Exemples de systèmes temps réel et choix d'implémentation

Françoise Simonot-Lion, Yvon Trinquet

Exemples de systèmes temps réel et choix d'implémentation

Françoise Simonot-Lion et Yvon Trinquet

Mots-clés : systèmes temps réel, implantation d'une application, approche asynchrone, exécutif temps réel, approche synchrone.

Résumé : ce chapitre présente les concepts de base liés aux systèmes temps réel. Après avoir illustré la problématique correspondante au travers d'exemples variés, il introduit les concepts techniques développés dans les chapitres suivants en se focalisant sur l'importance du choix d'implémentation des systèmes.

1 Exemples d'application temps réel

Comme cela a été montré dans l'introduction, le terme « temps réel » peut être utilisé pour plusieurs catégories d'applications[1]. Nous présentons ci-dessous trois exemples illustratifs de ces catégories, avec pour objectif d'identifier à chaque fois des problèmes spécifiques.

1.1 Application de contrôle de production

Soit, à titre d'exemple, une application de fabrication d'un produit illustrée par la figure 1. On ne donne que peu de détails sur le cahier des charges, l'exemple étant de compréhension assez intuitive. Le procédé sert à fabriquer un produit liquide, selon une recette, par :

- le mélange, selon une proportion définie par la recette, de deux produits liquides (bacs A et B, et C pour le mélange). La disponibilité des produits de base est supposée être assurée par une régulation à seuils pour chaque bac (capteurs à seuil Nmax et Nmin), à l'aide des vannes d'alimentation tout-ou-rien. La proportion est calculée à l'aide du capteur de niveau du bac C ;
- la dissolution de « pains » convoyés par un tapis roulant, en nombre défini par la recette. Un détecteur permet de connaître l'arrivée d'un pain soluble dans le bac, le tapis roulant étant à commande tout-ou-rien ;
- le brassage du mélange (hélice à commande tout-ou-rien) durant un temps défini par la recette, pour en assurer l'homogénéité.

L'ensemble des paramètres de la recette de fabrication sont variables (données utilisateur pour chaque lot de fabrication). Les pains solubles sont fabriqués par une autre unité en amont et le produit résultant est conditionné par une autre unité en aval, ces deux unités n'étant pas évoquées dans cet exemple.

Figure 1. Synoptique du procédé de fabrication

Diverses méthodes ont été proposées pour faire l'analyse d'un cahier des charges et en déduire par exemple l'architecture logicielle (ensemble d'entités logicielles coopérantes) s'exécutant sur une architecture matérielle donnée (ensemble de processeur(s) et réseau(x)). Il existe également des techniques permettant la conception conjointe (*co-design*) du matériel et du logiciel par partitionnement des fonctionnalités à assurer (Calvez, 1990), (Comete, 1998). Nous ne nous intéressons ici qu'au cas de la conception du logiciel pour une architecture matérielle donnée, ceci sans faire référence à une méthode particulière de conception.

[1] Les appellations « application temps réel » et « système temps réel » sont souvent utilisées comme synonymes dans ce texte. Elles désignent alors l'intégralité du système informatique réalisé. Lorsqu'on souhaite les différencier, on peut considèrer que « l'application » représente le logiciel créé par l'utilisateur pour le besoin applicatif spécifique, alors que le « système » représente les logiciels du support d'exécution (exécutif temps réel, pile de communication …).

1.1.1 Architecture fonctionnelle de l'application

L'analyse du cahier des charges de cette simple application, conduit à la mise en évidence des fonctionnalités à assurer par le système de contrôle. Ces fonctionnalités peuvent être réalisées au travers d'un ensemble de fonctions logicielles, dont l'exécution dépend de certaines conditions événementielles qui traduisent les coopérations entre les fonctions. La spécification du comportement est naturelle et lisible si on l'exprime sous formes d'actions parallèles, mais ce parallélisme d'expression repose sur l'hypothèse implicite que chaque traitement est réalisé sur un processeur dédié, ce qui ne sera pas forcément vrai à l'exécution. La figure 2 propose un exemple de solution (d'autres sont possibles), dans lequel on peut recenser les actions suivantes :

- deux actions, « CTR_Bac_A » et « CTR_Bac_B », pour maintenir le niveau entre deux seuils pour les bacs A et B ;
- une action « Supervision » pour la gestion de la fabrication. Cette action coopère avec l'utilisateur et avec les actions de contrôle du bac C et du tapis roulant ;
- une action « CTR_Bac_C » pour le contrôle du mélange des deux produits et le brassage ;
- une action « CTR_TR » pour le contrôle du tapis roulant et la détection des pains chutant dans le bac C.

Figure 2. Un exemple d'architecture fonctionnelle, solution au problème

La figure 2 illustre l'architecture fonctionnelle résultante sous la forme d'un schéma à flots d'événements. Cette architecture est, comme son nom l'indique, une organisation d'éléments. Elle ne peut à elle seule décrire la solution ; notamment chacune des actions doit être décrite, quant à son comportement, par un algorithme. On trouve sur la figure 2 :

- des variables d'entrées/sorties (variables partagées) représentant des éléments physiques du procédé accessibles via les capteurs/actionneurs (niveau mesuré, commande de vanne, commande de moteur ...) ;
- des liens de coopération entre les actions ou entre les actions et l'environnement qui sont matérialisés par le concept d' « événement ». Par exemple, la signalisation du franchissement d'un seuil sur un capteur de niveau à seuil est en relation avec l'action de contrôle du bac correspondant ;
- des liens de coopération entre les actions ou entre les actions et l'environnement qui sont matérialisés par le concept de messagerie asynchrone (boîte aux lettres). Par exemple, la transmission au tapis roulant par l'action superviseur d'une requête d'envoi de *n* pains dans le bac C.

1.1.2 Architectures matérielle et opérationnelle de l'application

L'application représentée par la structure fonctionnelle de la figure 2 est supposée être exécutée via un support d'exécution. Ce dernier doit fournir les moyens d'exécuter les algorithmes des actions, mais fournir également les services implémentant les diverses techniques de coopération entre les entités. Traditionnellement on découpe le support d'exécution en deux parties : celle qui concerne le support matériel d'exécution (processeur(s), réseau(x)), et celle qui concerne les services de base nécessaires à l'exécution des logiciels d'application (par exemple l'exécutif temps réel dans le cas de l'approche asynchrone).

En ce qui concerne l'architecture matérielle proprement dite, plusieurs configurations sont envisageables, même dans le cas de cet exemple simple. Le choix de la configuration et la sélection des équipements sont naturellement intimement liés, et outre les critères de coût ou de fiabilité, ils dépendent essentiellement des trois facteurs suivants.

Tout d'abord les contraintes « topologiques ». Ces contraintes apparaissent lorsque le procédé est une installation éclatée géographiquement, ou quand le procédé est doté d'une instrumentation intelligente qui en déporte certains traitements.

Un autre facteur concerne les contraintes de « sûreté de fonctionnement ». C'est l'impératif de maintien du service du système de contrôle (nominal ou dégradé) en cas d'anomalie partielle des équipements. Satisfaire cet impératif se traite par la redondance. Cette redondance peut concerner l'instrumentation mais elle concerne également le système informatique car sa défaillance engendre la perte totale du contrôle du procédé. Ainsi une architecture avec plusieurs calculateurs peut devenir nécessaire pour maintenir la continuité du service fourni en cas de défaillance d'un ou plusieurs calculateurs.

Enfin le dernier facteur concerne les contraintes « temporelles ». Comme les actions sont sollicitées par des événements (au sens large du terme) aux occurrences asynchrones, il se peut qu'à certains moments plusieurs actions entrent en concurrence pour

leur exécution. Si le nombre ou la puissance des processeurs du système de pilotage sont sous-dimensionnés, il se peut qu'ils ne puissent satisfaire les contraintes temporelles de toutes les actions en concurrence, c'est-à-dire répondre à toutes « à temps ». A titre d'illustration, pour notre exemple, des contraintes de temps de réaction maximal peuvent exister pour la fermeture des vannes (débordement d'un bac, précision pour le niveau admis dans le bac C ...), ou encore l'arrêt du tapis roulant. Pour éviter une situation dans laquelle une contrainte temporelle ne serait pas respectée, il faudra peut-être modifier l'architecture matérielle en conséquence : accroissement de la puissance des unités de traitement, utilisation de processeurs spécialisés, répartition des actions sur les processeurs, ou encore jouer sur les paramètres d'implémentation des actions, selon la technique utilisée (voir approches synchrone et asynchrone ci-dessous et la section sur l'ordonnancement).

Comme cela vient d'être évoqué, pour satisfaire une ou plusieurs de ces contraintes, on peut être amené à adopter un système informatique de pilotage composé de plusieurs stations de traitement interconnectées en réseau. Le réseau devient alors la colonne vertébrale des communications inter-systèmes et ses performances et capacités jouent, à leur tour, un rôle essentiel dans le respect des contraintes de sûreté et de temps. Cependant l'introduction des réseaux amène son lot de difficultés nouvelles qui n'existaient pas dans le cas monoprocesseur. En effet, lorsque les actions à entreprendre sur le procédé sont calculées en parallèle sur des calculateurs distincts, leurs observations de l'état du procédé peuvent différer de l'une à l'autre en raison de l'asynchronisme du fonctionnement des processeurs et des délais de communication. Dans ce cas, les actions appliquées au système séparément par ces stations peuvent ne pas être cohérentes entre elles. C'est pourquoi l'introduction d'un réseau dans l'architecture matérielle du système de commande est une donnée qui en conditionne également la réalisation logicielle. Ces deux aspects ne sont plus à dissocier et leur développement conjoint est nécessaire à la réalisation de l'application. L'aboutissement de ce développement conduit alors à la définition d'un matériel, d'une décomposition fonctionnelle et d'une répartition des actions implémentant les fonctionnalités sur le matériel, le tout garantissant les contraintes temporelles de l'application : ce résultat est appelé « architecture opérationnelle » de l'application. Il est bien évident qu'une telle architecture doit être validée par rapport aux contraintes mentionnées ci-avant, à la fois sur les aspects fonctionnels et non fonctionnels (temporels, sûreté de fonctionnement, etc.).

1.2 Application embarquée dans l'automobile

La part de l'électronique embarquée dans les véhicules automobiles s'accroît constamment. Le terme « électronique embarquée » recouvre, en fait, la mise en œuvre de techniques numériques pour réaliser des applications de contrôle. Certains de ces systèmes sont déjà opérationnels (système contrôlant l'injection au niveau du moteur, commande électronique des essuie-glaces, gestion des équipements multimédia intégrés dans l'habitacle, etc.). Actuellement, à l'instar de ce qui se passe dans l'avionique, émerge la notion de « tout électronique » (désigné plus souvent selon le terme anglais « *X-by-Wire* ») qui désigne le remplacement des systèmes de contrôle mécaniques et/ou hydrauliques par des systèmes uniquement numériques. Des études déjà avancées chez de nombreux constructeurs automobiles sont consacrées, notamment, au contrôle de direction ou « *Steer-by-Wire* » (Wilwert et al., 2004) dont un exemple simplifié est présenté ci-dessous. Il s'agit d'un système qui est, pour des raisons topologiques, entièrement distribué ; de plus, les lois de contrôle, par exemple celle qui commande la crémaillère d'entraînement de l'axe de direction, imposent des propriétés temporelles fortes au système. Enfin, le bon fonctionnement de ce système est crucial pour la sécurité du véhicule. Dans l'exemple qui suit, les noms utilisés dans la description sont volontairement tirés de l'application réelle.

1.2.1 Architecture fonctionnelle d'un système « Steer-by-Wire »

Fonctionnellement, ce système doit fournir, à tout instant, deux services. Le premier, σ_{FA} (FA : *Front axle Actuator*), a pour objectif de braquer les roues selon la requête du conducteur, captée au niveau du volant (angle de rotation par rapport à une position de référence et couple) ; cette requête constitue la consigne fournie à une loi de commande dont l'objectif est de commander la crémaillère d'entraînement de l'axe de direction, en fonction de valeurs représentatives de l'état du véhicule, comme sa vitesse, son adhérence et de l'état de l'environnement (état de la route, force du vent …).

La deuxième fonction, σ_{HW} (HW : *Hand Wheel*), fournit un retour d'effort au volant à partir du couple appliqué par le conducteur sur le volant et qui soit cohérent, notamment, avec l'état courant de la crémaillère.

1.2.2 *Architecture matérielle et opérationnelle d'un système « Steer-by-Wire »*

Une architecture informatique supportant les deux services listés ci-dessus est représentée à la figure 3.

Figure 3. Architecture informatique support d'un système "Steer-by-Wire"

Cette architecture physique très redondante inclut :

- les capteurs, C_HW1, C_HW2, C_HW3 (respectivement C_FA1, C_FA2, C_FA3), qui lisent de façon identique les consignes du conducteur (respectivement, la position de la crémaillère) ;
- les actionneurs, Moteur_ FA1, Moteur_ FA2 (respectivement, Moteur_ HW1, Moteur_ HW2) qui appliquent la même force d'entrainement (respectivement, le même retour de force) sur la crémaillère (respectivement, sur le volant) ;
- les calculateurs ou « *Electronic Control Unit* » (ECU), HW_ECU1 et HW_ECU2 (respectivement, FAA_ECU1 et FAA_ECU2) géographiquement implantés prés du volant (respectivement, près de la crémaillère) ;
- le bus redondé, dont les canaux sont désignés par BUS1 et BUS2 et sont gérés par un protocole à accès guidé par le temps (TDMA : *Time Division Multiple Access*). Dans ce type de protocole, l'intégralité de la bande passante est allouée à chaque station au cours d'un intervalle de temps prédéterminé (*time slot*) et un cycle TDMA correspond à la séquence prédéfinie des « time slots » ; ce cycle est indéfiniment répété.

Les capteurs C_HW1, C_HW2, C_HW3 (respectivement C_FA1, C_FA2, C_FA3) et les actionneurs Moteur_HW1, Moteur_HW2 (respectivement, Moteur_FA1, Moteur_FA2) sont reliés en point à point aux ECU HW_ECU1 et HW_ECU2 (respectivement, FAA_ECU1 et FAA_ECU2). Chaque ECU est connecté sur les deux canaux BUS1 et BUS2.

Dans la suite de l'exemple, nous n'étudions que l'implantation du service σ_{FA}. Les requêtes (demandes de braquage) du conducteur sont mesurées par les trois capteurs répliqués C_HW1, C_HW2, C_HW3 et lues par les applications implantées dans les ECU redondants HW_ECU1 et HW_ECU2. Chaque application réalise un vote sur les trois valeurs lues et produit une donnée, ou signal, dite « sécurisée ». Ce signal est transmis via les deux bus de communication BUS1 et BUS2. Les applications localisées sur les ECU redondants FAA_ECU1 et FAA_ECU2 consomment, à chacune de leur activation, un tel signal ainsi que la situation de vie[2] du véhicule et la dernière position de la crémaillère (fournie par les capteurs répliqués C_FA1, C_FA2, C_FA3) pour élaborer la consigne à donner aux actionneurs de la crémaillère.

1.2.3 *Caractéristiques temporelles du service σ_{FA} – Propriétés temporelles*

Les caractéristiques temporelles du service σ_{FA} concernent la période de production des signaux par les applications émettrices. L'application localisée sur l'ECU HW_ECU1 (respectivement, sur l'ECU redondant, HW_ECU2) produit les signaux Angle_HW et Couple_HW tous les ε_{HW} ; ces signaux sont regroupés au sein d'une trame de nom HW_Tr_ECU1 (respectivement HW_Tr_ECU2).

La loi de commande élaborant les consignes à l'actionneur de la crémaillère, doit être activée périodiquement selon une période définie par une étude amont d'automatique et égale à ε_{com}. A chacune de ses activations l'algorithme implantant la loi de commande consomme la dernière requête du conducteur transmise. La longueur du cycle de communication est donc ε_{com}. Notons de plus que le service σ_{HW} nécessite également l'émission de la trame FAA_Tr_ECU1 par l'ECU FAA_ECU1, et de son réplica par l'ECU redondant FAA_ECU2. Nous considérerons, ici, qu'elles sont produites avec une période $\varepsilon_{FAA}=\varepsilon_{HW}$. Une configuration possible du cycle de communication est visualisée à la figure 4.

Figure 4. Diagramme temporel des différentes activités

Propriété 1. La propriété de sûreté d'un véhicule est directement liée aux propriétés temporelles et de performances du système ; en effet, en raison du traitement et de la transmission par le réseau de la consigne conducteur, celle-ci est consommée par la loi de commande avec un certain retard par rapport à l'instant où cette consigne a effectivement été lue au volant. Des expériences sur banc de test et/ou des simulations permettent d'évaluer l'âge maximal, τ_{max}, que cette consigne peut avoir lors de son traitement par la loi de commande sans que la sécurité des passagers

[2] Situation de vie : ensemble des conditions instantanées appliquées au véhicule (vitesse, accélération, direction, adhérence de la route, ...).

et de l'environnement ne soit affectée. La propriété imposée au système est que, dans le pire des cas, l'âge de la donnée, c'est-à-dire le délai T entre la lecture de l'information au volant et sa consommation par l'ECU FAA_ECU1 (respectivement, l'ECU redondant FAA_ECU2) soit tel que $T < \tau_{max}$. Il s'agit donc de fixer certains paramètres afin de garantir toujours cette propriété ; par exemple, pour un ε_{com}, il faut fixer la taille et la place des trames dans les « time slots » d'un cycle et, éventuellement jouer sur la durée du traitement sur les nœuds HW_ECU1 et HW_ECU2 (durée d sur la figure 4) ainsi que sur la période d'activation de ces traitements (ε_{HW}, sur la figure 4).

Propriété 2. Les systèmes électroniques embarqués dans un véhicule sont très sensibles aux perturbations électromagnétiques provoquées, par exemple, par les puissants radars d'aéroport. Si on désire prouver des propriétés de sûreté du système dans un environnement potentiellement perturbé, on ne peut plus utiliser le principe de vérification de propriété vu ci-dessus puisque, en présence de perturbations, il existe un risque que des erreurs de transmission aient pour conséquence la perte de signaux. Par contre, les lois de commande ont une certaine robustesse intrinsèque et, par une série d'expériences sur banc test, ou en simulation, on peut déterminer η_{max}, le plus grand nombre de signaux perdus consécutivement, tel que la réaction du système n'affecte pas la sécurité du véhicule. L'environnement perturbant n'étant, par nature, pas déterministe, les propriétés à vérifier s'exprimeront de manière probabiliste. Il s'agit alors de prouver que la probabilité P de perdre plus de η_{max} signaux consécutifs est inférieure à un seuil fixé, pour l'instant, par des exigences internes aux entreprises du secteur (notons, que ce seuil est, comme pour l'avionique, de l'ordre de 10^{-9}).

En conclusion de cet exemple, on remarque que, d'emblée, les applications sont synchronisées sur des périodes liées à la production des signaux et à l'activation de la loi de commande ; l'accès au réseau de communication suit la même règle (protocole TDMA). Cette approche, entièrement guidée par le temps, offre des facilités pour vérifier de façon déterministe, en l'absence de perturbations, des propriétés sur la fraîcheur des données consommées et, donc d'assurer, sous ces mêmes hypothèses, la sûreté du système. C'est généralement l'approche préconisée pour tout système critique embarqué.

1.3 Qualité de service pour des réseaux supportant des applications multimédia temps réel

Ce dernier exemple d'applications qui nous permet d'illustrer le concept de « temps réel » se situe dans le contexte du contrôle des grands réseaux de communication à commutation de paquets (TCP/IP, ATM ou, plus généralement l'Internet). En effet, ces réseaux supportent de plus en plus des applications pour lesquelles des propriétés temporelles doivent être garanties. Parmi celles-ci, on peut noter les applications de télé-opération, la vidéoconférence, la transmission vidéo à la demande. Les propriétés à respecter par ces applications peuvent s'exprimer différemment : ce peut être une garantie de délai minimal entre la production d'une information par une station source jusqu'à la délivrance de cette information à son ou ses consommateurs et/ou la régularité de réception d'une suite de paquets comme dans le cas de transmission de flux vidéo. Cependant, pour vérifier le respect de ces propriétés et les assurer en ligne, les outils usuels de la communauté du temps réel ne sont pas directement applicables dans ce nouveau contexte. Entre autres, il n'est pas possible de caractériser le trafic circulant par exemple dans l'Internet, sous la forme d'arrivées périodiques ou sporadiques d'activités ; on doit, alors, avoir recours à des modèles d'arrivées aléatoires (loi de Poisson, loi exponentielle, etc.) ou à des modèles de rafales. Dans ce contexte, la communauté de la qualité de service (QdS) propose ce qui est usuellement désigné sous le terme d'architecture de qualité de service afin de répondre aux besoins en termes de temps réel de ces nouveaux types d'applications et ceci, sans modifier l'infrastructure initiale. La problématique générale se situe au niveau de l'adaptation des ressources disponibles du réseau (bande passante, tampons de stockage dans les différents nœuds traversés), pour assurer une qualité de service permettant le respect des propriétés attachées aux différents flux circulant sur ce réseau.

Les deux architectures de qualité de service actuellement déployées sont IntServ et DiffServ (pour plus de précision, se reporter aux sections 1.1 (ff Réseaux et télécommunications ff) et 1.4 (ff Systèmes multimédia ff) de l'encyclopédie). Dans chaque cas, il s'agit d'évaluer les ressources nécessaires et de mettre en place les mécanismes associés, lors d'une demande de connexion, à partir des caractéristiques annoncées du flux. IntServ, proposée en 1994 par l'IETF (IETF, 1994), repose sur le principe de la réservation de bande passante, par flux, sur le chemin de la source vers la destination. Cette approche, si elle assure une garantie de qualité de

service devient vite inapplicable sur des réseaux à large échelle. Pour pallier cet écueil, l'IETF a apporté en 1998 l'architecture DiffServ (IETF, 1998), qui consiste en l'agrégation des flux individuels dans un nombre limité de classes de service. Une demande de connexion, sous DiffServ, ne s'accompagne pas de réservation pour le flux. Par contre, dans un domaine DiffServ, sont mis en place des mécanismes de *contrôle d'admission*, généralement localisés à l'entrée des nœuds frontières du réseau, afin de refuser ou accepter l'entrée d'un flux, dans la classe de service demandée. Une caractéristique importante d'un flux, vis-à-vis de la qualité de service temps réel, est fournie par la spécification de la borne, sous la forme d'une fonction multilinéaire concave, de la quantité de travail cumulative induite par le flux. Dans chaque routeur sur le chemin d'un flux, est alors assuré un ensemble de mécanismes temps réel, dont le rôle est d'acheminer tout flux, accepté par le contrôle d'admission, de sa source à sa destination. Ces mécanismes sont représentés sur la figure 5. Les paquets sont orientés vers une des files d'attente en fonction de leurs caractéristiques par un service de classification. Les mécanismes de mise en forme et mise en vigueur garantissent qu'un flux reste conforme à la spécification qu'il avait lors de son admission, à savoir qu'il est toujours compris dans son enveloppe ; ils sont complétés par des algorithmes de gestion des files d'attente afin d'éviter les phénomènes de congestion. Ces deux types de mécanismes reposent sur des stratégies de rejet de paquets. Enfin, le dernier mécanisme ordonnance les demandes stockées dans les files d'attente. L'ensemble de ces mécanismes assure une qualité de service donnée à chacun des flux admis dans le domaine, en particulier, cet ensemble garantit qu'un paquet qui arrive à sa destination respecte l'échéance imposée à son temps de réponse.

Figure 5. Mécanismes de gestion de qualité de service

2 Implémentation d'une application temps réel

Dans un premier temps on va se limiter au cas d'une implémentation des actions d'une structure fonctionnelle, sur un support d'exécution monoprocesseur, équipé des interfaces d'entrées/sorties nécessaires à l'application. À première vue, il pourrait sembler que cette phase d'implémentation relève uniquement de techniques de développement, et que les décisions qui conditionnent le bon fonctionnement de l'application sont déjà prises. Or si cela est vrai sur le plan fonctionnel, rien n'est encore vraiment résolu sur le plan temporel. Le traitement des contraintes temporelles va être étroitement conditionné à ce moment par un choix majeur : celui de la réalisation du lien entre les actions et leurs événements activateurs. En effet, quels que soient les langages ou formalismes utilisés pour décrire l'application, quels que soient les outils adoptés pour l'implémentation, tous peuvent conduire, au niveau « machine », à deux techniques différentes de réalisation du mécanisme d'invocation des actions. Ces deux techniques relèvent de deux principes fondamentaux différents, l'approche « synchrone » et l'approche « asynchrone ».

2.1 La démarche synchrone

Cette démarche, qui a été à l'origine (principalement en France) de langages et environnements de développement pour les systèmes temps réel sera expliquée et détaillé dans le chapitre 5 de cette même section. Aussi seules quelques idées directrices sont données dans ce paragraphe. La démarche est fondée sur deux idées principales : l'hypothèse de réalisation des actions en temps nul et la possibilité de perception simultanée d'occurrences d'événements dans le système. De manière sous-jacente cela implique la non-préemption[3] des traitements, ce qui évite l'accès aux ressources critiques partagées en exclusion mutuelle. L'adoption de ces hypothèses n'est pas sans conséquence, notamment sur le plan temporel.

En effet, la perception qu'a le système de l'occurrence de tout événement est différée du temps d'exécution (réel) de l'action en cours (au moment de l'occurrence). Ce retard à la prise en compte peut affecter n'importe quel événement, quelle qu'en soit la gravité pour l'application. Il faudra donc vérifier sur l'architecture opérationnelle si l'hypothèse d'instantanéité des actions est valide, c'est-à-dire n'engendre pas le dépassement de certaines échéances temporelles quand on passe en situation réelle, c'est-à-dire avec les véritables durées d'exécution des actions (une action ne peut jamais être de durée nulle).

De plus, comme le système de contrôle ne constate l'arrivée de nouveaux événements qu'à la terminaison de chaque action, il reçoit à cet instant tous les événements en attente sans connaître leur ordre d'arrivée. Si cet ordre est important pour l'application, c'est-à-dire conditionne l'action du système de pilotage, quelle action alors entreprendre ? La technique adoptée dans l'approche synchrone pour traiter cette

[3] La préemption est l'action qui consiste à suspendre l'exécution d'une action au profit d'une autre jugée plus prioritaire (critère d'urgence par exemple).

circonstance consiste à considérer tous les événements observés au même instant comme étant survenus en même temps, c'est-à-dire comme simultanés. Cette simultanéité peut être alors perçue comme un nouvel événement pour lequel l'utilisateur peut définir une action spécifique à entreprendre en ce cas.

2.2 La démarche asynchrone et l'exécutif temps réel

La technique de mise en œuvre asynchrone consiste à observer en permanence les occurrences d'événement, y compris au cours de l'exécution des actions. Leurs dates d'arrivée et leurs ordres relatifs sont donc connus : la nature de l'action à entreprendre n'est pas ambiguë et son urgence peut être immédiatement prise en compte. Malheureusement, la préemption qui devient donc naturelle, outre son coût temporel, engendre de nouveaux problèmes.

A la différence de la démarche synchrone, il devient beaucoup plus difficile d'observer et donc de vérifier le bon comportement du système de contrôle. Il faut, comme en synchrone d'ailleurs, passer par une modélisation formelle du comportement de l'application pour pouvoir la vérifier, mais les modèles asynchrones qui permettent une telle preuve sont actuellement plus complexes qu'en démarche synchrone. Cela est en partie dû au fait que, macroscopiquement, des occurrences d'événement très voisines peuvent engendrer des réactions parfois bien différentes du même système ; certes, à l'examen très fin du déroulement de l'évolution de chaque action, ces différences de réaction sont parfaitement justifiées, et prédictibles, mais, à l'échelle temporelle du procédé, on prête parfois aux réactions asynchrones le caractère « d'indéterminisme » pour cette raison.

A l'instant d'occurrence de tout nouvel événement, on peut décider à qui allouer le processeur entre : 1° - la nouvelle action activée par cet événement, 2° - celle qui était déjà en exécution à ce même moment, 3° - une autre action en attente du processeur. De ce choix peut dépendre le respect des contraintes temporelles et les modalités de ce choix sont donc primordiales. Elles constituent ce qu'on appelle une « politique d'ordonnancement ». La mise en oeuvre à l'exécution de la plupart de ces politiques s'appuie sur un mécanisme de base bien connu, la priorité. L'emploi de ce mécanisme en ligne est alors très simple : pour suivre la politique d'ordonnancement adoptée, il suffit de choisir comme action à exécuter celle de plus forte priorité parmi toutes celles en concurrence. Les techniques d'ordonnancement ainsi que l'analyse d'ordonnançabilité d'une configuration constitue un domaine de recherche en soi. Ces aspects sont présentés et détaillés dans le chapitre 3 de cette même section.

Une fois la politique d'ordonnancement choisie, une fois son mécanisme de mise en œuvre défini, il s'agit de le réaliser. Pour cela il faut construire le dispositif qui centralisera la réception de tous les événements, qui suivra l'évolution de l'exécution des actions et qui, en conséquence, effectuera le choix à tout instant de l'action à exécuter. Ce centralisateur s'appelle « l'exécutif », et la composante de l'exécutif qui met en œuvre la politique d'ordonnancement s'appelle « l'ordonnanceur ». La structure d'un exécutif ainsi que des exemples de services offerts pour la coopération entre actions seront détaillés dans le chapitre suivant.

2.3 Architectures matérielles réparties

Lorsqu'on étend l'étude de la mise en œuvre d'une application temps réel aux implémentations sur une architecture matérielle constituée d'un réseau de calculateurs, les mêmes réflexions peuvent être menées qu'en implémentation centralisée, et le même choix entre synchrone et asynchrone peut être étudié et décidé, station par station. Mais il s'y rajoute maintenant un impératif majeur, celui de la coordination des actions qui sont exécutées en parallèle sur les différents calculateurs, chacun fonctionnant à sa vitesse propre. Il devient maintenant nécessaire de garantir la cohérence temporelle des actions s'exécutant sur des calculateurs différents. En effet, les observations des dynamiques du procédé effectuées par des actions réparties dans le réseau ne sont pas obligatoirement identiques car elles sont collectées ou reçues à des instants qui peuvent légèrement différer. Or, un faible décalage peut renverser un ordre d'apparition d'événements ; de même un délai de transmission dans l'actualisation d'une variable peut faire opérer deux actions à partir de deux valeurs différentes de cette variable. Éviter ce type d'incohérence implique fortement un nouvel acteur : le protocole de transmission des informations sur le réseau. Les caractéristiques de ce protocole conditionnent fortement la possibilité de pouvoir garantir le respect de propriétés du système temps réel. Ces caractéristiques sont de plusieurs ordres.

Tout d'abord la politique d'ordonnancement des messages. La détermination de cette politique est délicate à définir car s'entremêlent alors la priorité de l'action émettrice du message (sur son site), l'ordonnancement des messages dans les couches du protocole d'émission ainsi que dans celles du protocole de réception, la priorité de l'action réceptrice (sur le site de réception).

Une autre caractéristique est la politique d'accès au médium. Comme le médium est lui-même une ressource partagée entre les stations, il faut donc lui appliquer une politique d'allocation, c'est-à-dire un ordonnancement spécifique. Avec certaines politiques, la transmission d'un message pourra être garantie bornée (le réseau est alors qualifié de déterministe) et cette borne pourra même être calibrée en fonction des caractéristiques et des contraintes temporelles des données échangées. De telles possibilités sont, par exemple, offertes par certains réseaux de terrain dont le médium est rendu accessible aux stations connectées sur la base d'un cadencement cyclique (voir par exemple (Kopetz et Grünsteidl, 1994)). Des tranches temporelles (slot) sont ainsi allouées périodiquement à chaque station qui peut ainsi, soit transmettre les messages qu'elle a préalablement ordonnancés, soit transmettre des données désignées par le protocole lui-même. Cette technique, dite « dirigée par le temps » permet ainsi de garantir précisément le délai maximum d'échanges de données entre actions (hors défaillances non supportées bien sûr).

Enfin une dernière caractéristique est la qualification temporelle des données échangées. En accompagnant les données transmises d'indicateurs traduisant leur « fraîcheur » à la production comme à la consommation, on peut offrir aux actions réceptrices d'une donnée la possibilité d'adapter en conséquence le traitement à entreprendre sur cette donnée. Ces indicateurs sont mis à jour sur la base d'une datation globale (ou d'une synchronisation commune à toutes les stations) qui est offerte par le réseau. C'est l'ouverture vers la tolérance aux fautes temporelles, les origines des fautes pouvant être multiples : défaillances matérielles, surcharge sporadique et imprévue....

Le lecteur intéressé pourra se reporter à Thomesse (1999) pour des informations plus détaillées sur les réseaux utilisables pour les applications temps réel.

2.4 La démarche TT (*Time Triggered*)

Dans le cadre de l'approche asynchrone, la démarche qui vient d'être présentée en 3.2. est aussi parfois dénommée « *Event Driven* » ou « *Event Triggered* » car le comportement du système de contrôle est déterminé par l'occurrence des événements (au sens large du terme). Ce qualificatif illustre bien le comportement du logiciel applicatif de contrôle et de l'exécutif. En effet, les appels à l'exécutif sont la conséquence, par exemple :

- des occurrences d'événements issus du procédé ; concrètement, l'appel à l'exécutif est effectué par la procédure de réception et de traitement de l'interruption matérielle associée à ces événements ;
- du temps ; concrètement, l'appel à l'exécutif est provoqué par l'interruption régulièrement engendrée par une horloge temps réel équipant le calculateur ;
- des tâches elles-mêmes ; c'est le cas lorsqu'une tâche requiert des services offerts par l'exécutif (signalisation d'une occurrence d'événement, mise en attente d'une occurrence d'évènement, transmission d'un message dans une boîte aux lettres, etc.).

Toujours dans le cadre de l'approche asynchrone (qui tient compte des temps d'exécution des traitements), on peut concevoir le système de contrôle avec un seul événement externe, celui provenant d'une horloge périodique. Tous les traitements deviennent alors des activités périodiques dont les instants d'activation peuvent être planifiés afin de respecter les contraintes temporelles. Il n'y a plus d'ordonnancement en ligne comme dans le cas précédent mais un ordonnancement précalculé (et donc vérifié) hors ligne. Les actions procèdent par scrutation pour connaître l'état des entrées du procédé ; ainsi l'état d'un capteur dans les deux premiers exemples donnés précédemment, sera examiné régulièrement avant d'effectuer un éventuel traitement. Dans de tels systèmes la préemption n'a plus de sens puisque tous les traitement sont organisés hors ligne. De tels systèmes sont qualifiés de « *Time Driven* » ou « *Time Triggered* », l'exemple par excellence étant le système MARS (Kopetz et al., 1989). On notera qu'ils présentent une grande rigidité conceptuelle par rapport à l'approche *Event Driven*, mais que par contre ils sont très prédictibles, bien sûr sous le respect des hypothèses utilisées pour la conception. C'est la démarche suivie pour l'exemple de l'application automobile décrite au paragraphe 1.2.

3 Le besoin de vérification des systèmes temps réel

Quelle que soit l'approche suivie pour l'implémentation d'une application temps réel, son développement doit inclure une phase essentielle de vérification. Ainsi, développer des systèmes temps réel inclut de vérifier que les propriétés qui sont imposées, par leurs missions et par les caractéristiques des systèmes physiques avec lesquels ils interagissent, seront respectées durant la vie de ce système. Les propriétés particulières qui nous intéressent dans ce contexte sont celles qui s'expriment sous la forme de contraintes de temps et de performance. Deux voies

sont exploitables pour garantir ces propriétés. La première consiste à les assurer par conception du système, par exemple, en construisant l'ordonnancement d'activités sous ces contraintes. La deuxième, nécessaire dès que le système devient complexe ou dès qu'il est conçu dans une démarche impliquant plusieurs acteurs, met en œuvre des techniques de vérification de ce système à toute étape de son développement ; il s'agit, dans les phases de spécification et conception de ce système, d'en faire un modèle, c'est-à-dire une abstraction, sur lequel des techniques d'analyse peuvent être appliquées. Ces techniques relèvent de méthodes mathématiques, reposant sur des calculs exacts ou approchés, ou sur la simulation / exécution de modèles. Une dernière voie pour vérifier qu'un système temps réel respecte les propriétés qui lui sont imposées est le test de ce système, ou d'une de ses parties, une fois qu'il est réalisé. Les techniques de test s'appuient soit sur des modèles construits lors du développement du système, soit sur des scénarios issus de mesures réelles ou statistiques.

Les chapitres suivants de cette section vont permettre de détailler un ensemble de points présentés dans ce chapitre. Ainsi, les chapitres 2 et 3 se placent dans le cadre de la démarche asynchrone et vont présenter, pour le premier la structure et les services d'un noyau d'exécutif temps réel, et pour le second les principales techniques d'ordonnancement et d'analyse de l'ordonnançabilité d'une configuration d'actions (appelées tâches par la suite). Le chapitre 4, quant à lui, présentera les résultats essentiels concernant les techniques de vérification adaptées aux systèmes temps réel. Enfin, le chapitre 5 présentera l'approche synchrone pour le développement des systèmes temps réel.

Bibliographie

Calvez J.P. (1990), *Spécification et conception des systèmes, une méthodologie*, Masson.

Comete (1998), *Co-design : conception conjointe logiciel – matériel, C.T.I, COMETE*, ouvrage collectif, Collection Technique et Scientifique des Télécommunications, Eyrolles.

IEF (1994), RFC1633, *Integrated Services in the Internet Architecture: an overview*, R. Braden, D. Clark, S. Schenker, IETF, Juillet.

IETF (1998), RFC2475, *An Architecture for Differentiated Service*, S. Blake et al., IETF, Décembre.

Kopetz H., Damm A., Koza C., Mulazzani M., Senft C., Zainlinger R. (1989), The MARS approach, *IEEE Micro*, volume 9, number 1, p. 25-40.

Kopetz H., Grünsteidl G. (1994), TTP - A Protocol for Fault-Tolerant Real-Time Systems, *IEEE Computer*, volume 27, number 1, p. 14-23.

Thomesse J.P. (1999), *Les réseaux temps réel industriels*, Editeur Innovation, Paris.

Wilwert C., Navet N., Song Y.Q., Simonot-Lion F. (2004), Design of automotive x-by-wire system, *The Industrial Communication Technology Handbook*, Editeur R. Zurawski, Décembre.

Figures 1 et 2, dessinées avec Micrograph Designer 7

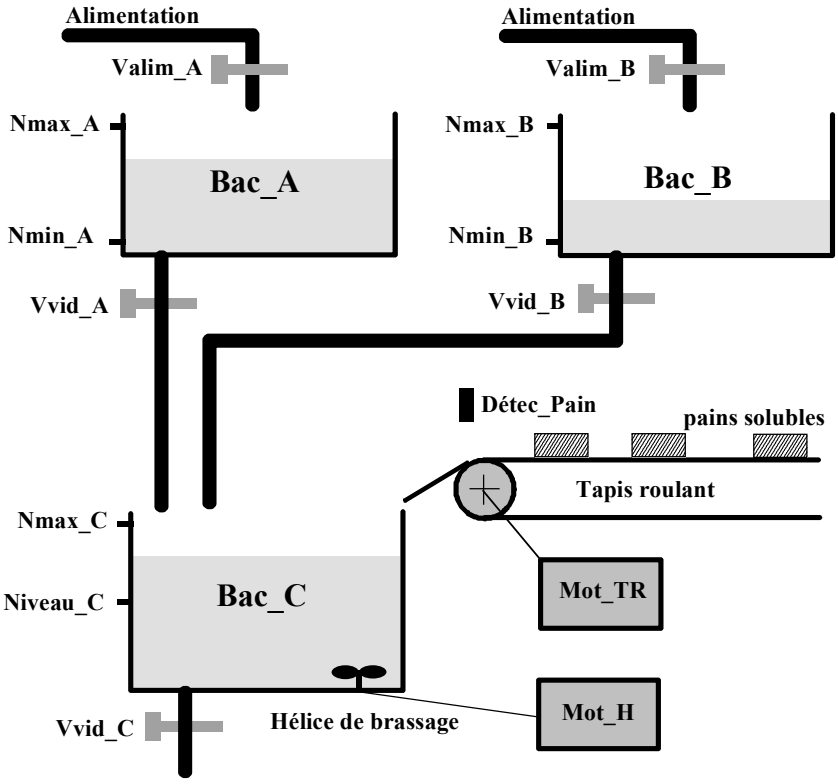

Figure 1. Synoptique du procédé de fabrication

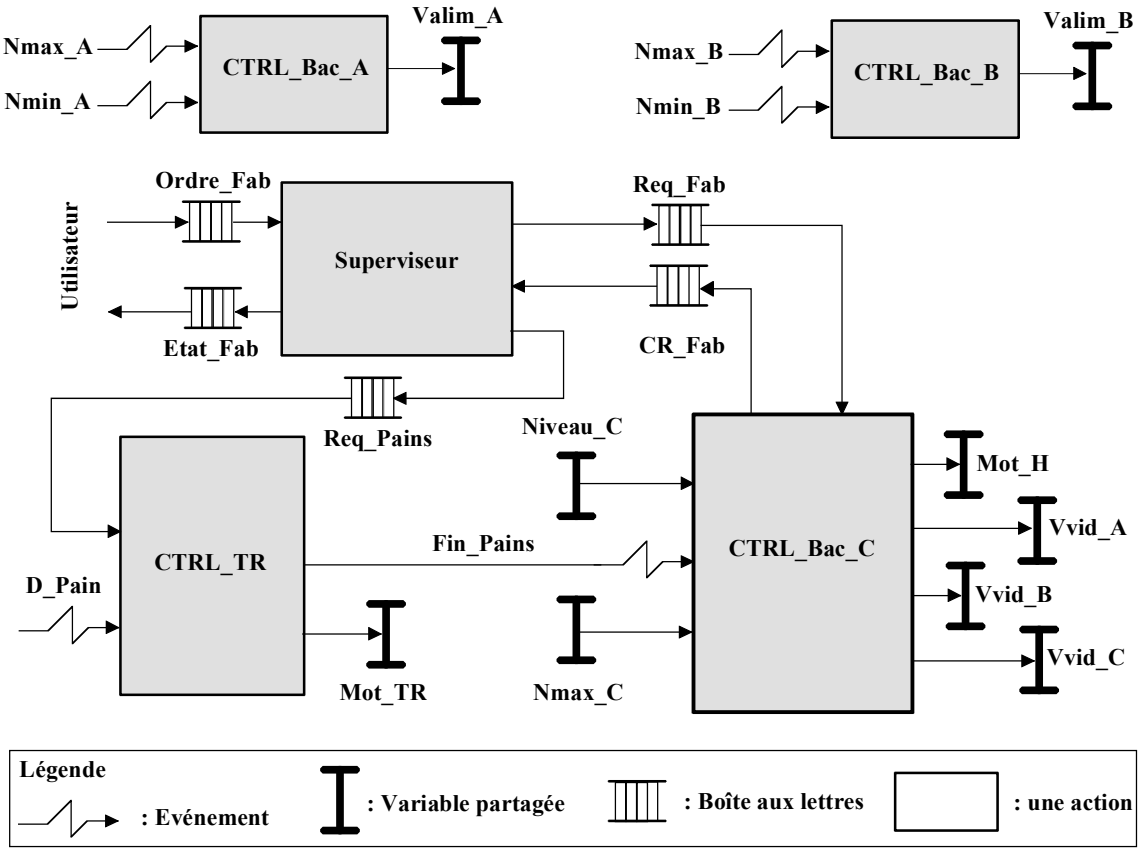

Figure 2. Un exemple d'architecture fonctionnelle, solution au problème

Figures 3, 4 et 5 dessinées avec Visio2000.

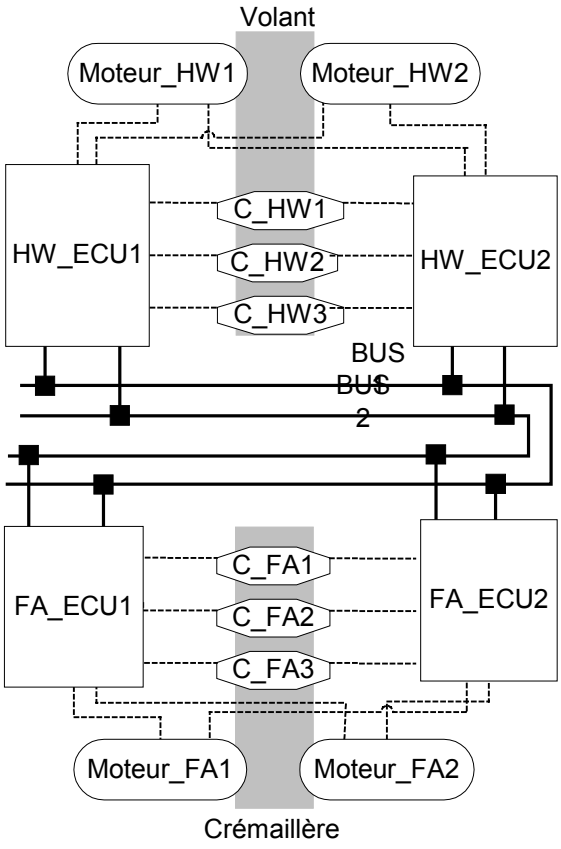

Figure 3. Architecture informatique support d'un système "Steer-by-Wire"

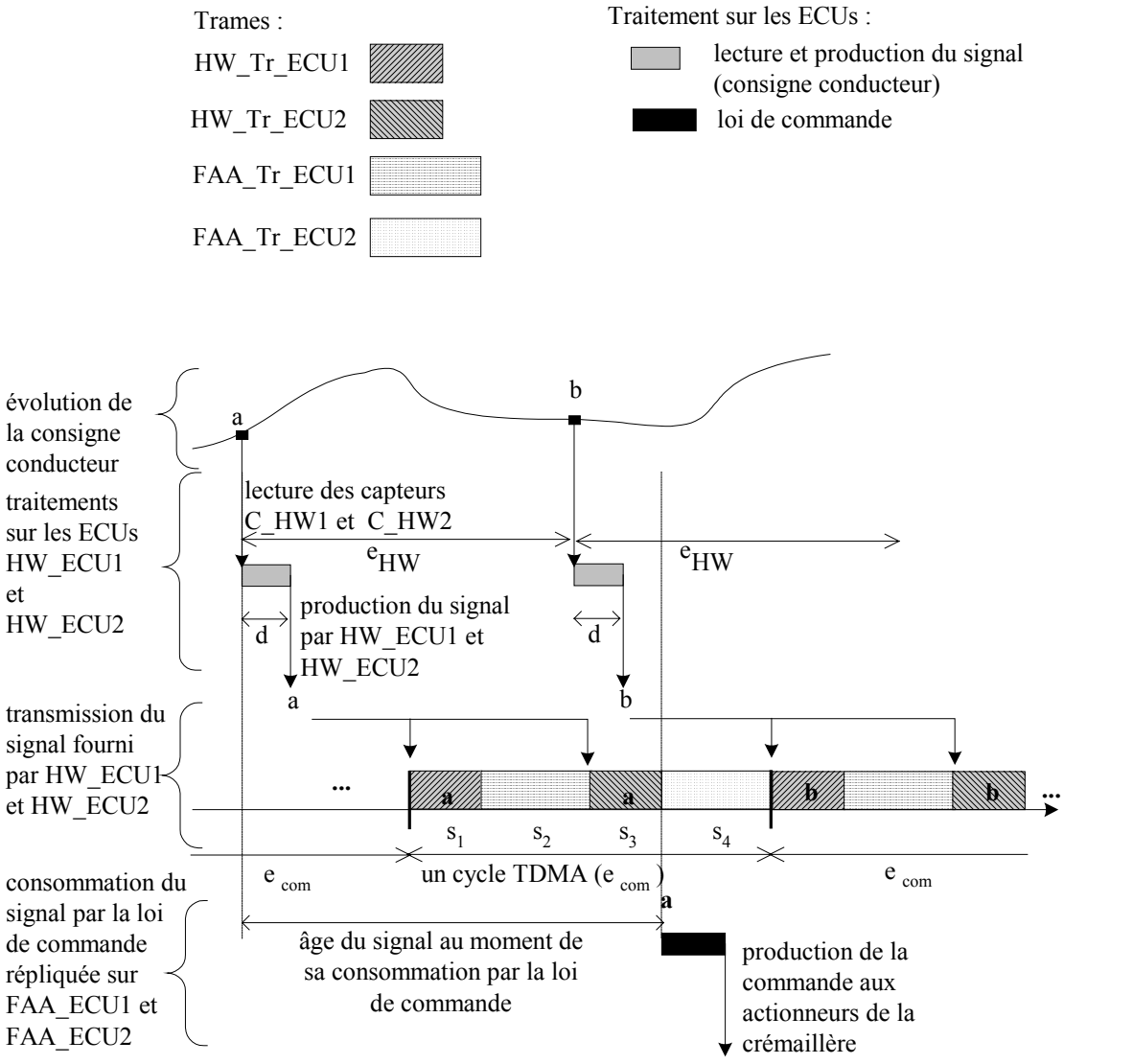

Figure 4. Diagramme temporel des différentes activités.

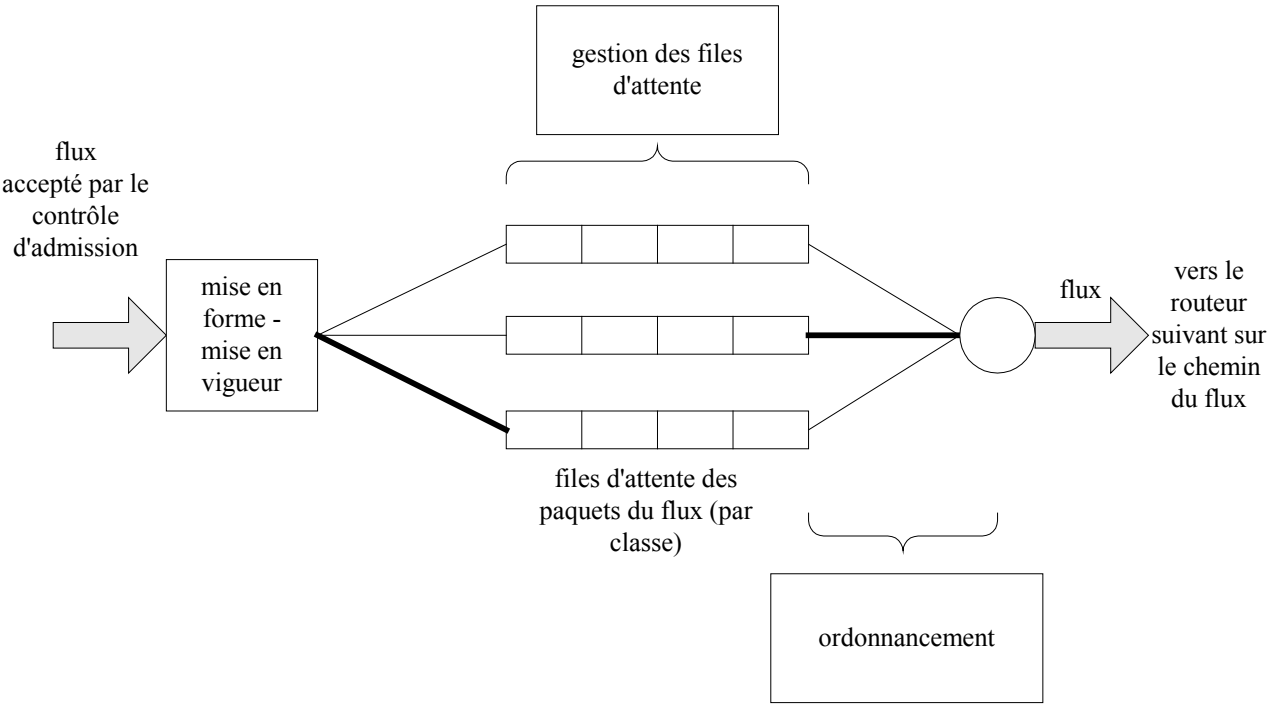

Figure 5. Mécanismes de gestion de Qualité de Service