# Evaluation de Micro-Noyaux Temps Réel pour les Applications d'Informatique Industrielle d'EDF

E. Becquet, P. Decogné, E. Gressier-Soudan, L. Bacon

becquet@cnam.fr, decogne@cnam.fr, laurent.bacon@edf.fr, gressier@cnam.fr

Cnam, Laboratoire Cedric, 292 rue St Martin 75141 Paris Cedex 03
EDF-DER, 6 Quai Watier, 78401 Chatou Cedex

**Résumé :**

EDF évalue des micro-noyaux possibles pour la rénovation de certaines applications d'informatique industrielle. Les besoins des équipes d'intégrations font apparaître une nécessité d'utiliser des outils du commerce sur étagère dans la mise en œuvre des futures applications. De plus les enquêtes ont permis aussi de mettre en évidence les besoins en terme de fonctionnalités, de performances et d'outils d'ingénierie. Dans cette optique, quatre systèmes d'exploitation temps réel du commerce : ChorusOS, LynxOS, pSOS+ et VxWorks sont à étudier. Ce papier évalue les trois derniers systèmes d'exploitation suivant les points de vue mis en évidence par les enquêtes. ChorusOS est en cours d'évaluation fonctionnelle, l'évaluation opérationnelle devant être faite très prochainement.

**Mots-Clefs :** système d'exploitation temps réel, évaluation, performances, ingénierie, LynxOS, pSOS+, VxWorks.

## 1 Introduction

Notre étude s'inscrit dans une démarche globale concernant la mise en œuvre d'application d'informatique industrielle distribuées orientées objet temps réel à partir d'une approche orientée "composants du marché sur étagères" (encore désignée par l'acronyme COTS pour Commercial Off-The-Shelf) afin de réduire les coûts de développement et de maintenance [BAC00]. La construction de telles architectures concernent les bus logiciels à objets, les réseaux, les micro-noyaux temps réel et leur coopération. Cet article décrit l'étude faite sur les micro-noyaux temps réel. Parmi les propriétés assurées par les micro-noyaux candidats, le déterminisme temporel et la performance occupent un rôle essentiel. Notre étude concerne les systèmes d'exploitation temps réel, plus précisément, les fonctionnalités et les propriétés qui seront utiles à la conception des applications d'informatique industrielle réparties d'EDF [BAC99a].
Notre travail vise à faire émerger un certain nombre de points de repères à propos des technologies candidates pour de futurs projets d'expérimentation. L'étude sur les systèmes d'exploitation temps réel se découpe en plusieurs étapes. La première étape correspond à une analyse des besoins « utilisateurs » en matière de systèmes d'exploitation temps réel pour les applications d'Informatique Industrielle. Cette première étape s'est caractérisée par un questionnaire [BAC98a] et plusieurs enquêtes [BAC99b]. La seconde étape est constituée par une étude de l'architecture des systèmes d'exploitation et se termine par une mise en œuvre sur une plate-forme d'expérimentation. Cette plate-forme reflète les environnements matériels standards des solutions du commerce pour des applications d'informatique industrielle. Cette étude sera poursuivie par l'étude des solutions de répartition temps réel [BEC00].

Les systèmes d'exploitation du commerce évalués ont été sélectionnés en raison de leur représentativité dans le domaine du temps réel pour l'industrie, pour leur architecture, pour les propriétés qu'ils offrent [BAC98b]. Dans l'obtention des résultats, nous nous sommes autant attachés aux performances qu'aux fonctionnalités offertes pour supporter des applications temps réel, en particulier les aspects liés à l'ingénierie des applications temps réel, et à l'administration des environnements d'exécution.

Cet article est organisé comme suit : nous commencerons par caractériser le domaine applicatif visé dans la partie 2 et nous en déduirons les objectifs de l'étude que nous nous sommes fixés dans la partie 3. Nous verrons l'évaluation des systèmes temps réel, au niveau performances dans la partie 4 et au niveau de l'ingénierie dans la partie 5. Enfin nous conclurons et proposerons des perspectives à notre travail en partie 6.

## 2 Caractérisation du domaine applicatif

La première étape de l'étude a consisté à recueillir les contraintes qui portent sur les systèmes d'exploitation temps réel au moyen d'entretiens. Un questionnaire [BAC98a] a eu pour objectif de dégager les besoins en performances et certaines caractéristiques temporelles d'applications d'informatique industrielle, par exemple : temps de traitement, temps d'acheminement des informations, temps de réponse, débit et longueur des données, nature des informations échangées et leurs échéances ou périodes... Ce

questionnaire a permis par ailleurs de caractériser les fonctionnalités des systèmes d'exploitation temps réel attendues par ces applications. Il aborde aussi des aspects plus généraux : la facilité d'administration, l'extensibilité du système (composants, serveurs, modules dynamiquement chargeables), les processeurs et bus supportés, la capacité de reprise sur panne, les protocoles de communication, l'existence d'une couche de type bus logiciel...

Une synthèse globale des entretiens [BAC99b] indique, sous forme de recommandations, les points clefs à évaluer dans la poursuite de l'étude : sur les systèmes d'exploitation temps réel, et sur les plates-formes d'exécution réparties à objets. Par ailleurs, cette synthèse résume les points caractéristiques de chaque enquête et permet l'élaboration d'un profil type d'application industrielle EDF [BAC99c]. Les applications d'informatique industrielle en centrale se répartissent sur plusieurs niveaux. Le niveau 0 est chargé de l'acquisition physique de données (majorité des échanges) et des commandes aux équipements (capteurs et actionneurs). Le niveau 1 effectue un traitement temps réel des données acquises et produit des données de synthèse. Le niveau 2 est chargé de traitements de supervision, de l'archivage, du poste opérateur et de l'interface avec les autres fonctions du contrôle de production globales à la centrale. Les architectures matérielle et logicielle types d'une plate-forme d'informatique industrielle sont données dans la figure 1. La figure 2 schématise la circulation des données entre calculateurs de niveaux 1 et 2.

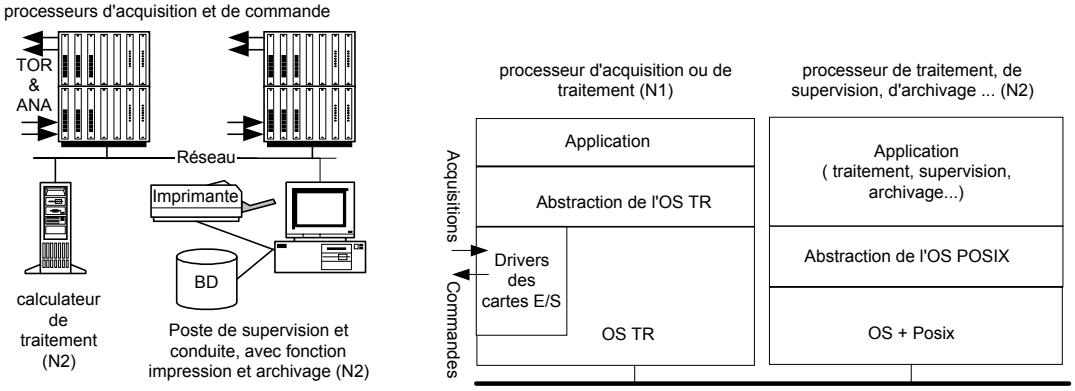


**Figure 1. Architectures matérielle et logicielle types**

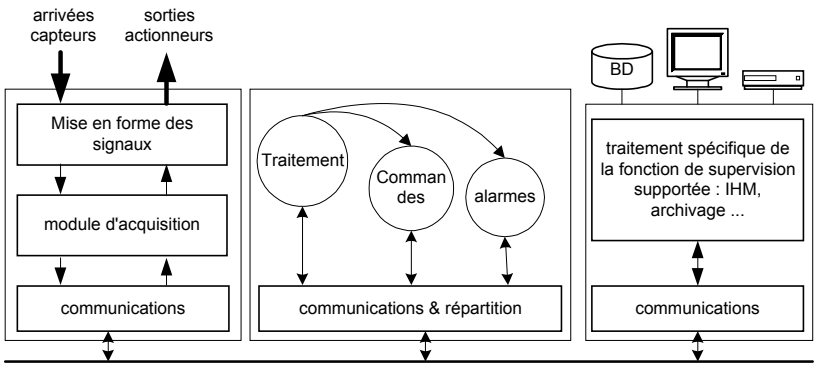


**Figure 2. Flots de données du contrôle-commande**

Les contraintes temps réel se situent aux Niveaux 1 et 2 précédents et concernent l'élaboration de quelques milliers d'objets par seconde. Les contraintes de temps de bout en bout sur les commandes et les données procédé sont de 500 ms voire 1s parfois. Par contre il faut être capable de garantir des débits de niveau applicatif sur les messages pour le flot d'acquisition. Les pertes de données ne sont pas tolérées.

Les abstractions des systèmes d'exploitation temps réel majoritairement utilisées sont : des processus, des activités (threads), des tampons mémoire, de la mémoire partagée, des mécanismes de datation et de temporisation, des sémaphores, des files de messages et des évènements ou des signaux, des boites aux lettres (sockets) au-dessus de TCP/IP et d'Ethernet. Ces mécanismes participent à la mise en œuvre d'un contrôle de flux de niveau applicatif qui évite toute saturation de l'infrastructure impliquée dans le traitement des flots de données. Les contraintes temporelles sont donc plutôt faibles et laissent une réelle ouverture quant au choix du système temps réel.

Les enquêtes ont fait émerger d'autres caractéristiques importantes qu'il est nécessaire d'évaluer avant de choisir un système d'exploitation temps réel. On a retrouvé bien sûr la contrainte de pérennité du produit, du fournisseur, et la qualité du support après vente. La qualité de l'environnement de développement est revenue comme une contrainte primordiale. En effet, cet aspect influe directement sur les temps de développement et d'intégration. Par conséquent, nous l'avons incluse dans notre évaluation

## 3 Objectifs de l'Etude

Nous étudions quatre systèmes d'exploitation temps réel : ChorusOS [URL01] (Sun Microsystems), LynxOS [URL02] (Lynx Real Time Systems Europe), pSOS+ [URL03] (Integrated Systems Inc) et VxWorks [URL04] (Wind River). La plate-forme d'expérimentation est construite à partir des éléments suivants [BAC98a]:

- des processeurs Intel en réseau Ethernet, pour simuler les processeurs du niveau supervision (niveau 2 de la hiérarchie du contrôle-commande EDF,
- des processeurs PowerPC sur bus VME en réseau Ethernet, pour simuler les processeurs du niveau traitement des acquisitions et des commandes (niveau 1 de la hiérarchie du contrôle-commande EDF),
- d'un processeur PowerPC sur une carte MBX, pour simuler les capteurs et actionneurs intelligents.

La grille de tests d'évaluation en fonction des plates-formes matérielles sélectionnées est la suivante :

| OS | Intel - Ethernet | PPC - VME | PPC - MBX |
|---|---|---|---|
| ChorusOS | X | X | |
| LynxOS | X | X | |
| pSOS+ | | X | X |

| | | | |
|---|---|---|---|
| VxWorks | | X | X |

La suite de tests comporte trois types d'évaluations : des tests de performance avec un benchmark générique indépendant des systèmes, des tests d'ingénierie en environnement croisé avec un hôte sous WindowsNT, et des tests d'administration.

Nous avons choisi une approche boite noire qui ne nécessite pas d'intrusion dans le système temps réel évalué et qui donc permet une économie de temps et de moyens.

Initialement, peu de temps étaient prévu pour la mise en œuvre des tests et l'obtention des résultats, une durée totale de cinq mois se répartissant en une charge de huit hommes mois. Dans la réalité, ce délai n'a permis d'évaluer que trois systèmes d'exploitation : pSOS+, VxWorks, LynxOS dans l'ordre chronologique d'évaluation. Nous expliquons les raisons de cet accroissement de délai un peu plus loin.

L'approche boîte noire permet également d'avoir une évaluation plus uniforme pour tous les systèmes. Par ailleurs, pour les tests de performance, la granularité de temps utilisée (quelques microsecondes) a été suffisante étant donnés les résultats obtenus vis à vis des contraintes applicatives portées à notre connaissance.

La suite de tests a été développée spécifiquement. Nous n'avons pas utilisé une suite de tests déjà écrite telle que le Rhealstone [KAR89,KAS90] car au démarrage de l'étude, nous n'en disposions pas. Quand nous l'avons récupérée, le premier système d'exploitation n'était plus évaluable, sa licence d'évaluation avait expiré.

## 4 Evaluation de Performance

### 4.1 Objectifs

La mesure de performances permet de vérifier que les systèmes faisant partie de l'étude répondent bien aux exigences requises par les applications EDF. Les contraintes temps réel sont assez lâches, de l'ordre de quelques centaines de milli-secondes, et devraient donc être satisfaites.

Cette mesure doit également nous permettre de vérifier le comportement des différentes abstractions des systèmes temps réel sous test. Si certaines abstractions sont communes aux quatre systèmes que nous devons étudié, des particularités émergent. L'étude de performance nous fournit des premiers éléments de distinction fonctionnelle entre les systèmes d'exploitation temps réel.

La comparaison directe des performances obtenues, n'est en fait qu'un but secondaire.

### 4.2 Technique d'évaluation

Nous mesurons de la même manière tous les appels systèmes des différents OS. Dans les systèmes temps réel, les APIs se ressemblent beaucoup et on y retrouve souvent les mêmes fonctions, au nom et ordre des paramètres près. C'est en partant de ce constat que nous avons développé notre programme de tests.

Tous les bancs de tests sont écrits à l'aide de macros dont les définitions vont varier suivant le système cible. Les sources du banc de tests vont donc être divisés en deux groupes, suivant qu'ils dépendent ou non du système cible. Le « travail » de portage du

banc de tests sur un nouveau système revient alors à redéfinir l'ensemble des macros utilisées. Ainsi, le portage sur le deuxième système nous a pris une journée, en comptant l'exécution des tests eux-mêmes et la récupération des résultats.

La solution retenue pour avoir une échelle de temps assez fine pour nos mesures est celle déjà utilisée dans [MCV96], nous allons exécuter l'appel système à mesurer un certain nombre de fois, suffisamment grand pour obtenir un temps moyen crédible. De plus, nous effectuerons cette mesure plusieurs fois afin de pouvoir fournir un minimum, un maximum, une moyenne et un écart-type pour chaque temps mesuré.

Les tests réseaux sont un cas à part puisqu'ils utilisent l'interface des sockets, commune à quasiment tous les systèmes d'exploitation et disponibles sur les quatre systèmes de notre étude. Le même code, à quelques petites adaptations près, peut donc être utilisé. Dans le même ordre d'idée, l'utilisation d'APIs POSIX lorsqu'elles existent permet de les réutiliser pour un autre système les supportant également. Ainsi, l'API POSIX pour le temps est disponible sous LynxOS et sous ChorusOS, nous avons donc pu faire l'économie d'un portage de cette API.

### 4.3 Mise en œuvre

Nombre de tests mis en œuvre se résument à exécuter séquentiellement N fois l'appel système en cours d'évaluation, puis à diviser le temps ainsi obtenu pour avoir un temps moyen. Mais certains appels systèmes ne peuvent pas être évalués de cette manière et exigent un traitement spécifique.

Prenons pour exemple le calcul du temps du changement de contexte. Comme pour tous les temps, il faut trouver un moyen d'exécuter N changements de contexte sans autres appels systèmes parasites, pour pouvoir diviser ensuite le temps recueilli et en déduire un temps moyen. Pour ce faire, on utilise P tâches de même priorité qui vont demander un ré-ordonnancement dès qu'elles ont la main, via une macro SWITCH_OP() qui sera redéfinie dans chaque système. Dans chaque système, on positionne l'ordonnancement pour que les tâches de même priorité soient gérées en tourniquet. On compte les changements de contexte grâce à un compteur local, partagé par les tâches. Les opérations sur ce compteur (affectation et test) ont été évaluées négligeables en termes de temps d'exécution devant celui du changement de contexte.

Nous avons préféré cette méthode à celle décrite dans [MCV96], basée sur le même principe mais qui faisait appel aux tubes nommés[1] pour provoquer les changements de contexte. Nous avons remplacé ceux-ci par un appel système moins invasif et qui perturbe moins les résultats. Dans certains systèmes, une fonction est même préconisée pour effectuer un changement de contexte.

### 4.4 Résultats

Les résultats présentés sont ceux obtenus pour les systèmes LynxOS 3.0.1 (Lynx Real-Time Systems), pSOS+ 2.1.1 (Integrated Systems Inc.) et VxWorks 5.3.1 (Wind River). Les temps retenus sont un sous-ensemble de ceux obtenus puisque le banc de tests

[1] Pipes à la UNIX.

complet fourni une centaine de temps différents par OS, suivant les abstractions supportées.

Enfin, les temps présentés correspondent à la carte MBX (PPC 821 à 40 Mhz), au Pentium II (à 350 Mhz) et à la carte PowerPC MVME 2700 (PPC 750 à 266 Mhz) sachant que cette dernière est la seule plate-forme commune aux quatre systèmes de l'étude. Nous exposons les temps obtenus pour les sémaphores sous pSOS+, ceux pour les files d'attente pour LynxOS et les résultats réseau pour VxWorks. Les résultats complets pour chaque système sont disponibles dans [BAC99d], [BAC99e] et [BAC99f], respectivement pour LynxOS, pSOS+ et VxWorks.

L'histogramme en figure 3 synthétise les résultats obtenus avec pSOS+ à propos des sémaphores sur une carte MBX. Les résultats sont présentés normalisés. Les tests correspondent aux opérations classiques sur les sémaphores : création, prise de sémaphore (P), libération de sémaphore (V), libération entrainant le réveil d'une tâche bloquée (V ready), prise entraînant un blocage (P wait) et libération entraînant une préemption (V preempt). Le test « P/V Preempt » a servi à déduire « V Preempt ».

La figure 4 expose les temps (normalisés) obtenus sous LynxOS sur un Pentium II, pour les files de messages. Les temps mesurés ici sont la création, la destruction, l'envoi et la réception avec les variantes quand ces opérations entraînent le réveil d'une tâche (ready), la préemption de la tâche appelante (preempt) ou son blocage (wait). Ces tests ont été de plus effectués avec une taille de message T, puis T*4, puis T*16, ceci afin de noter une influence éventuelle de la taille des messages sur les résultats.

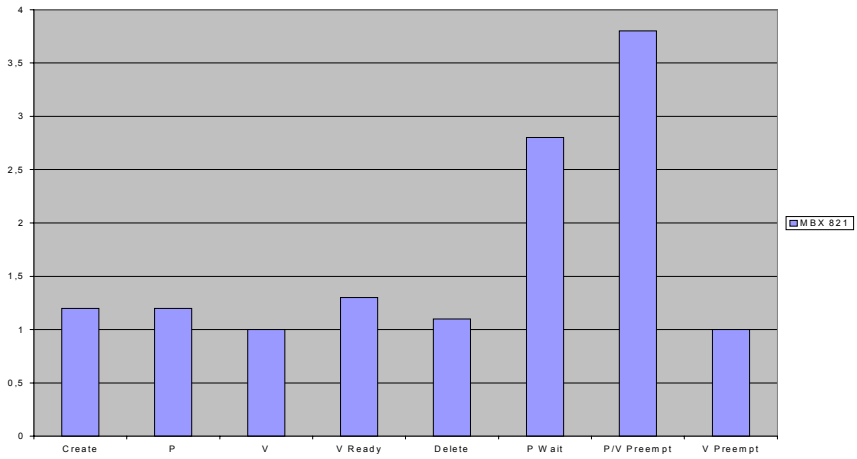


**Figure 3. Résultats des tests sur les sémaphores sous pSOS+**

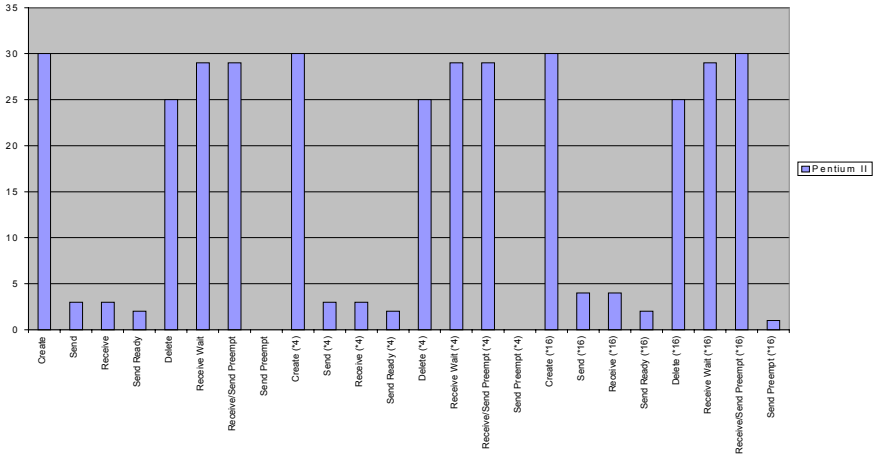


**Figure 4. Résultats des tests sur les files de messages sous LynxOS**

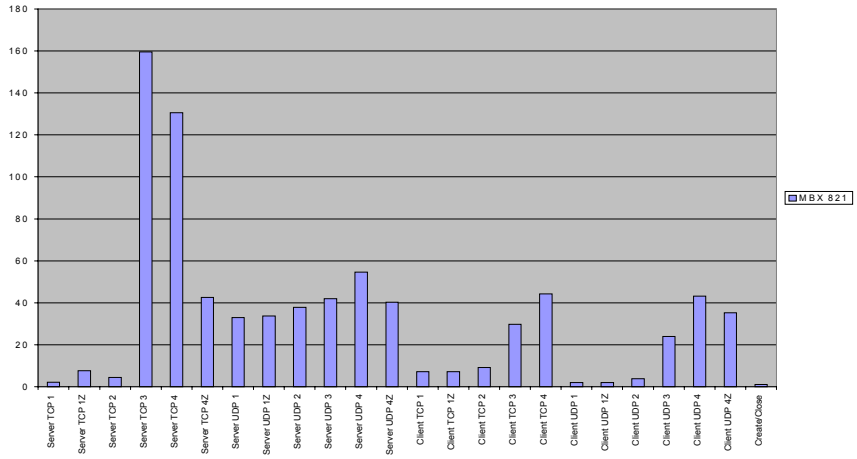


**Figure 5. Résultats des tests réseau sous VxWorks**

Les tests réseaux ont porté sur 4 tailles différentes de messages choisies de façon à mettre en évidence les problèmes liés à la fragmentation. De plus, la machine testée a joué successivement le rôle du serveur et celui du client, ceci en UDP et en TCP. Enfin nous avons testé le temps de création/destruction d'une socket. Nous avons aussi testé l'implantation « zero-copy » disponible sous VxWorks et pSOS+ (Tests suffixés par Z). Les résultats pour VxWorks sont présentés sur la figure 5. Les résultats de ces tests sont concluants en ce qui concerne les contraintes exprimées par les développeurs contactés lors des enquêtes. On remarque particulièrement les gains appréciables en ce qui concerne l'implantation « zero-copy » des protocoles TCP-UDP/IP.

### 4.5 Extensibilité du Benchmark

L'avantage de cette méthode de programmation pour l'écriture du banc de tests (utilisation massive de macros), c'est d'une part le portage simplifié sur un nouveau système temps réel, d'autre part l'extension des tests facilitée. En effet, pour appréhender la vitesse réelle du système, en utilisation plus « réaliste », on peut écrire des scenarii plus ou moins crédibles, à partie des macros du banc de tests. Une seule

écriture suffit alors, il n'y aura plus qu'à recompiler sous tous les systèmes déjà portés pour avoir les résultats des tests.

Evidemment, l'extensibilité se situe aussi aux niveau de l'ajout d'un nouveau système, qui se fait assez rapidement. D'autant plus rapidement que le nouveau système intègre des API POSIX. En effet, il est préconisé lors des portages d'utiliser une API POSIX pour une abstraction si elle est disponible. Cela permettra de réutiliser la définition des macros lors d'un portage d'un nouveau système qui offre l'API POSIX. Ainsi, l'utilisation de l'API POSIX pour la gestion du temps dans LynxOS et VxWorks a permis de n'écrire qu'une fois le portage de cette API.

### 4.6 Bilan et Perspectives

Du point de vue des performances, les systèmes d'exploitation évalués tiennent les contraintes des applications d'EDF visées. En construisant une chaîne de traitement du capteur au niveau 0 jusqu'au poste opérateur en niveau 2, nous observons que les temps passés en gestion système (envois/réceptions en files, commutations de threads, communications réseaux avec passage par les sockets, utilisation de mémoire partagée et de sémaphores) quelle que soit la plate-forme ne sont pas prohibitifs. Ceci représente moins de 10% de la contrainte énoncée plus haut de 500ms de bout en bout.

Notre étude mériterait d'être poursuivie pour une évaluation plus fine des plates-formes ci-dessus. En effet, nous n'avons testé que l'API dans un mode idéal, c'est à dire sans aucune perturbation due à une application qui s'exécute en concurrence. Il faudrait donc faire des mesures avec des machines chargées par une application perturbatrice (avec des entrées sorties capteurs, actionneurs ou réseau). Dans un second temps, il faudrait évaluer les systèmes d'exploitation avec des applications possédant les profils EDF, par exemple avec différents régimes : régime permanent avec ou sans occurrence d'avalanches. Ces deux aspects ne seront pas considérés directement pour l'évaluation des OS eux-mêmes. Ce type d'approche sera plutôt envisagé lors de l'étude de bus logiciels au-dessus d'exécutifs temps réel.

## 5 Ingénierie

### 5.1 Objectifs

Les enquêtes ont fait apparaître la nécessité d'évaluer les fonctions d'ingénierie, pas uniquement les performances des systèmes d'exploitation temps réel ci-dessus. En effet, les propriétés de la fonction ingénierie peuvent être un atout dans la mise en œuvre d'un projet temps réel. Les problèmes de coûts et de temps de développement sont importants et doivent être évalués. Dans notre cas, nous avons vu que les contraintes temps réel liées aux applications industrielles d'EDF ne sont pas trop dures. Comme les systèmes évalués conviennent aux besoins applicatifs, les fonctions offertes par la partie ingénierie peuvent donc avoir une grande influence dans le choix d'une plate-forme.

L'étude des fonctions d'ingénierie va permettre d'évaluer la qualité des outils fournis aux développeurs pour créer leurs programmes et les mettre au point. Il faut donc tester : l'environnement de développement, la gestion de sources, le déverminage, la prise de traces et leur affichage, l'aide à l'optimisation des applications et du noyau, la documentation et le support technique. Dans la partie ingénierie, nous avons incorporé

tout ce qui concerne aussi l'administration ; plusieurs aspects y sont liés: l'installation, le paramétrage du noyau des cibles (configurabilité et extensibilité en terme de fonctions-modules), et la gestion de l'environnement réseau et système (maintenabilité et extensibilité en terme de machines).

Il n'est pas possible de mettre au point une méthode systématique d'évaluation de cet aspect des systèmes d'exploitation. En effet, chaque système instancie une culture spécifique pour le développement et l'administration. Nous avons adopté une démarche qui a essayé d'être pragmatique. Dans un premier temps, nous avons, à l'utilisation, évalué la richesse de l'environnement en termes d'outils disponibles et la richesse fonctionnelle de ces outils. Par exemple, existe-t-il un outil de trace et si oui peut-on exporter les données vers d'autres formats pour exploiter les résultats facilement ? Dans un deuxième temps, nous avons testé la « facilité » d'utilisation.

### 5.2 Résultats

Les trois systèmes évalués pour l'instant n'appartiennent pas à la même catégorie de produit. En effet, si pSOS+ et VxWorks sont comparables, puisque ce sont des micro-noyaux et que le développement se fait en mode hôte/cible, LynxOS est à part puisqu'il se présente comme un système complet. Pour développer sous LynxOS, il est préconisé d'utiliser plutôt le mode natif.

Nous avons donné nos résultats sous forme de tableaux, nous en donnons trois issus de l'étude. Le tableau 1. concerne la fonction de déverminage. Le tableau 2. donne les avantages et les inconvénients du point de vue des outils de développement. Le tableau 3 concerne la fonction de trace.

| LynxOS | VxWorks | pSOS+ |
|---|---|---|
| • nombreuses fonctions (basées sur GNU Gdb)<br>• informations sur les objets système<br>• interface graphique<br>• facile à utiliser | • nombreuses fonctions (basées sur GNU Gdb)<br>• informations sur les objets système<br>• interface graphique paramétrable<br>• facile à utiliser<br>• excellente intégration | • facilité de mise en œuvre<br>• nombreuses fonctions<br>• informations sur les objets système<br>• interface graphique<br>• bonne intégration<br>• pas de recherche de motif en mémoire |

**Tableau 1. Fonction de Déverminage**

| LynxOS | VxWorks | pSOS+ |
|---|---|---|
| • env. X Window (multi-fenêtrage)<br>• coloration syntaxique et indentation automatique dans l'éditeur<br>• outils GNU<br>• manque d'intégration<br>• pas de générateur de fichiers make<br>• pas de gestion de version | • concepts Unix<br>• outils GNU<br>• coloration syntaxique dans l'éditeur<br>• excellente intégration<br>• interface graphique paramétrable<br>• pas de générateur de fichiers make<br>• pas de gestion de version | • outils de recherche et de gestion des sources<br>• arbre d'utilisation ses symboles<br>• gestion des versions<br>• menu d'exécution des cibles des makefiles<br>• bonne intégration<br>• pas d'outil de génération des fichiers make |

**Tableau 2. Environnement de Développement**

| LynxOS | VxWorks | pSOS+ |
|---|---|---|
| • Pas d'outils de gestion de trace livré avec la version évaluée | • trace les opérations réalisées sur la cible<br>• traces des appels systèmes<br>• traces des tâches (changement de contexte)<br>• positionnement d'évènements déclencheurs<br>• filtre paramétrable<br>• très ergonomique, utilisation souple et très facile | • trace les opérations réalisées sur la cible<br>• traces des appels systèmes<br>• traces des tâches (changement de contexte)<br>• positionnement d'évènements déclencheurs<br>• filtre paramétrable<br>• très ergonomique, utilisation facile |

**Tableau 3. Fonction de Trace**

D'une manière générale, nous remarquons que les outils d'ingénierie fournis avec les systèmes d'exploitation VxWorks et pSOS+ sont bien intégrés et leur utilisation est facilitée par une interface graphique paramétrable. La meilleure intégration des divers outils est faite par VxWorks. Les outils utilisables avec LynxOS sont moins bien intégrés et leur facilité d'utilisation est variable, car certains outils possèdent une interface graphique très complète (par exemple le dévermineur), alors que d'autres sont rustiques et offrent un mode texte (par exemple l'outil de recherche dans les fichiers se réduit à la commande grep).

Les outils de déverminage sont très complets sur les trois systèmes d'exploitation et leur utilisation est aisée. Sur VxWorks et pSOS+, l'outil de trace comporte de nombreuses

fonctionnalités, par contre aucun outil de ce type n'est fourni avec la version de LynxOS que nous avons testée. Les trois systèmes comportent aussi des outils d'observation des performances et de dimensionnement pour l'optimisation du système (profiler) ayant de nombreuses fonctionnalités.

L'administration des systèmes d'exploitation évalués se réduit à du paramètrage de noyau, et à la configuration de la partie réseau. LynxOS s distingue puisqu'il est bâti suivant le modèle Unix, et s'administre de la même façon.

Ces résultats sont cependant à nuancer : ils ne sont valables qu'à un instant donné, pour une version donnée. La seule conclusion possible est sur la tendance. Les constructeurs sortent régulièrement des nouvelles versions qui sont de mieux en mieux outillées. En effet, LynxOS, pSOS et WxWorks ont très nettement perçu l'intérêt de fournir à leur client des outils assurant une réduction (coûts et délais) des phases d'ingénièrie et de maintenance d'une application temps réel.

## 6 Conclusion et Perspectives

Cette évaluation a permis de vérifier la bonne tenue de trois systèmes temps réel par rapport aux contraintes de temps énoncées lors d'enquêtes sur des applications d'informatique industrielle. Elle a permis également de qualifier les systèmes en ce qui concerne les fonctions d'ingénierie, aspect très souvent oublié dans ce genre d'études qui se limitent la plupart des cas à une comparaison de performances brutes.

Plusieurs types de bilans sont possibles. Le premier concerne le temps consacré à la partie étude de performances proprement dite. Nous avons mis plus de temps que prévu à faire nos tests, ceci étant dû essentiellement à la partie matérielle. L'installation, même si elle est bien documentée pose des problèmes de contraintes matérielles qui ne sont généralement pas immédiats à résoudre. pSOS+ a été assez long à tester pour cette raison. Par contre, l'installation et l'évaluation de VxWorks a été relativement rapide car nous avons bénéficié d'une personne à temps plein pendant deux jours. Nous avons opté pour tester les systèmes temps réel les uns après les autres car nous étions limités à une seule station de développement. Ce choix a finalement allongé la durée de l'évaluation, nous recommandons, si le nombre de cibles est suffisant d'évaluer tous les systèmes en même temps.

Le monde des systèmes embarqués change rapidement et depuis le début de cette étude, Wind River (VxWorks) a racheté Integrated Systems Inc. (pSOS+) et Lynx Real-Time Systems est devenu LynuxWorks et vend un noyau LinuX étendu avec des fonctionnalités temps réel ainsi qu'un ensemble d'outils de développement (compilateurs croisés, librairies, debugger...). Il semble qu'ils gardent une même approche que LynxOS au niveau de l'environnement de développement. Il serait intéressant d'évaluer ce nouveau système.

Ce travail va être suivi d'une évaluation des différentes solutions de répartition disponibles sur ces systèmes temps réel. En effet une autre des caractéristiques des applications industrielles visées est d'être fortement réparties. Il est donc nécessaire de proposer des solutions permettant de gérer cette répartition au-dessus des systèmes temps réel que nous venons d'étudier. Les solutions qui s'orientent essentiellement vers CORBA [OMG00] et son extension temps réel [OMG99] dont TAO [SCH98] peut être

vu comme un bon représentant. Java muni d'une extension temps réel également [JCO99,NIS99] s'avère être une autre piste. Il nous apparaît fondamental par la suite d'évaluer comment ces solutions savent prendre en compte les propriétés temps réel, quand elles existent, des systèmes sous-jacents, ces questions ont été abordées dans [LIZ00] et [BAC00].

**Bibliographie :**


[BAC98a] L. Bacon, E. Gressier-Soudan. Etude des contraintes utilisateur pour les systèmes d'exploitation temps réel - Questionnaire. HP-32/98/095/A. Septembre 1998.

[BAC98b] L. Bacon, E. Becquet, E. Gressier-Soudan. SETR : Systèmes d'Exploitation temps réel – Compte rendu des présentations des fournisseurs. HP-32/98/077/A. 1998.

[BAC99a] L. Bacon, E. Gressier-Soudan. SETR : Systèmes d'Exploitation Temps Réel - Préconisation d'utilisation et normes. HP-32/98/095/A .1999.

[BAC99b] L. Bacon, E. Gressier-Soudan. SETR : Systèmes d'Exploitation Temps Réel - Etude des contraintes utilisateurs – Synthèse entretiens. HP-32/98/079/A. 1999.

[BAC99c] L. Bacon, E. Becquet, E. Gressier-Soudan. Etude comparative de micro-noyaux temps réel du commerce. 7ème Séminaire du Laboratoire d'Ingénierie de la Sûreté de fonctionnement (LIS). Bordeaux. 16-17 Mars 1999.

[BAC99d] L. Bacon, P. Decogné. SETR : systèmes d'exploitation temps réel – LynxOS – Architecture . HP-32/99/099/A. Juillet 1999.

[BAC99e] L. Bacon, E. Becquet. SETR : systèmes d'exploitation temps réel – pSOS – Architecture. HP-32/98/098/A. Mars 1999.

[BAC99f] L. Bacon, E. Becquet. SETR : systèmes d'exploitation temps réel – VxWorks – Architecture. HP-32/98/100/A. Septembre 1999.

[BAC00] L. Bacon, E. Becquet, E. Gressier-Soudan, C. Lizzi, C. Logé, L. Reveilleau. Provisioning QoS in Real-Time Distributed Object Architectures for Power Plant Control Applications. DOA'2000. September 2000.

[BEC00] E. Becquet, L. Réveilleau, L. Bacon, E. Gressier-Soudan, F. Horn. "Towards a Java Based Embedded Remote Monitoring Tool for Small and Medium Power Plant Units". JISA'2000. Anaheim. 6-8 Décembre 2000.

[JCO99] J consortium. Draft International J Consortium Specification. September 1999.

[KAR89] R. P. Kar, K. Porter. Rhealstone: A Real-Time Benchmarking Proposal, An independently verifiable metric for complex multitaskers. Dr. Dobb's Journal. February 1989.

[KAS90] Glenn Kasten, David Howard, Bob Walsh, (letter authors) and Rabindra P. Kar (answers to the authors) . LETTERS : Rhealstone Recommendations. Dr. Dobb's Journal. September 1990.

[LIZ00] C. Lizzi, L. Bacon, E. Becquet, E. Gressier-Soudan. Prototyping QoS based Architecture for Power Plant Control Applications. In Proceedings of the

3rd IEEE International Workshop on Factory Communication Systems (WFCS'2000). Porto, Portugal. September 2000.

[MCV96] L. McVOY, C. STAELIN. Lmbench : Portable tools for performance analysis. In Proc. Winter 1996 USENIX, San Diego, CA, pp. 279-284. January 1996.

[NIS99] NIST. Report from the Requirements Group for Real-Time Extensions For the Java Platform. September 1999.

[OMG99] Object Management Group. Real-Time CORBA. Joint Revised Submission. March 1999.

[OMG00] Object Management Group. The Common Object Request Broker : Architecture and Specification. Revision 2.4. Octobre 2000.

[SCH98] D.C. Schmidt, D.L. Levine, S. Mungee. The Design of the TAO Real-Time Object Request Broker. Computer Communications, Elvisier Science, Volume 21, No 4, April 1998.

[URL01] <URL :www.sun.com/chorusos>

[URL02] <URL :www.lynx.com>

[URL03] <URL :www.isi.com>

[URL04] <URL :www.windriver.com>