5. TRAVAUX DE L'ÉQUIPE CRATÈRE

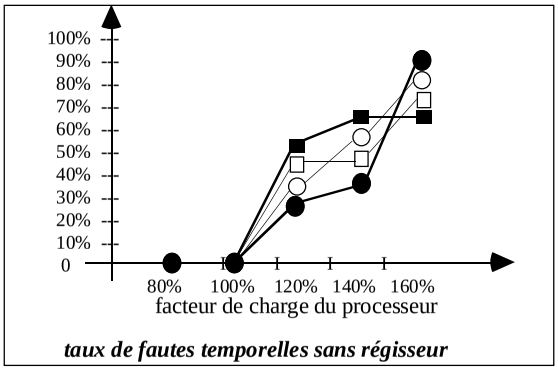
1. RÉSORPTION CONTRÔLÉE DES PANNES TEMPORELLES DUES AUX SURCHARGES

SURCHARGES FONCTIONNELLES
augmentation de la durée d'exécution des tâches, de la durée de transmission des messages
avalanche de tâches apériodiques (incidents ou alarmes) plus nombreuses qu'attendues
SURCHARGES DES RESSOURCES PARTAGÉES
augmentation de probabilité de congestion, de conflits d'accès, d'inversion de priorités, d'interblocage
PANNES D'UNE PARTIE DU SYSTÈME : processeur, liaison

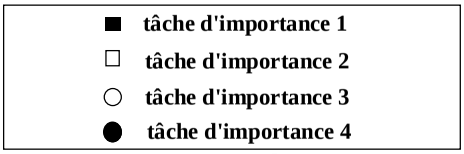
INSTABILITÉS DES ALGORITHMES D'ORDONNANCEMENT DU PROCESSEUR
faute temporelle subie par une tâche non fautive, choix aléatoire de la tâche touchée
avalanches de fautes temporelles par report des retards
Avec un serveur des tâches apériodiques : favorise les tâches périodiques et supprime des apériodiques

IMPORTANCE
Distinguer urgence et primordialité. Plus la tâche est vitale, plus longtemps il faut éviter de la supprimer rang d'importance = rang de suppression.
Importance gérée par un régisseur d'importance selon un contrat défini pour chaque tâche de l'application

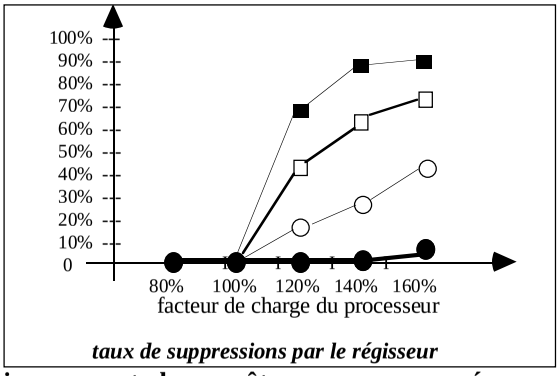
STABILISATION PAR IMPORTANCE



taux de fautes temporelles sans régisseur

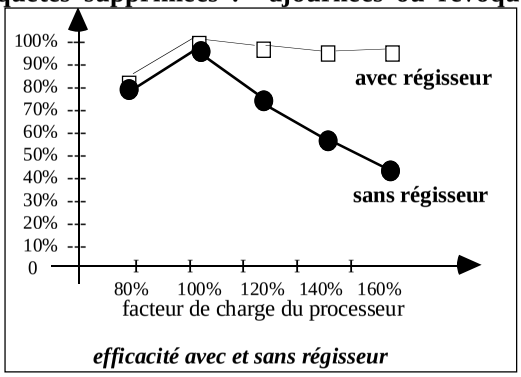


importance = rang de suppression



taux de suppressions par le régisseur

ajournement de requêtes non commencées
révocation de requêtes commencées
requêtes supprimées := ajournées ou révoquées



efficacité avec et sans régisseur

Facteur de charge du processeur : Σ (Ci/Pi)

Efficacité : Σ (Ni*Ci/d)
où Ni : nombre de requêtes menées à terme
Ci : durée de la requête
d : durée de l'expérience et du comptage des Ni requêtes à terme := ni ajournées ni révoquées

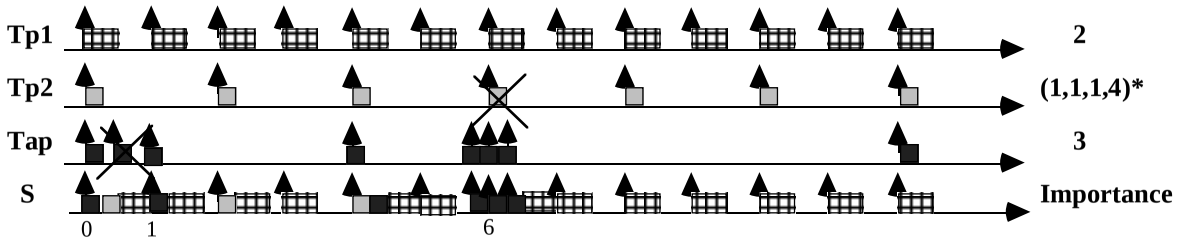
NOUVEAU TEST D'ACCEPTABILITÉ MIS EN OEUVRE PAR LE RÉGISSEUR

test d'acceptabilité d'une nouvelle requête s déclenchée à t (date de réveil r = t, durée C, échéance D, Imp)
Soit R(t) : liste des requêtes déclenchées à t, ordonnées selon l'algorithme d'ordonnancement (EDF, RM)
soit laxité conditionnelle d'une requête i de R(t) :

LCi(t) = Di(t) - Σ Cj(t), avec Σ pour tout j ∈ R(t) tel que rang(j) <= rang(i) [EDF : dj <= di)]
avec Cj(t) = durée résiduelle de la requête j
Alors laxité de R(t) : L(t) = Min (LCi(t)) pour tout i de R(t)

TEST(s) : si L(t) >= 0 alors la requête s est acceptée;
sinon boucle soit requête u telle que Imp(u) = Min(Imp(i) pour i ∈ R(t))
si Imp(u) >= Imp(s) alors refuser et ajourner la requête s; fin de boucle;
sinon R(t) = R(t) - u; révoquer u ; si L(t) >= 0 alors la requête s est acceptée;
fin de boucle; fin si;

EXEMPLE AVEC IMPORTANCE VARIABLE : COMMANDE DE DÉGRADATION DOUCE
tâches périodiques Tp1(r = 0 + k*T, C = T/2, D = T +k*T), Tp1(r = 0 + k*2T, C = T/4, D = T/2 +k*T)
tâches apériodiques Tap(r, C = T/4, D = r + T/2)



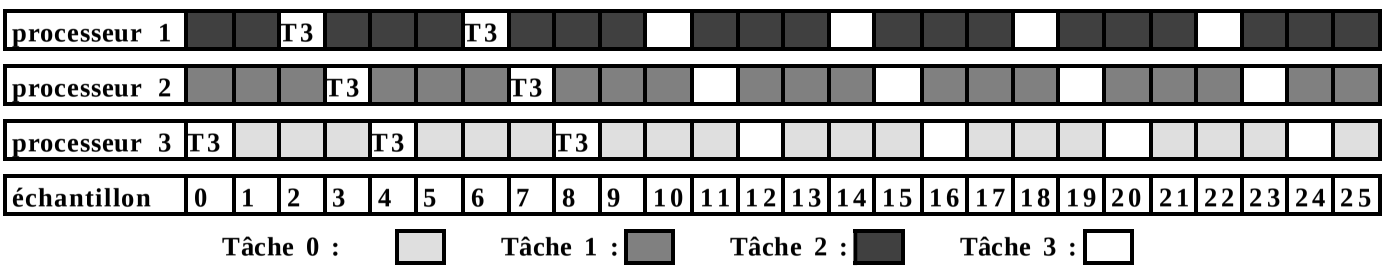
À 6T, Tap et Tp2 ont même échéance, Tap est élu. À 6T + T/4, laxité négative, Tp2 est ajournée

TOLÉRANCE AUX PANNES PAR VALEUR D'IMPORTANCE VARIABLE

Tâches périodiques de commande d'un asservissement échantillonné exécutées sur quatre processeurs
T0 (premier déclenchement r0 = 00 période P = 0,200 durée C0 = 0,110 délai critique R0 = 0,200)
T1 (premier déclenchement r0 = 20 période P = 0,200 durée C0 = 0,110 délai critique R0 = 0,180)
T2 (premier déclenchement r0 = 40 période P = 0,200 durée C0 = 0,110 délai critique R0 = 0,160)
T3 (premier déclenchement r0 = 60 période P = 0,200 durée C0 = 0,110 délai critique R0 = 0,140)
En cas de panne d'un processeur, on préfère sauter un échantillon sur quatre .

Commande avec importance variable : Imp(kème requête de Ti) = (i + k) modulo 4

Algorithme réparti de placement dynamique par importance
les requêtes sont envoyées à chaque processeur, chacun, i, calcule la laxité de la liste des requêtes locales et diffuse l'importance de la requête qu'il doit supprimer (s'il ne supprime rien, il diffuse - i)
Election répartie du processeur avec la plus petite réponse.

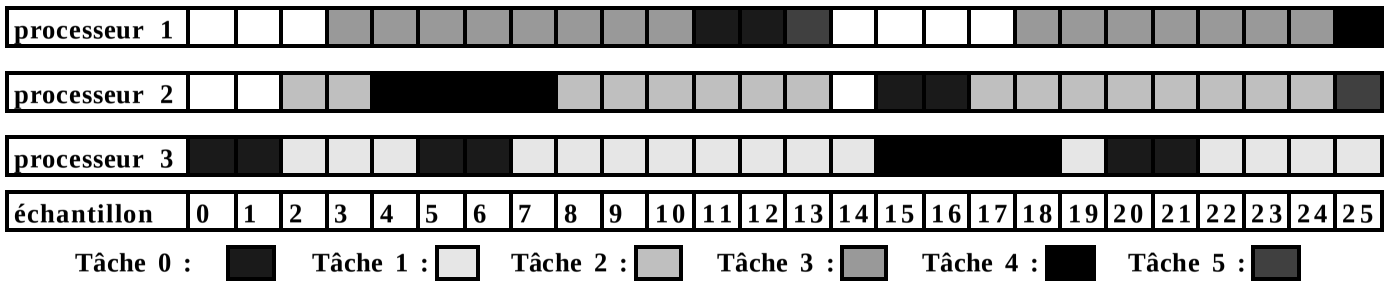


PLACEMENT DYNAMIQUE COMMANDÉ PAR L'IMPORTANCE

Tâches périodiques d'un système réparti à exécuter sur trois processeurs

T0 (premier déclenchement r0 = 00 période P = 100 durée C0 = 50 délai critique R0 = 100)
T1 (premier déclenchement r0 = 20 période P = 300 durée C0 = 180 délai critique R0 = 300)
T2 (premier déclenchement r0 = 40 période P = 300 durée C0 = 160 délai critique R0 = 280)
T3 (premier déclenchement r0 = 60 période P = 200 durée C0 = 160 délai critique R0 = 260)
T4 (premier déclenchement r0 = 80 période P = 200 durée C0 = 80 délai critique R0 = 120)
T5 (premier déclenchement r0 = 120 période P = 300 durée C0 = 20 délai critique R0 = 200)

La configuration représente une charge de 262% et n'a pas de solution en placement statique



Tâche 0 : Tâche 1 : Tâche 2 : Tâche 3 : Tâche 4 : Tâche 5 :

échantillon de 20 millisecondes
certaines tâches, T1, T2, T3, ont toutes leurs requêtes sur le même processeur ; elles ne migrent pas.
T0_0, T0_1 sont sur p3, T0_2 est sur p1, T0_3 est sur p2, T0_4 est sur p3, T0_5 est sur p2,...
T4_0 est sur p2, T4_1 est sur p3, T4_2 est sur p1, T4_3 est sur p2, T4_4 est sur p3,...
T5_0 est sur p1, T5_1 est sur p2, T5_2 est sur p1, T5_3 est sur p2,...

PLACEMENT DYNAMIQUE DE TÂCHES TEMPS RÉEL (PETRARQUE, TSI 17,1 janvier 1998)

1) ***à la date t de déclenchement d'une nouvelle requête req avec :*** **req : C(t), R(t) ou échéance(t), Imp**
=> diffusion de la requête à tous les sites s par MESSAGE_D(req)

2) ***analyse concurrente de la requête sur chaque site s_i à la réception de MESSAGE_D(req)***

soit $R_i(t)$, liste des requêtes déclenchées et déjà placées sur s_i,
on fait : $R'_i(t) = R_i(t) \cup \{req\}$ et le site s_i calcule $L'(t)$ la laxité de $R'_i(t)$
•• si $L'(t) < 0$, le site s_i crée $R''_i(t)$ en éliminant les tâches de plus faible importance de $R'_i(t)$
jusqu'à ce que $L''(t) \geq 0$ où $L''(t)$ = laxité de $R''(t)$
Appelons $Imp(surcharge_i)$ = imp max des tâches supprimées de $R'(t)$
• si $Imp(surcharge_i) = Imp$, la requête ne peut être acceptée par s_i
• si $Imp(surcharge_i) < Imp$, la requête pourrait être acceptée en éliminant une requête déjà déclenchée et d'importance $Imp(surcharge_i)$
=> diffusion de MESSAGE_E(s_i, $Imp(surcharge_i)$)
•• si $L'(t) \geq 0$, le site s_i peut accepter la requête sans problème, et par convention $Imp(surcharge_i) = -s_i$ et il diffuse MESSAGE_E(s_i, $-s_i$)

3) ***phase d'attente de tous les messages des sites de placement acceptables***
durée limitée au délai maximal de diffusion sur le réseau

4) ***élection entre les sites de placement acceptables après réception des MESSAGE_E(s_i, $Imp(surcharge_i)$)***
soit s'_i tel que $Imp(surcharge_i') \leq Imp(surcharge_i)$ pour tous les messages reçus pendant ce délai par s_i
si $s'_i = s_i$ alors s_i est élu et req est placée sur s_i
si $Imp(surcharge_i) <$ alors req est acceptée sans remaniement de $R'_i(t)$
sinon on élimine les requêtes d'importance inférieures à Imp avant d'accepter req, en utilisant $R''_i(t)$

NOTA : Hypothèse de communication synchrone (délais d'attente bornés)

IMPORTANCE ET RÉGISSEUR

CONTRAT D'IMPORTANCE

contrat défini hors ligne par le concepteur : consignes pour l'exécutif au niveau tâche ou requête

paramètre spécifique : l'importance ; paramètres complémentaires : révocabilité, ajournabilité d'une requête

MODÈLE DE TÂCHE AVEC PLUSIEURS MODES DE FONCTIONNEMENT

Tâche T est

Contrat : expression des tolérances de suppression et des conditions de changement de mode

Début

Mode_Normal : actions du mode normal (durée d'exécution C, Propriétés, Importance);

Mode_Révocation : actions du mode révocation (durée d'exécution Cr, Propriétés, Importance);

Mode_Ajournement : actions du mode normal (durée d'exécution Ca, Propriétés, Importance);

Mode_Faute_Temporelle : actions du mode faute (durée d'exécution Cf, Propriétés, Importance);

Fin;

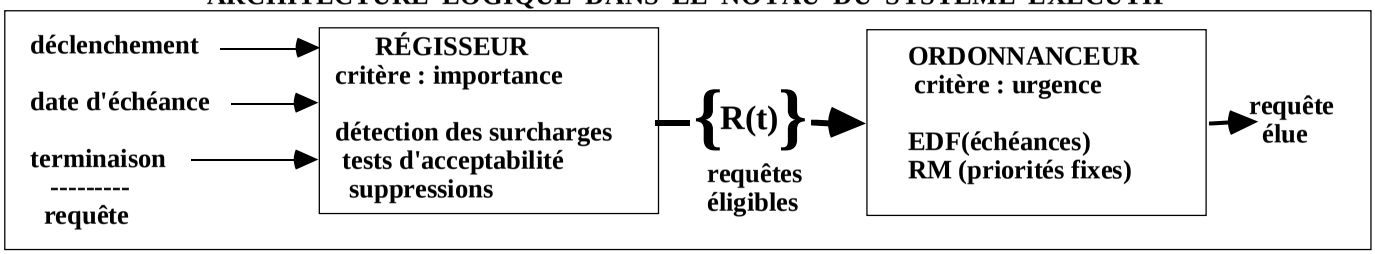
RÉGISSEUR : automate chargé de mettre en oeuvre le contrat

événements chronométriques : déclenchement, arrivée à échéance, terminaison d'une requête

résultat : ensemble de requêtes à ordonnancer, avec leurs modes d'exécution

utilise le test d'acceptabilité avec importance pour décider des changements de mode ou la suppression

ARCHITECTURE LOGIQUE DANS LE NOYAU DU SYSTÈME EXÉCUTIF



2. EXPÉRIMENTATIONS SUR DES PLATES-FORMES

PLATE-FORME CHORUS (1994-1998)

dans micro-noyau CHORUS, intégration d'un régisseur et d'un ordonnanceur à échéance

a) régisseur simplifié : importance + suppression ou non de requête + un seul mode normal pour les tâches

but : remédier à l'instabilité de la politique "earliest deadline first"

b) placement dynamique par importance dans un système réparti comprenant quatre processeurs

PLATE-FORME LINUX-RT (1999-2001)

modification du noyau RT-LINUX pour installer un régisseur et un ordonnanceur à échéance

mode normal et modes de survie pour les tâches + ajournement ou révocation de requête

TRAVAUX DE L'ÉQUIPE CRATÈRE

3. SÛRETÉ ET EFFICACITÉ DE LA CONCURRENCE DANS LES APPLICATIONS TEMPS RÉEL

3.1. sûreté de la concurrence, ruinée par la priorité, sauvée par passe-droit au leader

Allocation dynamique d'un ensemble de m ressources identiques à n tâches

Prévention d'interblocage par l'algorithme du banquier : annonce de la demande max de chaque tâche
Cas simplifié où toutes les annonces sont égales à C
Soit Ai(t) le nombre de ressources déjà allouées à la tâche Ti, soit $Dist_i(t) = C - Ai(t)$.
Soit leader(t) la tâche k pour laquelle $Dist_k(t) = Min(Dist_i(t))$
Prévention d'interblocage. CNS : garder assez de ressources pour pouvoir toujours servir la tâche leader(t)

EXEMPLE, sans contrainte d'ordre (priorité ou FIFO) : n = 3, m = 6 et C = 4.
À t0, on a A(t0) = (0, 2, 4) ; Dist(t0) = (4 2 0) ; Stock = 0
à t1, T1 demande 1 ressource, T1 est bloquée,
à t2, T2 demande 1 ressource, T2 est bloquée,
à t3, T3 libère 2 ressources, Stock = 2,
T1 n'est pas servie car avec Stock = 1, A=(1, 2, 2) n'est pas fiable ; T2 est servie car A=(0, 3, 2) est fiable

MÊME EXEMPLE avec service selon les priorités (prio(T1)>prio(T2)>prio(T3)) ou selon FIFO
à t3, T1 ne peut être servie, donc on arrête le service car prio(T1)>prio(T2)
à t4, T3 demande une ressource et est mis en queue car prio(T1)>prio(T2)>prio(T3).
c'est une situation d'INTERBLOCAGE

SOLUTION : à la fin d'un service, on doit toujours regarder si la tâche leader(t) attend, et si oui, la servir
PREUVES par réseaux de Petri colorés

3.2. limitation de la taille de files d'attente et gain en efficacité

EXEMPLE : n = 100, m = 200, C = 4 pare-feu de réseaux avec 53 tâches de routine et 47 tâches réactives

Scénario d'avalanche (pire cas de fonctionnement)
•• Service avec prévention d'interblocage et service ordonné(FIFO) non ruiné
à t0, allocation A(t0) = (A[1..53] = 3, A[54..100] = 0), Stock = 41.
à t1, avalanche de tentatives d'intrusion => chaque tâche réactive demande 1 ressource. On peut en servir 40
à t2, chaque tâche de routine demande 1 ressource de plus => 1 est servie et file d'attente avec 7 + 52 tâches
à t3 + k, chacune des 40 tâches réactives non bloquées demande 2 ressources de plus attente 7 + 52+ 40
une seule tâche, de routine, n'est pas bloquée!
Quand les tâches sont servies, elles s'exécute un temps puis libèrent des ressources, revenant à l'état de t0

•• Service avec antichambre et nf tâches autorisées à faire des demandes
si nf = 66, avec 200 ressources, il y a assez de ressources pour qu'il n'y ait jamais possibilité d'interblocage (on réserve au moins 3 ressources par tâche des nf tâches, + 1 ressource pour qu'une tâche atteigne C)

•• RÉSULTATS DE SIMULATIONS

nf	*allocation*	*antichambre*	*rétention*	*concurrence*	*temps total*
100	prévention	non	61%	18	1213
82	prévention	oui	26%	37	608
66	sans souci	oui	13%	51	407
50	sans souci	oui	8%	37	610

rétention : taux moyen de ressources inutilisées alors qu'il y a des tâches en attente de ressource
concurrence : nombre moyen de tâches que l'allocateur autorise à être concurrentes en toute sûreté
CONCLUSION : pour une meilleure efficacité, assurer un fort facteur de concurrence

Bibliographie :

[Ada 1995a] [ADA 95 REFERENCE MANUAL : *Language and Standard Libraries. International standard ANSI/ISO/IEC-8652*, 1995 Also available from Springer-Verlag, LNCS 1246],

[Ada 1995b] [ADA 95 RATIONALE : *Language and Standard Libraries.* Intermetrics, 1995 Also available from Springer-Verlag, LNCS 1247]

[Blair 1998] BLAIR G., STEFANI J-B., *Open Distributed Processing and Multimedia*, 452 pages, Addison-Wesley, 1998

[Bonnet 1999] BONNET C., DEMEURE I, *Introduction aux systèmes temps réel*, 207 pages, Paris, Hermès Science, 1999

[Burns 2001] [BURNS A. Guide for the use of the Ada Ravenscar Profile in high Integrity Systems. *Ada User Journal,* 22,4 September 2001]

[Chevochot 1999] CHEVOCHOT P., PUAUT I., Tolérance aux fautes dans les systèmes répartis temps réel strict, *TSI 18,8*, pages 837-870, Hermès, 1999

[Cottet 1999] COTTET F., DELACROIX J., KAISER C., MAMMERI Z., *Ordonnancement temps réel, ordonnancement centralisé, ordonnancement réparti*, Collection Techniques de l'Ingénieur, Traité Mesures et contrôle, Articles R8055, 28 pages, et R8056, 26 pages, 1999.

[Cottet 2000] COTTET F., DELACROIX J., KAISER C., MAMMERI Z., *Ordonnancement temps réel*, 207 pages, Paris, Hermès Science, 2000.

[Delacroix 1994] DELACROIX J. Stabilité et régisseur d'ordonnancement en temps réel, *TSI 3(2)*, pages 223-250, 1994

[Delacroix 2000] DELACROIX J., MÉNIVAL C., Intégration d'un contrôle de charge par importance au sein du systeme RT-Linux, *Congrès RTS'2000*, Paris, 2000

[Halbwachs 1993] [HALBWACHS N., *Synchronous programming of reactive systems*, Kluwer Academic Publishers, 174 pages, 1993]

[Humpris 20 01]. [HUMPRIS D. Integrating Ada into a Distributed Systems Environment. *Ada User Journal*, 22,1 March 2001]

[ISO 1995] INTERNATIONAL STANDARD ORGANISATION, *Ada 95 Reference Manual*, ISO Standard 8652/1995, ISO, Genève 1995

[Kaiser 1998a] KAISER C., SANTELLANI C., Pétrarque : Plate-forme d'expérimentation pour l'ordonnancement adaptatif temps réel strict d'applications réparties. *Revue T.S.I. Technique et Science Informatique*, vol 17 n° 1, pages 39-62, janvier 1998

[Kaiser 1998b] KAISER C, PRADAT-PEYRE.J.F. Reliable, Fair and Efficient Concurrent Software with Dynamic Allocation of Identical Resources. *Congrès MCSEAI'98*, pages 109-125, Tunis, December 1998.

[Kaiser 2001] KAISER C., Description et critique d'un système temps réel pour le suivi d'un laminoir, *TSI 20,2*, pages 179-211, Hermès, 2001

[Kopetz 1989] KOPETZ H., et al., Distributed Fault-Tolerant Real-Time systems: The Mars Approach, *IEEE Micro*, vol. 9, pages 25-40, 1989

[Kopetz 1997] KOPETZ H., *Real-Time Systems : Design Principles for Distributed Embedded Applications*, Kluwer Academic Publishers, 1997

[Ledinot 1998] LEDINOT E., LELANN G., Évaluation d'une méthode de génie système pour l'avionique modulaire, *Proceedings of Real-Time Systems*, p. 267-275, Paris 1999

[Lelann 1999] LELANN G., Designing real-time dependable distributed systems, *Computer Communications Journal*, vol.15,4, pages 225-234, 1992

[Ménival 2000] MÉNIVAL C., *Intégration d'un contrôle de charge par importance au sein du systeme RT-Linux*, Mémoire d'ingénieur, CNAM 2000

[OMG 2001] OBJECT MANAGEMENT GROUP, Realtime CORBA Joint Revised Submission. *OMG Document orbos*/01-04-01, 2001

[Pautet 1998] [PAUTET L., QUINOT T., TARDIEU S. Corba &DSA: Divorce or Marriage? Int. Conf. on Reliable Software Technologies, Ada-Europe'99, in *LNCS 1622*, Springer Verlag pages 211-225, June 1999]

[Pinho 2001] [PINHO L.M., Session Summary: distribution and Real-Time. Proceedings of the 10th International Real-Time Ada Workshop, *Ada Letters* 21(1), ACM SIGAda, 2001],

[Richard 2001] P. RICHARD P., COTTET F., KAISER C. Précédences généralisées et ordonnançabilité des tâches de suivi temps réel d'un laminoir. *Revue APII-JESA*, pp. 1055-1071, vol.35 n° 9. 2001

[Trinquet 1999] TRINQUET Y., ELLOY J-P., *Les systèmes d'application temps réel*, Collection Techniques de l'Ingénieur, Traité Mesures et contrôle, Article R8054, 25 pages, 1999.

[Zaffalon 1999] ZAFFALON L., BREGUET P., *Programmation concurrente et temps réel avec Ada 95*, Presses Polytechniques et Universitaires Romandes 1999 (559 pages)

[Wellings 1998] WELLINGS A., BEUS-DUKIC L. et POWELL D., Real-Time Scheduling in a Generic Fault-Tolerant Architecture, *Proceedings of the 19th IEEE real-Time Systems Symposium*, pages 390-398, 1998

≈≈

ADA : Ada-France [ada-france.org/] Ada Europe page : [www.ada-europe.org] Adalog perso.: [wanadoo.fr/adalog]

GNAT compilateur Ada en logiciel libre [http://wwwgnat.com]. [http ://www.act-europe.fr/]

JAVA TEMPS RÉEL : [http://www.j-consortium.org] [//www.rtj.org] [http://www.real-time.org] [http://www.drtsj.org]

LINUX : [http://www.linux.org]

UML Temps réel : [http://www.omg.org]

≈≈≈≈≈≈≈≈≈≈≈≈≈≈≈≈≈≈≈≈≈≈≈≈≈≈≈≈≈≈≈≈

LABORATOIRE CÉDRIC : http://cedric.cnam.fr