{y}(t) = \mathbf{g}(\mathbf{u}(t), t)$$

  Notez qu'on suppose ici que toutes les entrées sont connues à l'avance $(u(t), \forall t \geq t_0)$, comme si on les lisait des données depuis un fichier prédéfini.

- Cette forme de modélisation a été intensivement étudiée pour décrire formellement des systèmes déterministes : mécaniques, électriques, pneumatiques, chimiques, biologiques, ...

La notion d'état

- Le principe selon lequel un système pourrait être totalement décrit par une fonction entre ses entrées et ses sorties est (le plus) souvent une trop forte simplification de la réalité.
- La plupart des systèmes exhibent une « mémoire » de leur évolution passée qui influe sur les sorties pour *une même entrée*.
- Cette mémoire peut être représentée par un ensemble de variables $\mathbf{x} = x_1, \ldots, x_k$ dont les valeurs $\mathbf{x}(t)$ à un instant donné dénotent un *état* du système.
- *Comment modéliser la dépendance du système et de son état envers son évolution passée ?*

  Il est logique de supposer que :
  - l'état $\mathbf{x}(t)$ dépend de l'état initial $\mathbf{x}(t_0) = \mathbf{x}_0$ du système et des entrées $\mathbf{u}(s), t_0 \leq s \leq t_f$ depuis le début du fonctionnement et que
  - les sorties $\mathbf{y}(t)$ dépendent de l'état courant $\mathbf{x}(t)$ et des entrées courantes $\mathbf{u}(t)$, l'état courant résumant donc tout le passé.

Équations modélisant la dynamique de l'état

Équations d'état

Dans un modèle, l'ensemble des équations requises pour spécifier l'état $\mathbf{x}(t)$ pour tout $t \geq t_0$, étant donnés l'état initial $\mathbf{x}(t_0) = \mathbf{x}_0$ et une fonction $\mathbf{u}(t), t \geq t_0$, des entrées, sont appelées *équations d'états*.

- Les équations d'états peuvent prendre de nombreuses formes, mais les systèmes continus ont été souvent étudiés sous l'angle des *équations différentielles* de la forme :

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t)$$

- Un modèle de système à espace d'états est alors défini par le système d'équations :

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t) \qquad \mathbf{x}(t_0) = \mathbf{x}_0 \\ \mathbf{y}(t) &= \mathbf{g}(\mathbf{x}(t), \mathbf{u}(t), t) \end{aligned}$$

Schéma de modélisation par espace d'états

$$\mathbf{u}(t) \longrightarrow \boxed{\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t)} \longrightarrow \mathbf{y}(t) = \mathbf{g}(\mathbf{x}(t), \mathbf{u}(t), t)$$

Où :

- $\mathbf{u}(t)$: entrées (contrôlées ou imposées par l'environnement) ;
- $\dot{\mathbf{x}}(t)$: dérivée (*i.e.*, taux instantané de l'évolution) de l'état ;
- $\mathbf{x}(t)$: état ;
- $\mathbf{y}(t)$: sortie.

Remarques

- La forme de modélisation précédente propose une approche élégante pour comprendre les systèmes, mais soulève deux difficultés centrales :
  - exprimer précisément le comportement d'un système par un ensemble de variables et d'équations *conformes à la réalité*, et
  - résoudre ces équations de manière à pouvoir calculer et prédire de manière sûre le comportement du système grâce au modèle (*i.e.*, obtenir les fonction $\mathbf{x}(t)$ et $\mathbf{y}(t)$ pour essayer d'en déduire le comportement *réel* du système).
- En particulier, de nombreux systèmes nécessitent des équations (différentielles) si complexes que les mathématiciens *n'arrivent pas* à les résoudre *analytiquement*.
  Deux alternatives s'offrent alors aux modélisateurs :
  1. le calcul numérique du modèle pas-à-pas sur $[t_0, t_f]$, ou
  2. la simplification du modèle pour retomber sur des équations qu'on saura résoudre analytiquement dont, par exemple, l'approximation *linéaire* qui a été beaucoup étudiée et utilisée en pratique.

Systèmes stochastiques

- La plupart des systèmes réalistes ne sont pas *déterministes*.
  - Beaucoup exhibent des *variations aléatoires* plutôt limitées autour d'un comportement de base déterministe, ce que l'on peut assimiler à des *erreurs* (par exemple, du « jeu » dans une mécanique).

    ⇒ *Peuvent être analysés plus facilement, surtout si l'erreur est du « bruit blanc »*.
  - D'autres exhibent des *comportements aléatoires* intrinsèques comme, par exemple, le délai entre les arrivées des clients à un guichet (phénomène discret) ou encore la variation de la bande passante d'un réseau sans fil (phénomène continu).
- Capturer ces phénomènes exige des *modèles stochastiques*.

Systèmes stochastiques

Systèmes dont certains comportements peuvent se produire à des moments aléatoires en temps continu, et causer des évolutions aléatoires qui doivent être modélisés de manière stochastique.

Plan

1. Théorie des systèmes
2. Systèmes hybrides
3. Modélisation hybride d'un système auto-adaptable
4. Modélisation hybride modulaire

Modèles de comportement discrets

- En informatique théorique et en programmation concurrente, le comportement des systèmes a été étudié en considérant des *espaces d'états* et de *transitions discrets*.
  - Les modèles discrets développés à cet effet sont les *automates*, déclinés dans de nombreuses formes (AEF déterministes, non-déterministes ou probabilistes ; réseaux de Petri ; ...).
  - Ces modèles sont ensuite utilisés pour étudier le comportement puis vérifier les systèmes (*model-checking*, par exemple), c'est-à-dire en démontrer des propriétés comme la vivacité, l'atteignabilité de certains états, etc.
- Mais lorsque ces modèles sont appliqués à des systèmes qui ont des composantes continues (ou discrètes mais infinies), ils forcent une *discrétisation* et une restriction à des ensembles finis qui ne capturent pas toujours assez précisément les propriétés essentielles du système.

Vers des modèles comportementaux hybrides

- Ainsi, les modèles à base d'automates ne prennent en compte que le comportement par événements discrets se produisant en temps discret sur des espaces d'états discrets et finis.
- Les systèmes auxquels nous sommes confrontés ont aussi des comportements en temps continu, pouvant être modélisés par un espace d'états continu soumis :
  - à des *transitions* par événements définis par des paramètres *discrets <u>et</u> continus* et se produisant en *temps continu* mais à des instants *ponctuels*[1] ;
  - à des *évolutions* en temps continu pouvant être décrites par les équations continues de la théorie des systèmes classique.
- Une modélisation complète de ce type de systèmes exige des approches combinant *événements* (ponctuels mais à paramètres continus) en temps continu avec des *équations continues*.

[1] donc bien une notion d'événement *différente* de celle de modèles discrets car n'étant plus définie par des espaces finis ou même dénombrables.

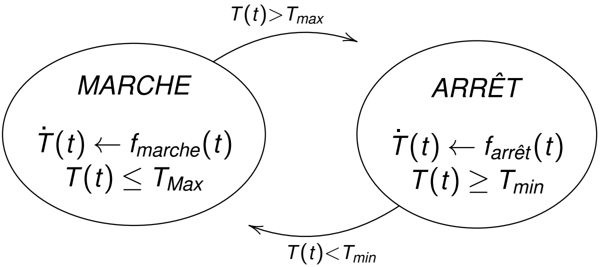
Théorie des systèmes hybrides

Définition

La théorie des système hybrides étudie le comportement des systèmes et les formalismes de modélisation qui impliquent à la fois des évolutions continues sur des espaces continus, décrites par des équations continues, et des transitions discrètes ou ponctuelles en temps continu décrites par des automates à états.

- Objectif affiché : analyser, vérifier et optimiser selon des modèles rigoureux les systèmes de contrôle cyber-physiques et les systèmes autonomiques.

Exemple classique de système hybride : thermostat



- Deux modes de fonctionnement : *MARCHE* (chauffe) ou *ARRÊT*.
- Deux seuils :
  - T_{max} pour passer de l'état *MARCHE* à l'état *ARRÊT*, et
  - T_{min} pour l'inverse

  avec $T_{min} < T_{max}$ *i.e.*, avec inertie ou *hystérésis*.
- Deux systèmes d'équations différentielles distincts modélisent l'évolution de la température selon le mode courant de fonctionnement.

Observations

- La nature hybride du système de chauffage thermostatique vient du fait qu'il possède
  - un espace d'états continu, dont la température ambiante,
  - et un espace d'états discret, son mode de fonctionnement,

  qui *s'influencent l'un l'autre*.
- Le comportement continu (température) dépend du comportement discret (mode de fonctionnement) car l'évolution de la température dépend du fait qu'on chauffe (marche) ou non (arrêt).
- Le comportement discret (transition de modes) résulte du comportement continu (température), dans la mesure où le basculement d'un mode à l'autre dépend de la température ambiante par les deux seuils T_{min} et T_{max}.
  - Note : ici, ces seuils sont liés à des seuils choisis mais dans d'autres cas, ils peuvent aussi apparaître naturellement ; par exemple, une cuve qui déborde...

Interactions continu/discret

- Vision hybride : systèmes dont l'évolution continue est interrompue par des phénomènes discrets dont l'impact est d'abord de changer l'état discret mais aussi l'état et le comportement continu (entrées, sorties, état, équations).
- Deux formes de transitions discrètes du comportement continu :
  1. saut : changement *ponctuel* d'*état* (discontinuïté d'état) ;
  2. commutation : changement *ponctuel* de *modèle d'évolution* (discontinuïté de modèle *i.e.*, nouvelles équations).
- Quatre types de phénomènes discrets hybrides [Branicky, 2005] :
  1. *saut autonome* : saut provoqué de manière endogène (ex.: franchissement du seuil cuve pleine) ;
  2. *commutation autonome* : commutation provoquée de manière endogène (ex.: évolution du niveau d'eau dans la cuve) ;
  3. *saut contrôlé* : saut provoqué par une modification d'une variable d'entrée (contrôlée) ; et,
  4. *commutation contrôlée* : commutation provoquée par une modification d'une variable d'entrée (contrôlée).

Plan

Exemple fil rouge : MolèNE[1]

- Composants échangeant des données via réseau WiFi :
  - deux composants échangent des données lourdes (images), dont l'un sur un PC (donc sur batterie) ;
  - possibilité de compresser les données et les décompresser pour accélérer la transmission sous bande passante faible ;
  - mais compresser coûte de la batterie car cela nécessite du calcul, donc à éviter si on veut garder le PC plus longtemps en fonction.
- Objectif de l'adaptabilité :
  - Améliorer le taux de transfert des données (incluant compression et décompression) tout en maintenant une durée de fonctionnement du PC sur batterie la plus longue possible.
- Phénomènes que l'on cherche à adapter (contrôler) :
  - taux de transfert des données en $Mbits/s$, influencé par la compression des données (contrôlée) mais aussi par la bande passante (subie) ;
  - taux d'attrition de la batterie en mAh/s, en inhibant le mode compression qui exige beaucoup de calculs.

[1] Inspiré de la thèse de Maria-Teresa Segarra, IRISA/Rennes I, 2000.

Éléments de modélisation I

- Données du modèle :
  - $p(t)$: bande passante du réseau sans fil en $Mbits/s$, $0 \leq p(t) \leq P_{max}$, avec interruptions aléatoires.
  - ΔB_{NC} (resp. ΔB_C, ΔB_0) : taux d'attrition de la batterie sans compression (resp. avec compression, sans réseau) en mAh/s
  - v_c : vitesse de compression en $Mbits/s$
  - v_d : vitesse de décompression en $Mbits/s$
  - τ_c : taux de compression moyen des données, $0 < \tau_c < 1$
- Évolutions contrôlées :
  1. taux de transfert des données τ_t selon le mode choisi (avec ou sans compression) et la bande passante courante $p(t)$.
  2. consommation batterie supposée déterministe selon le mode :

$$\dot{b}(t) = -\Delta B_{NC} \qquad \dot{b}(t) = -\Delta B_C \qquad \dot{b}(t) = -\Delta B_0$$

Éléments de modélisation II

- Évolutions non-contrôlées :
  1. bande passante $\Rightarrow$ ED stochastique.

$$\dot{p}(t) = \sigma(p(t))d\mathcal{P}(t)$$

$$d\mathcal{P}(t) \sim \mathbf{Exp}[\lambda_p] \in [0,\infty[\qquad 1/\lambda_p = \text{moyenne}$$

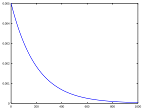
$$\sigma(p(t)) = \begin{cases} -1 & \textbf{if } u(t) < p(t)/P_{max} \\ 0 & \textbf{if } u(t) = p(t)/P_{max} \\ 1 & \textbf{if } u(t) > p(t)/P_{max} \end{cases} \qquad \text{avec } u(t) \sim \mathbf{U}[0,1] \in [0,1]$$

  2. interruptions du réseau : modèle de pannes

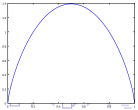
$$\text{durée entre les interruptions} = x_1 \sim \mathbf{Exp}[\lambda_1]$$

$$\text{durée des interruptions} = x_2 \sim \mathbf{Exp}[\lambda_2]$$

$$\text{bande passante à la reprise} = rP_{max} \text{ où } r \sim \mathbf{Beta}[\alpha_p, \beta_p] \in [0,1]$$



Exp[200]



Beta[1.75, 1.75]

Adaptations

- Trois opérations d'adaptation :
  1. Mise en place de la compression.
  2. Retrait de la compression.
  3. Passage en mode « batterie faible » (compression interdite).
  - Intuitivement, il existe une BP seuil p_s au-dessus de laquelle le taux de transfert est plus élevé sans compression mais en-dessous de laquelle il est plus élevé avec.
- Choix de politique d'adaptation (loi de contrôle) :
  - Politique à seuils avec hystérésis ($P_{inf} < P_{sup}$) pour la compression :
    - $p \geq P_{sup} > p_s$: suppression de la compression
    - $p \leq P_{inf} < p_s$: mise en place de la compression
    - ex.: $P_{sup} = 25$ Mbits/s et $P_{inf} = 21$ Mbits/s
  - Politique à seuil B pour la batterie : *le niveau baissant continument, pas besoin d'hystérésis*, un seul seuil suffit
    - $B < b \leq B_{max}$: autoriser la compression
    - $b \leq B < B_{max}$: ne plus autoriser la compression
    - ex.: $B_{max} = 5000$ mAh, seuil $B = 2000$ mAh

Estimation du seuil p_s de BP par le taux de transfert

- Soit K *Mbits* de données à transmettre, la durée d nécessaire est :

$$d = \begin{cases} \frac{K}{p} & \neg\, compression \\ \frac{K}{v_c} + \frac{\tau_c K}{p} + \frac{\tau_c K}{v_d} & compression \end{cases}$$

- Quelques manipulations donnent le taux de transfert :

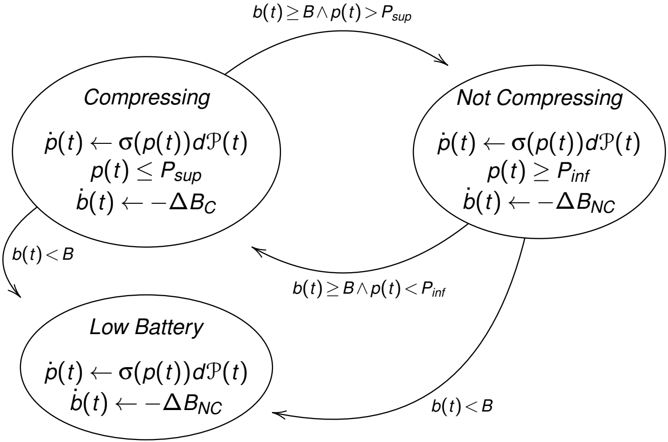
$$\tau_t = \begin{cases} p & \neg\, compression \\ \frac{v_c p v_d}{p v_d + \tau_c v_c v_d + \tau_c p v_c} & compression \end{cases}$$

- Et le seuil p_s justifiant le passage d'un mode à l'autre :

$$p_s = \frac{(1 - \tau_c) v_c v_d}{v_d + \tau_c v_c}$$

Ex.: pour $v_c = 50\, Mbits/s$, $\tau_c = 0.4$, $v_d = 75\, Mbits/s$, on trouve $p_s \approx 23.68\, Mbits/s$

Présentation graphique (sans interruptions réseau)



Nota : un système hybride monolithique
« incluant » le contrôle dans ses conditions de transition.

Plan

Un domaine de recherche en pleine ébullition

- Sur une même base conceptuelle, pléthore de choix possibles.
- Donc, sans surprise, foison de modèles issus de la recherche.
- État actuel de la recherche : exploration des possibles.
- Quatre principales approches ou communautés de recherche :
  - Informatique théorique : automate hybride (Henzinger, Lynch) ; vivacité ; composition.
  - Mathématiques : garanties sur l'existence de solutions ; vers des formulations uniformisant évolutions discrètes et continues.
  - Automatique : contrôle continu/discret ; couplage entre deux systèmes hybrides, contrôlé (*plant*) et contrôleur.
  - Décision : synthèse de contrôleur optimaux ; modèles stochastiques ; processus de décisions markoviens (PDM) et apprentissage par renforcement.
- Notre objectif étant de modéliser des architectures logicielles *modulaires*, nous allons maintenant nous intéresser aux approches hybrides adaptées à la modélisation *modulaire* : les *automates hybrides*.

Automates hybrides d'Henzinger

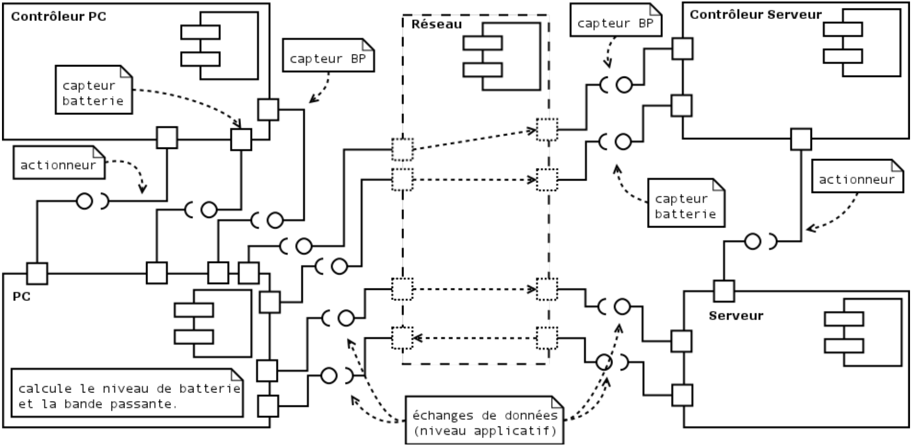
Travaux remontant essentiellement aux années '90 :

- *espace d'états* discrets potentiellement *infini* incluant des *variables continues* ;
- à la fois des *événements* (appelés actions) *discrets* et des *évolutions continues* ;
- ensembles de variables *homogènes* entre états « discrets » (modes) ;
- description du comportement par *traces* alternant *transitions discrètes* et *fonctions continues* ;
- pas de restriction sur la manière de définir les trajectoires continues (équations, équations différentielles, . . .) ;
- *composition* d'automates par *partage* d'événements et de variables ;
- notion de *vivacité* du comportement (appelée réceptivité).

Familles d'automates hybrides E/S de Lynch et al.

- Hybrid I/O Automata (mi '90 $\rightarrow$ 2003) :
  - partition des événements et des variables en internes ou externes, puis les externes caractérisés comme en entrée (importée) ou en sortie (exportée) ;
  - composition d'HIOA :
    - un seul modèle exportateur pour chaque variable externe ;
    - connexion d'événements et de variables en sortie d'un HIOA avec des événements et variables en entrée d'un autre.
- Timed I/O Automata ($\subseteq$ HIOA, $\neq$ timed automata)
  - pas de communication par connexion de variables continues, seulement par connexion d'événements (mais pouvant inclure des valeurs ponctuelles de variables continues) ;
  - tient compte de la nature discrète des échanges par réseau.
- Vision conception logicielle/système :
  - HIOA : modélisation modulaire d'artéfacts *monolithiques centralisés*.
  - TIOA : modélisation modulaire d'artéfacts *décentralisés*.
  - Analogie avec les GALS : composition de TIOA formés par composition d'HIOA où toutes les variables continues sont internes.

Exemple MolèNE implanté par des composants

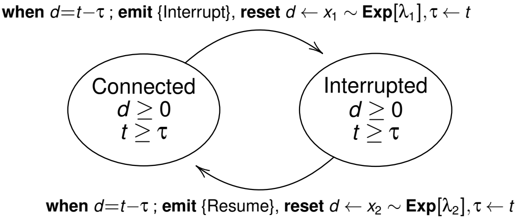


- Les deux contrôleurs prennent des décisions identiques sur la base de mêmes données puis les deux composants PC et Serveur se coordonnent pour réaliser les adaptations.

Modélisation modulaire

- Modélisation par automate hybride précédente : monolithique, peut rapidement devenir complexe, peu réutilisable.
- Passage à une modélisation modulaire, plus fidèle à l'architecture logicielle, en utilisant l'approche HIOA/TIOA.
- Illustration sur l'exemple :
  - Modèle de bande passante exportant des événements d'interruption et de reprise et la variable p (lié au PC chargé de la métrique réseau).
  - Modèle du PC émettant la variable b et important les événements de contrôle.
  - Modèles de contrôleur côté PC et serveur qui importent les variables p et b et émettent des événements de contrôle forçant le passage à la compression, le retrait de la compression ou le mode batterie faible.
  - Modèle du serveur important les événements de contrôle.

Bande passante : modèle des interruptions du réseau



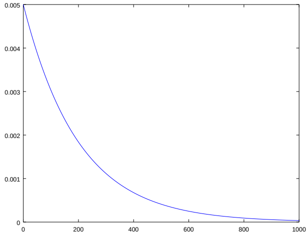
t représente le temps.

Importe	Exporte
	Interrupt Resume

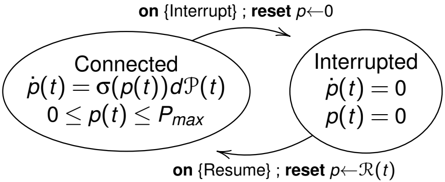
où :

$$1/\lambda_1 = \text{durée moyenne des interruptions}$$
$$1/\lambda_2 = \text{temps moyen entre les interruptions}$$
$$t_0 = 0$$
$$d_0 = 0$$
$$\tau_0 = 0$$
$$s_0 = (\text{Interrupted}, d_0, \tau_0)$$

Exp[200]



Bande passante : évolution continue



Importe	Exporte
Interrupt Resume	
	p

où :

$$d\mathcal{P}(t) \sim \textbf{Exp}[\lambda_p] \in [0, \infty[$$

$$\lambda_p = \text{dérivée moyenne de la bande passante}$$

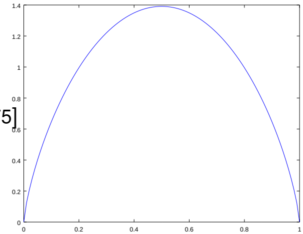
$$\sigma(p(t)) = \begin{cases} -1 & \textbf{if } u(t) < p(t)/P_{max} \\ 0 & \textbf{if } u(t) = p(t)/P_{max} \\ 1 & \textbf{if } u(t) > p(t)/P_{max} \end{cases}$$

$$\text{avec } u \sim \textbf{U}[0,1] \in [0,1]$$

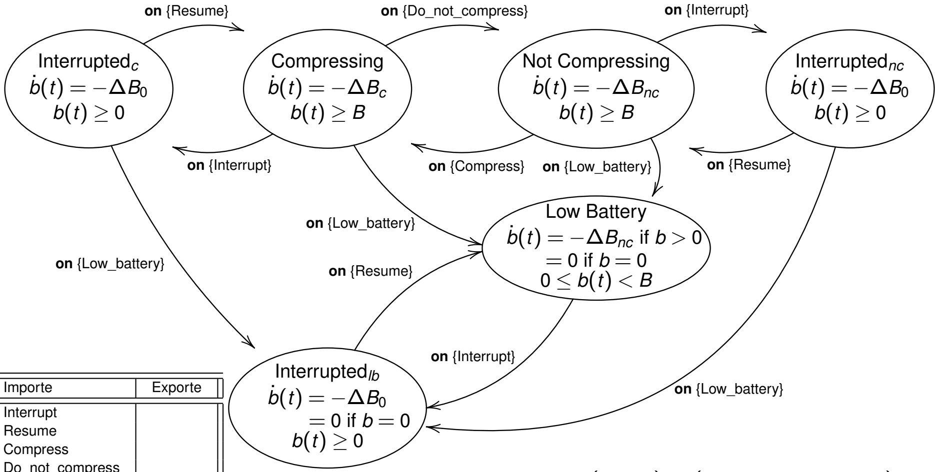
$$\mathcal{R}(t) = r(t)P_{max}, \text{avec } r(t) \sim \textbf{Beta}[\alpha_p, \beta_p] \in [0,1]$$

$$s_0 = (q_0, p_0) = (\text{Interrupted}, 0)$$

Beta$[1.75, 1.75]$



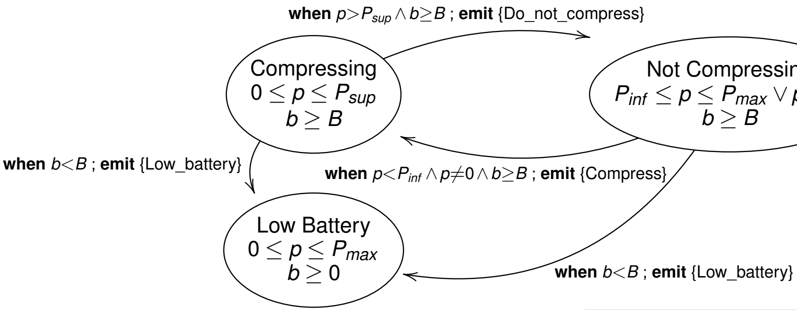
Modèle du PC : niveau de la batterie



Importe	Exporte
Interrupt Resume Compress Do_not_compress Low_battery	
	b

où : $s_0 = (q_0, b_0) = (\text{Interrupted}_{nc}, B_{max})$

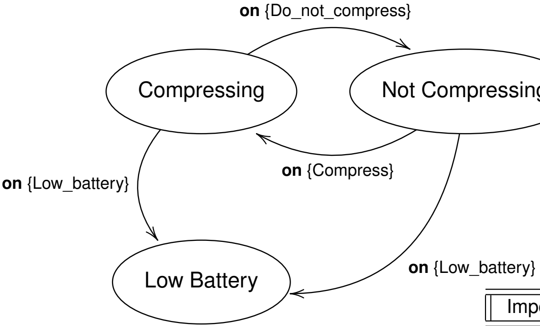
Modèle du contrôleur du PC (et côté serveur) : décisions



Importe	Exporte
	Compress Do_not_compress Low_battery
p, b	

où $s_0 = (q_0, p_0, b_0) = (\text{Not Compressing}, p_0, b_0)$
avec p, b importées

Modèle du serveur

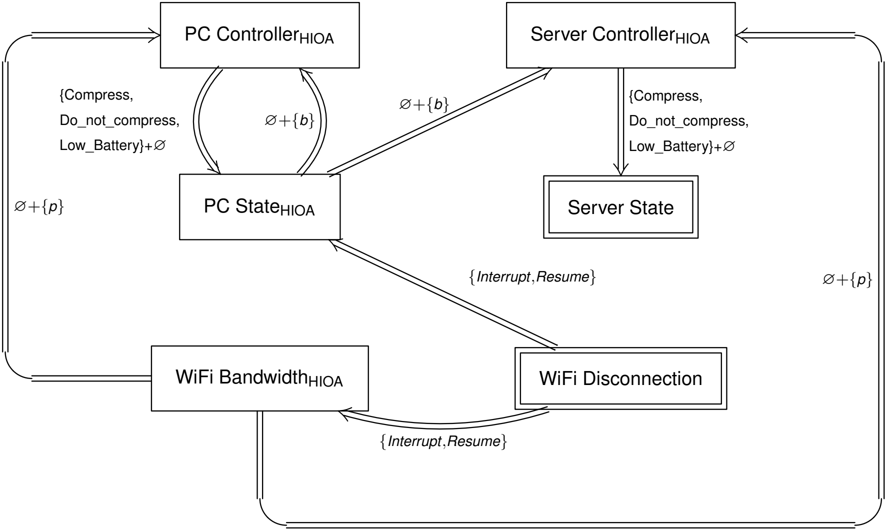


Importe	Exporte
Compress Do_not_compress Low_battery	

où $s_0 = q_0 =$ Not Compressing

- En pratique, le serveur se contente mettre en place la compression ou la retirer.

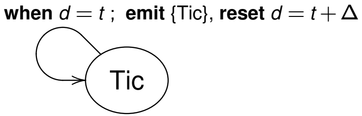
Composition du modèle complet (HIOA+TIOA)

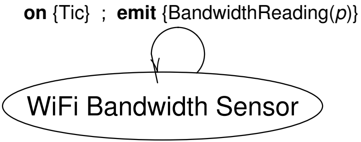


Réduction du couplage : vers une composition de TIOA

- La décomposition en HIOA implique le partage de variables continues et donc des modèles *fortement couplés, centralisés*.
- Pourquoi ? Observations :
  1. Modélisation irréaliste puisque l'achitecture logicielle correspondante *doit* être répartie entre le portable et le serveur.
  2. Or, aucun procédé numérique ne permet à un contrôleur informatique d'accéder *en continu* à la bande passante : il y aura nécessairement *échantillonnage* discret.
- Comment découpler les modèles ?
  - Former des TIOA en introduisant des modèles « échantillonneurs » : en quelque sorte des modèles de *capteurs*.
  - Ils s'interposent pour « lire » les valeurs des variables continues b, p et fournir des événements BandwidthReading et BatteryReading portant des valeurs *ponctuelles*.

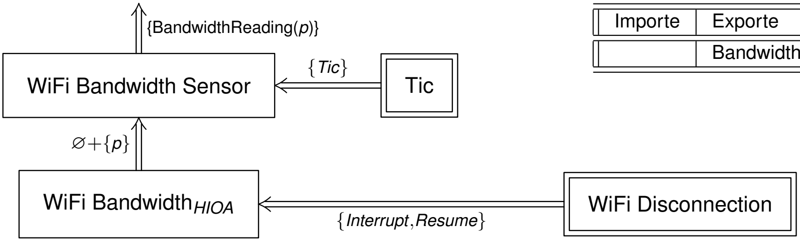
Modèle TIOA de bande passante





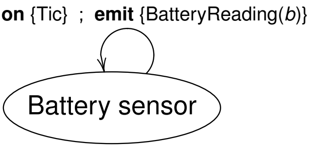
Importe	Exporte
	Tic

Importe	Exporte
Tic	BandwidthReading
p	



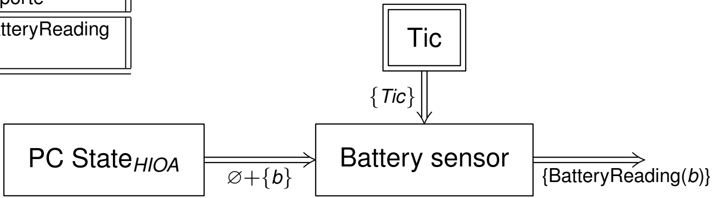
Importe	Exporte
	BandwidthReading

Modèle TIOA du PC

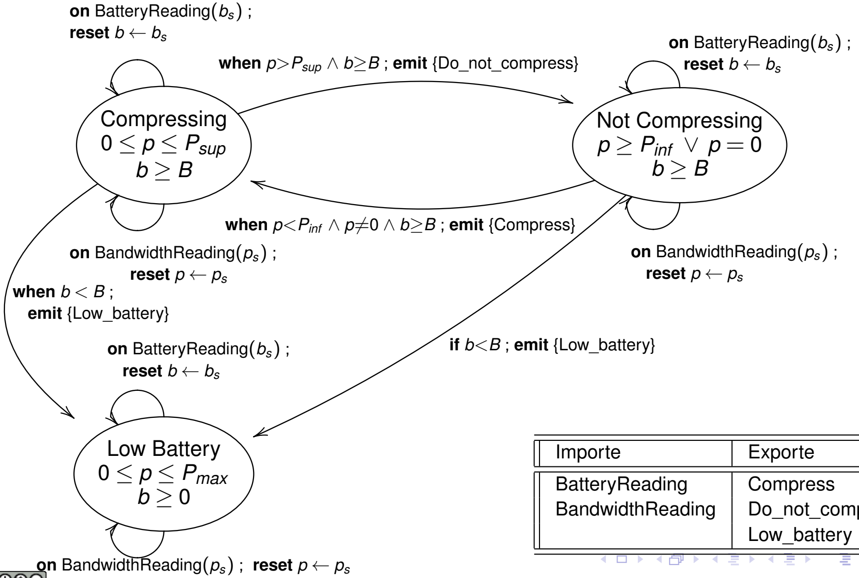


Importe	Exporte
Tic	BatteryReading
b	

Importe	Exporte
Interrupt Resume	BatteryReading

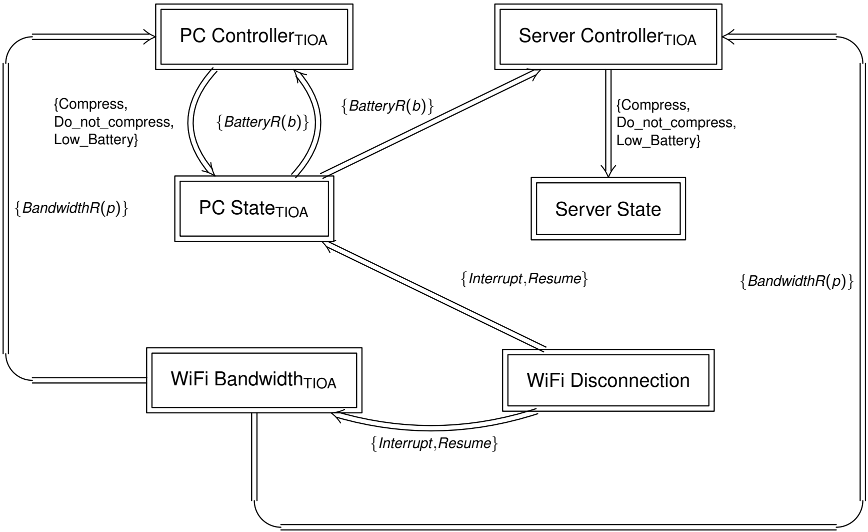


Modèle TIOA du contrôleur du PC



Importe	Exporte
BatteryReading BandwidthReading	Compress Do_not_compress Low_battery

Modèle complet décomposé en TIOA uniquement



Introduction de la communication par réseau

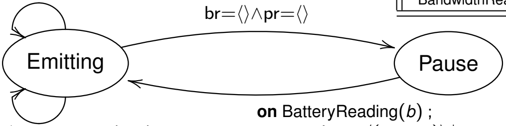
- Le modèle précédent satisfait le besoin de découplage, mais n'est toujours pas réaliste.
- S'il y a un contrôleur côté serveur, les événements BatteryReading(b) doivent transiter par le réseau et donc être retardés par un certain délai de transmission.
- Si la bande passante n'est mesurée que d'un côté (cohérence), alors les événements BandwidthReading(p) devront aussi être transmis via le réseau.
- Comment modéliser cela ?

  $\Rightarrow$ Par un modèle recevant des événements puis les réémettant après un délai aléatoire.
- Illustre le potentiel des modèles hybrides (++) qui sont des modèles de calcul à part entière.

Modèle de transmission réseau

Importe	Exporte
BatteryReading BandwidthReading	BatteryReading BandwidthReading

when $t = e_t(\mathrm{br}, \mathrm{pr})$; **emit** $e_v(\mathrm{br}, \mathrm{pr}, t)$, **reset** $(\mathrm{br}, \mathrm{pr}) \leftarrow r_2(\mathrm{br}, \mathrm{pr}, t)$



on BatteryReading(b) ;
reset br $\leftarrow (t+\tau, b)\S\,\mathrm{br} \mid \tau \sim \mathbf{Gamma}(\kappa, \theta)$
on BandwidthReading(p) ;
reset pr $\leftarrow (t+\tau, p)\S\,\mathrm{pr} \mid \tau \sim \mathbf{Gamma}(\kappa, \theta)$

on BatteryReading(b) ;
reset br $\leftarrow \langle (t+\tau, b)\rangle \mid \tau \sim \mathbf{Gamma}(\kappa, \theta)$
on BandwidthReading(p) ;
reset pr $\leftarrow \langle (t+\tau, p)\rangle \mid \tau \sim \mathbf{Gamma}(\kappa, \theta)$

where :

$$\mathrm{br}(t_0) = \langle\rangle \qquad \mathrm{pr}(t_0) = \langle\rangle$$

$$e_t(l_1, l_2) = \min\{x \mid (x, \cdot) \in l_1 \vee (x, \cdot) \in l_2\}$$

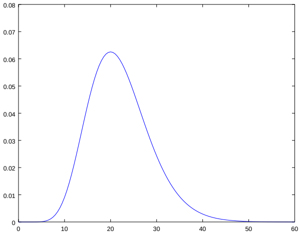
$$e_v(l_1, l_2, t) = \begin{cases} \text{BatteryReading}(b) & \text{if } (t, b) \in l_1 \\ \text{BandwidthReading}(p) & \text{if } (t, p) \in l_2 \end{cases}$$

$$r_2(l_1, l_2, t) = \begin{cases} (r_1(l_1, t), l_2) & \text{if } (t, \cdot) \in l_1 \\ (l_1, r_1(l_2, t)) & \text{if } (t, \cdot) \in l_2 \end{cases}$$

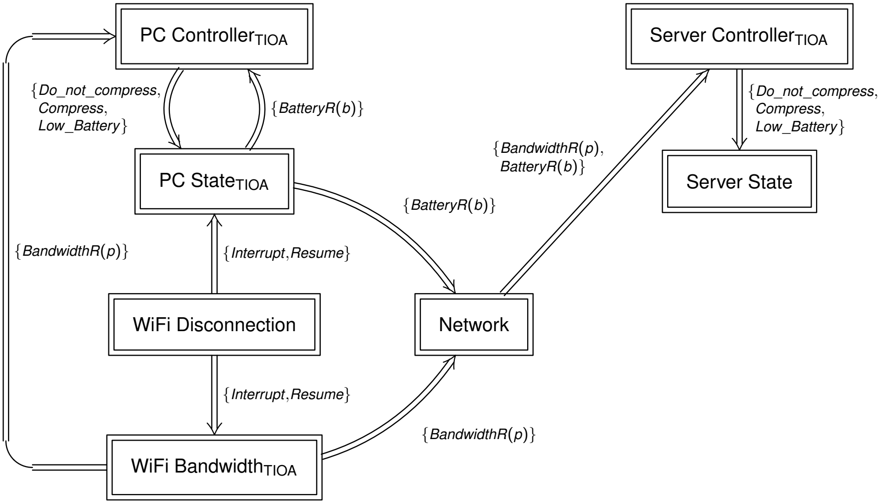
$$r_1(l, t) = \begin{cases} \langle\rangle & \text{if } l = \langle\rangle \\ l_r & \text{if } l = (t, \cdot)\S l_r \\ (z, x)\S r_1(l_r, t) & \text{if } l = (z, x)\S l_r \wedge z \neq t \end{cases}$$

Gamma$[11, 2]$

mean =
22 msec.



Modèle complet avec répartition explicite



Récapitulons...

1. En ***théorie des systèmes***, les ***modèles quantitatifs***, à base d'états, permettent d'appréhender les variations dans les valeurs des propriétés des systèmes (comme la qualité de service) et donc d'introduire des possibilités de ***contrôle*** de leurs propriétés.
2. Les modèles quantitatifs classiques de la théorie des systèmes sont fondés sur des ***variables d'états continues***, des ***lois de contrôles continues*** et des évolutions exprimées par des ***équations différentielles***.
3. L'informatique et d'autres disciplines similaires ont aussi développé des modèles de ***systèmes discrets***, à base d'***automates***, d'états discrets et d'événements provoquant des transitions à des instants ponctuels, en temps continu ou discret.
4. Les systèmes ***cyber-physiques*** mélangent des aspects ***discrets et continus*** ; ils nécessitent des modèles capables de capturer tous ces phénomènes à la fois, en interaction : les ***systèmes hybrides***.

Pour aller plus loin : sélection de lectures recommandées

- *Introduction to Discrete-Event Systems*, C.G. Cassandras et S. Lafortune, Springer, 2008, chapitres 1 et 2.
  écrit pour des informaticiens, mais ne touche qu'aux systèmes discrets puis temporisés.
- *Introduction to hybrid systems*, W.P.M.H. Heemels, D. Lehmann, J. Lunze et B. De Schutter, chapitre 1 de *Handbook of Hybrid Systems Control — Theory, Tools, Applications*, Cambridge University Press, 2009.
  texte accessible, bien qu'écrit pour des mathématiciens.
- *Hybrid I/O Automata*, Nancy Lynch, Roberto Segala and Frits Vaandrager, Information and Computation 185, 2003, pages 105–157.
- *Timed I/O Automata : A Mathematical Framework for Modeling and Analyzing Real-Time Systems*, Dilsun K. Kaynar, Nancy Lynch, Roberto Segala and Frits Vaandrager, Proceedings of the 24th IEEE International Real-Time Systems Symposium, 2003, pages 166–177.
- *Stochastic Hybrid Systems Meet Software Components for Well-Founded Cyber-Physical Systems Software Architectures*, Jacques Malenfant. European Conference on Software Architecture (ECSA), September 9-13, 2019, Paris, France.