Modèles à files d'attente et gestion des ressources dans les systèmes informatiques

Dominique Potier

THESE

présentée à

Université Scientifique et Médicale de Grenoble

Institut National Polytechnique de Grenoble

pour obtenir le grade de

DOCTEUR ES SCIENCES
MATHEMATIQUES

par

Dominique POTIER

MODELES A FILES D'ATTENTE ET GESTION DES RESSOURCES DANS LES SYSTEMES INFORMATIQUES.

Thèse soutenue le 15 janvier 1977 devant la Commission d'Examen :

Président : L. BOLLIET

Examinateurs : D. FERRARI
E. GELENBE
S. KRAKOWIAK
C. PAIR
J. SAKAROVITCH
J.P. VERJUS

UNIVERSITE SCIENTIFIQUE
ET MEDICALE DE GRENOBLE

Monsieur Gabriel CAU : Président
Monsieur Pierre JULLIEN : Vice Président

MEMBRES DU CORPS ENSEIGNANT DE L'U.S.M.G.

PROFESSEURS TITULAIRES

MM.	ARNAUD Paul	Chimie
	AUBERT Guy	Physique
	AYANT Yves	Physique approfondie
Mme	BARBIER Marie-Jeanne	Electrochimie
MM.	BARBIER Jean-Claude	Physique Expérimentale
	BARBIER Reynold	Géologie appliquée
	BARJON Robert	Physique nucléaire
	BARNOUD Fernand	Biosynthèse de la cellulose
	BARRA Jean-René	Statistiques
	BARRIE Joseph	Clinique chirurgicale
	BEAUDOING André	Clinique de Pédiatrie et Puériculture
	BERNARD Alain	Mathématiques Pures
Mme	BERTRANDIAS Françoise	Mathématiques Pures
MM.	BERTRANDIAS Jean-Paul	Mathématiques Pures
	BEZES Henri	Pathologie chirurgicale
	BLAMBERT Maurice	Mathématiques Pures
	BOLLIET Louis	Informatique (IUT B)
	BONNET Georges	Electrotechnique
	BONNET Jean-Louis	Clinique ophtalmologique
	BONNET-EYMARD Joseph	Clinique gastro-entérologique
Mme	BONNIER Marie-Jeanne	Chimie générale
MM.	BOUCHERLE André	Chimie et toxicologie
	BOUCHEZ Robert	Physique nucléaire
	BOUSSARD Jean-Claude	Mathématiques appliquées
	BOUTET DE MONTVEL Louis	Mathématiques pures
	BRAVARD Yves	Géographie
	CABANEL Guy	Clinique rhumatologique et hydrologique
	CALAS François	Anatomie
	CARLIER Georges	Biologie végétale
	CARRAZ Gilbert	Biologie animale et pharmacodynamie
	CAU Gabriel	Médecine légale et toxicologie
	CAUQUIS Georges	Chimie organique
	CHABAUTY Claude	Mathématiques Pures
	CHARACHON Robert	Clinique Oto-rhino-laryngologique
	CHATEAU Robert	Clinique de neurologie
	CHIBON Pierre	Biologie animale
	COEUR André	Pharmacie chimique et chimie analytique
	CONTAMIN Robert	Clinique gynécologique
	COUDERC Pierre	Anatomie pathologique
Mme	DEBELMAS Anne-Marie	Matière médicale
MM.	DEBELMAS Jacques	Géologie générale
	DEGRANGE Charles	Zoologie
	DELORMAS Pierre	Pneumophtisiologie

MM. DEPORTES Charles	Chimie minérale
DESRE Pierre	Métallurgie
DESSAUX Georges	Physiologie animale
DODU Jacques	Mécanique appliquée (IUT A)
DOLIQUE Jean-Michel	Physique des plasmas
DREYFUS Bernard	Thermodynamique
DUCROS Pierre	Cristallographie
DUGOIS Pierre	Clinique de dermatologie et syphiligraphie
GAGNAIRE Didier	Chimie physique
GALLISSOT François	Mathématiques Pures
GALVANI Octave	Mathématiques Pures
GASTINEL Noël	Analyse numérique
GAVEND Michel	Pharmacologie
GEINDRE Michel	Electroradiologie
GERBER Robert	Mathématiques Pures
GERMAIN Jean-Pierre	Mécanique
GIRAUD Pierre	Géologie
JANIN Bernard	Géographie
KAHANE André	Physique générale
KLEIN Joseph	Mathématiques pures
KOSZUL Jean-Louis	Mathématiques pures
KRAVTCHENKO Julien	Mécanique
KUNTZMANN Jean	Mathématiques appliquées
LACAZE Albert	Thermodynamique
LACHARME Jean	Biologie végétale
Mme LAJZEROWICZ Janine	Physique
MM. LAJZEROWICZ Joseph	Physique
LATREILLE René	Chirurgie générale
LATURAZE Jean	Biochimie pharmaceutique
LAURENT Pierre	Mathématiques appliquées
LEDRU Jean	Clinique médicale B
LLIBOUTRY Louis	Géophysique
LOISEAUX Pierre	Sciences nucléaires
LONGEQUEUE Jean-Pierre	Physique nucléaire
LOUP Jean	Géographie
Melle LUTZ Elisabeth	Mathématiques Pures
MM. MALGRANGE Bernard	Mathématiques Pures
MALINAS Yves	Clinique obstétricale
MARTIN-NOEL Pierre	Clinique cardiologique
MAZARE Yves	Clinique médicale A
MICHEL Robert	Minéralogie et Pétrographie
MICOUD Max	Clinique maladies infectieuses
MOURIQUAND Claude	Histologie
MOUSSA André	Chimie nucléaire
MULLER Jean-Michel	Thérapeutique (néphrologie)
NEEL Louis	Physique du Solide
OZENDA Paul	Botanique
PAYAN Jean-Jacques	Mathématiques Pures
PEBAY-PEYROULA Jean-Claude	Physique
RASSAT André	Chimie systématique
RENARD Michel	Thermodynamique
REVOL Michel	Urologie
RINALDI Renaud	Physique
DE ROUGEMONT Jacques	Neuro-chirurgie
SEIGNEURIN Raymond	Microbiologie et Hygiène
SENGEL Philippe	Zoologie

MM.	SIBILLE Robert	Construction mécanique (IUT A)
	SOUTIF Michel	Physique générale
	TANCHE Maurice	Physiologie
	TRAYNARD Philippe	Chimie générale
	VAILLANT François	Zoologie
	VALENTIN Jacques	Physique nucléaire
	VAUQUOIS Bernard	Calcul électronique
Mme	VERAIN Alice	Pharmacie galénique
MM.	VERAIN André	Physique
	VEYRET Paul	Géographie
	VIGNAIS Pierre	Biochimie médicale
	YOCCOZ Jean	Physique nucléaire théorique

PROFESSEURS ASSOCIES

MM.	CLARK Gilbert	Spectrométrie physique
	CRABBE Pierre	CERMO
	ENGLMAN Robert	Spectrométrie physique
	HOLTZBERG Frédéric	Basses températures
	DEMBICKI Eugéniuz	Mécanique
	MATSUSHIMA Yozo	Mathématiques Pures

PROFESSEURS SANS CHAIRE

Mle	AGNIUS-DELORD Claudine	Physique pharmaceutique
	ALARY Josette	Chimie analytique
MM.	AMBROISE-THOMAS Pierre	Parasitologie
	BELORIZKY Elie	Physique
	BENZAKEN Claude	Mathématiques appliquées
	BIAREZ Jean-Pierre	Mécanique
	BILLET Jean	Géographie
	BOUCHET Yves	Anatomie
	BRUGEL Lucien	Energétique (IUT A)
	BUISSON René	Physique (IUT A)
	BUTEL Jean	Orthopédie
	COHEN ADDAD Pierre	Spectrométrie physique
	COLOMB Maurice	Biochimie
	CONTE René	Physique (IUT A)
	DEPASSEL Roger	Mécanique des fluides
	FONTAINE Jean-Marc	Mathématiques Pures
	GAUTHIER Yves	Sciences Biologiques
	GAUTRON René	Chimie
	GIDON Paul	Géologie et Minéralogie
	GLENAT René	Chimie organique
	GROULADE Joseph	Biochimie médicale
	HACQUES Gérard	Calcul numérique
	HOLLARD Daniel	Hématologie
	HUGONOT Robert	Hygiène et Médecine préventive
	IDELMAN Simon	Physiologie animale
	JOLY Jean-René	Mathématiques Pures
	JULLIEN Pierre	Mathématiques appliquées
Mme	KAHANE Josette	Physique
MM.	KRAKOWIACK Sacha	Mathématiques appliquées
	KUHN Gérard	Physique (IUT A)
	LE ROY Philippe	Mécanique (IUT A)
	LUU DUC Cuong	Chimie organique

MM. MAYNARD Roger	Physique du solide
Mme MINIER Colette	Physique (IUT A)
MM. PELMONT Jean	Biochimie
PERRIAUX Jean-Jacques	Géologie et Minéralogie
PFISTER Jean-Claude	Physique du solide
Mle PIERY Yvette	Physiologie animale
MM. RAYNAUD Hervé	M.I.A.G.
REBECQ Jacques	Biologie (CUS)
REYMOND Jean-Charles	Chirurgie générale
RICHARD Lucien	Biologie végétale
Mme RINAUDO Marguerite	Chimie macromoléculaire
MM. ROBERT André	Chimie papetière
SARRAZIN Roger	Anatomie et chirurgie
SARROT-REYNAULD Jean	Géologie
SIROT Louis	Chirurgie générale
Mme SOUTIF Jeanne	Physique générale
MM. STREGLITZ Paul	Anesthésiologie
VIALON Pierre	Géologie
VAN CUTSEM Bernard	Mathématiques appliquées

MAITRES DE CONFERENCES ET MAITRES DE CONFERENCES AGREGES

MM. AMBLARD Pierre	Dermatologie
ARMAND Gilbert	Géographie
ARMAND Yves	Chimie (IUT A)
BACHELOT Yvan	Endocrinologie
BARGE Michel	Neuro chirurgie
BARJOLLE Michel	M.I.A.G.
BEGUIN Claude	Chimie organique
Mme BERIEL Hélène	Pharmacodynamie
MM. BOST Michel	Pédiatrie
BOUCHARLAT Jacques	Psychiatrie adultes
Mme BOUCHE Liane	Mathématiques (CUS)
MM. BRODEAU François	Mathématiques (IUT B)
CHAMBAZ Edmond	Biochimie médicale
CHAMPETIER Jean	Anatomie et organogénèse
CHARDON Michel	Géographie
CHERADAME Hervé	Chimie papetière
CHIAVERINA Jean	Biologie appliquée (EFP)
CONTAMIN Charles	Chirurgie thoracique et cardio-vasculaire
CORDONNIER Daniel	Néphrologie
COULOMB Max	Radiologie
CROUZET Guy	Radiologie
CYROT Michel	Physique du solide
DELOBEL Claude	M.I.A.G.
DENIS Bernard	Cardiologie
DOUCE Roland	Physiologie végétale
DUSSAUD René	Mathématiques (CUS)
Mme ETERRADOSSI Jacqueline	Physiologie
MM. FAURE Jacques	Médecine légale
FAURE Gilbert	Urologie
GAUTIER Robert	Chirurgie générale
GENSAC Pierre	Botanique
GIDON Maurice	Géologie
GROS Yves	Physiques (IUT A)

MM. GUITTON Jacques	Chimie
HICTER Pierre	Chimie
IVANES Marcel	Electricité
JALBERT Pierre	Histologie
JUNIEN-LAVILLAVROY Claude	O.R.L.
KOLODIE Lucien	Hématologie
LE NOC Pierre	Bactériologie-virologie
LEROY Philippe	IUT A
MACHE Régis	Physiologie végétale
MAGNIN Robert	Hygiène et médecine préventive
MALLION Jean-Michel	Médecine du travail
MARECHAL Jean	Mécanique (IUT A)
MARTIN-BOUYER Michel	Chimie (CUS)
MICHOULIER Jean	Physique (IUT A)
NEGRE Robert	Mécanique (IUT A)
NEMOZ Alain	Thermodynamique
NOUGARET Marcel	Automatique (IUT A)
PARAMELLE Bernard	Pneumologie
PECCOUD François	Analyse (IUT B)
PEFFEN René	Métallurgie (IUT A)
PERRET Jean	Neurologie
PERRIER Guy	Géophysique - Glaciologie
PHELIP Xavier	Rhumatologie
RACHAIL Michel	Médecine interne
RACINET Claude	Gynécologie et obstétrique
RAMBAUD André	Hygiène et hydrologie
RAMBAUD Pierre	Pédiatrie
Mme RENAUDET Jacqueline	Bactériologie
MM. ROBERT Jean-Bernard	Chimie Physique
ROMIER Guy	Mathématiques (IUT B)
SHOM Jean-Claude	Chimie générale
STOEBNER Pierre	Anatomie pathologique
VROUSOS Constantin	Radiologie

MAITRE DE CONFERENCES ASSOCIES

M. COLE Antony	Sciences nucléaires

Fait à SAINT MARTIN D'HERES, AVRIL 1976.

Je remercie,

Monsieur le Professeur Louis Bolliet de me faire l'honneur de présider le jury de Thèse,

Messieurs les Professeur C. Pair, D. Ferrari, S. Krakowiak, J. Sakarovitch de me faire l'honneur d'en faire partie,

Monsieur le Professeur J.P. Verjus d'avoir bien voulu diriger mon travail de thèse et me conseiller tout au long de la préparation et de la rédaction de cet ouvrage,

Monsieur le Professeur E. Gelenbe qui m'a guidé et aidé dans le développement des recherches présentées ici. Il m'a initié aux problèmes posés par l'évaluation des performances des ordinateurs, et ses encouragements amicaux ont permis à ce travail de voir le jour,

Les chercheurs du Laboria qui m'ont aidé à chaque instant de mon travail, et tout particulièrement Jacques Leroudier, Michel Parent, Anne Schroeder, Marc Badel et Didier Merle. Leur compétence scientifique et leur amitié m'ont été un soutien constant,

enfin, Madame Laissus qui, à l'Université de Grenoble, m'a guidé et aidé tout au long de la préparation de la soutenance de cette thèse, et Madame Theis, Madame Poulicet, Mademoiselle Delabarre et Monsieur Mallet qui, à l'IRIA, ont assuré de façon impeccable la réalisation matérielle de ce mémoire.

PREFACE

Il nous paraît utile, en préface aux chapitres de cette thèse consacrée principalement à la modélisation des systèmes informatiques, de situer cette technique d'analyse dans le cadre plus vaste de l'évaluation des performances. Nous présentons dans les pages qui suivent un bilan rapide des directions de recherche en évaluation des performances suivies ces dernières années, bilan qui fera apparaître les relations entre les différentes approches d'évaluation des performances et la part croissante prise par la modélisation dans ces activités.

Domaine encore peu développé il y a dix ans, l'évaluation des performances des systèmes informatiques a progressivement conquis au cours des dernières années son autonomie vis à vis des autres disciplines de l'informatique. Cette évolution a été particulièrement sensible dans le domaine de la recherche, comme en témoignent l'importance de la bibliographie sur le sujet (plus de 400 références dans un article bibliographique récent) la part de plus en plus large consacrée à ce thème dans les revues scientifiques, l'organisation régulière de colloques et de séminaires internationaux spécialisés. Dans les centres de recherche publics et privés cette évolution s'est manifestée par la constitution d'équipes de recherche travaillant de façon spécifique sur le domaine et rassemblant des chercheurs autrefois dispersés entre les groupes de recherche opérationnelle et les équipes de développement de systèmes.

Axes de Recherches en évaluation des performances

L'activité de recherche en évaluation des performances a deux composantes essentielles :

- élaborer des méthodes et des outils de mesure et d'analyse du comportement des systèmes informatiques,

- parvenir à une description et une compréhension de ces comportements aussi claire que possible.

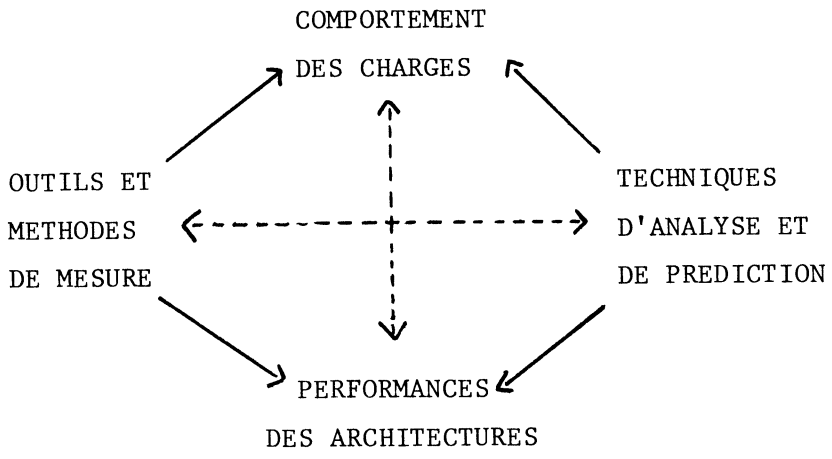
Il s'agit d'une démarche expérimentale semblable à celle du physicien qui, devant un phénomène mal connu, va être amené à construire des instruments de mesure appropriés, à observer le phénomène, à l'analyser et tenter à l'aide de modèles à expliquer et prévoir son comportement.

Le comportement d'un système informatique est déterminé par l'organisation et la dynamique de deux éléments distincts :

- l'architecture du système, que nous définissions comme l'ensemble des ressources en matériel et logiciel et leur organisation,

- la charge du système, c'est-à-dire l'ensemble des traitements soumis au système.

Evaluer un système, c'est donc d'abord caractériser sa charge en étudiant comment les différents traitements font appel aux ressources de l'architecture, ensuite observer et analyser les performances de l'architecture suivant la charge traitée. Distinguer ces deux étapes est indispensable aussi bien lorsqu'il s'agit d'observer et d'améliorer le fonctionnement d'un système en place ou de choisir une nouvelle configuration que lors de la conception d'une nouvelle architecture.



Nous avons représenté (voir figure) les quatre directions que nous venons de dégager. D'une part les outils et les méthodes de mesure, les techniques d'analyse et de prédiction des performances ; d'autre part les deux objets sur lesquels vont porter l'observation et l'analyse : comportements des charges et performances des architectures.

Les relations entre ces quatres approches apparaîtront dans les paragraphes suivants, qui présentent les recherches poursuivies dans ces différentes directions.

Outils et méthode de mesure

La première difficulté rencontrée lors de la mesure des systèmes informatiques est que la plupart d'entre eux n'ont pas été conçus pour être mesurés. Les outils de mesure doivent donc être développés a posteriori avec les limitations imposées par la structure du système et les difficultés d'accès aux points de mesure que cela entraîne.

Les données expérimentales disponibles sur le comportement des architectures et des charges sont relativement peu nombreuses en raison même de l'effort que demande leur collecte. Un accès plus aisé aux paramètres caractéristiques du fonctionnement d'un système passe par le développement dans le système lui-même d'un environnement de mesure permettant au mesureur de construire à la demande un moniteur de mesure répondant à ses besoins particuliers. C'est, par exemple, l'approche développée à l'Université de Californie à Berkeley dans le cadre du système PRIME.

Cette évolution vers l'intégration des outils de mesure est également favorisée par deux exigences : le besoin du système lui-même d'observer et d'analyser son comportement dans le but, comme nous le verrons dans le chapitre I de décider de l'allocation de ses ressources et de réagier dynamiquement aux variations de charge et aux phénomènes de congestion ; la nécessité de disposer d'une comptabilité précise des consommations de ressources. L'instrumentation nécessaire à ces prises de mesure et la définition d'une structure de système

fournissant un environnement de mesure sont des aspects importants à prendre en compte lors de la conception d'une architecture.

Une autre caractéristique de la mesure des systèmes informatiques est la quantité souvent importante des informations recueillies en raison de la rapidité des phénomènes observés. Il est donc essentiel de pouvoir définir des plans d'expériences pour limiter le nombre de mesures, et de savoir déterminer le niveau d'observation en fonction des résultats recherchés. En l'absence d'indications sur le niveau de détail des mesures, on risque en effet de mener une observation trop fine conduisant à un volume de résultats longs et coûteux à traiter et pouvant masquer les phénomènes les plus importants. Construire un plan d'expériences est également nécessaire afin de déterminer le nombre minimum de mesures nécessaires pour répondre aux questions posées. Les méthodes disponibles dans ces domaines sont encore limitées. Si les techniques de planification d'expériences sont largement utilisées dans d'autres applications, par exemple en agronomie, leur emploi pour la définition des campagnes de mesure des systèmes informatiques est encore peu développé.

Pour la détermination du niveau de détail d'une mesure et de grandeur mesurées, les indications les plus utiles sont fournies par l'étude des modèles de systèmes informatiques. En effet, un modèle est une simplification de la réalité, qui en conserve cependant les propriétés les plus importantes. Cette simplification peut donc guider le mesureur dans le choix des grandeurs observées et permettre de définir un ensemble de mesures cohérent. L'évolution dans cette voie se manifeste par un dialogue étroit entre modélisation et mesure, et par la construction d'outils de mesure définis à partir de modèles de prédiction de performances.

Techniques d'analyse et de prédiction des performances

Le premier outil d'analyse disponible pour traiter et tenter d'interpréter les résultats de mesure est l'outil statistique. Son usage est indispensable pour réduire les données de mesures et les présenter sous une forme utilisable, par exemple lorsqu'il s'agit de caractériser les charges aussi bien

par des modèles statistiques de consommation des ressources que par des typologies de comportement. Le développement de cette approche conduit à la mise au point de produits logiciels statistiques spécialisés pour le traitement des données de mesure. L'analyse statistique est une étape intermédiaire indispensable entre la mesure proprement dite et la modélisation, qui requiert le plus souvent des données réduites sous la forme de modèles probabilistes ainsi que l'indiquent les études présentées dans les chapîtres II, III et IV.

La part la plus importante de l'activité de recherche en évaluation des performances ces dernières années a été consacrée aux techniques de modélisation, soit par modèles mathématiques, soit par simulation. La modélisation mathématique par réseaux de files d'attente a donné lieu à un nombre important de travaux, aussi bien sur le plan théorique pour élargir les conditions d'utilisation de ces modèles que sur le plan pratique pour en valider l'application à la prédiction des performances.

Les progrès réalisés ont permis le développement de produits logiciel de modélisation, rendant ainsi ces techniques accessibles à des non-spécialistes, et montré la possibilité de construire des outils de prédiction des performances des systèmes informatiques combinant un outil de mesure et d'analyse de comportement des charges à un modèle global de système. Les recherches en cours doivent permettre une meilleure compréhension des propriétés de ce type de modèles (cf. Chap. I), et de généraliser leurs conditions d'application à l'aide de techniques de résolution approchée dont les principes sont rappelés dans le chapitre I.

Les recherches en simulation, guidées et stimulées par les études menées parallèlement en modélisation, ont principalement porté sur le problème du contrôle et de la précision des résultats fournis par un simulateur, et sur les relations entre modèles mathématiques et modèles de simulation. Comme pour la mise en oeuvre des outils de mesure, les résultats obtenus permettent une simplification des simulations et une plus grande rigueur dans l'utilisation de cette technique. L'opposition entre modèles mathématiques et modèles de

simulation tend à s'estomper, et des modèles de prédiction hybrides réunissant les avantages des deux techniques sont en développement.

D'autres recherches sont nécessaires pour réaliser des aides à la programmation et à la mise au point de simulateurs, et développer des simulateurs hybrides faisant appel à du matériel pour réaliser certaines de leurs fonctions.

Etude du comportement des charges

"Computer programs, in the large, are still relatively mysterious objects, and merit considerably more empirical investigation that has been accorded to date. It seems somewhat strange that our highly sophisticated "scientific" society is spending billions of dollars annualy on machines to process a workload of tasks with essentially unknown characteristics".

Cette constation du Professeur A.W. Batson, de l'Université de Virginie, illustre l'état de nos connaissances sur le comportement des charges et l'importance des travaux menés dans ce domaine. Choisir une configuration ou concevoir une nouvelle architecture n'a en effet de sens que si l'on connaît les traitements qui seront exécutés, et il est facile de constater que l'évolution des architectures va dans le sens d'une meilleure utilisation des propriétés de comportement des programmes.

Le fonctionnement des systèmes à temps partagé (cf. Chap. II), des mémoires-caches, des structures pipe-line ou des architectures à mémoire virtuelle (cf. Chap. IV) reposent directement sur ces propriétés. De nombreux travaux ont été réalisés ces dernières années dans ce domaine en ce qui concerne les caractéristiques des suites d'instructions générées par un programme, des références faites dans l'espace d'adressage, des entrées-sorties. Le but de ces études est de mettre en évidence des lois générales, aussi indépendantes que possible des programmes ou des charges particulières à partir desquelles peuvent être définies des règles d'allocation des ressources. La plus connue est le principe de localité, qui est à la base du fonctionnement des systèmes à mémoire virtuelle, et dont nous présentons les bases expérimentales dans le chapitre IV.

Comprendre comment l'information est référencée et traitée est essentiel pour pouvoir l'organiser et la gérer de façon efficace. Les connaissances actuelles sont encore fragmentaires et certaines propriétés sont mal connues, comme celles des comportements des requêtes dans les grands ensembles de données ou le parallélisme des programmes. De toutes les activités d'évaluation des performances, l'étude du comportement des charges est la plus informatique au sens propre. Elle touche à la nature et à l'organisation de l'information et intéresse directement la recherche en programmation et en structuration des données.

Performances des architectures

La complexité des architecture modernes rend difficile l'estimation de leurs performances suivant le type de charge qui leur est soumis, et l'utilisation d'outils de mesure et d'analyse est indispensable. Mis en oeuvre seuls, les outils de mesures sont insuffisants, car s'ils permettent de diagnostiquer un mauvais fonctionnement, ils fournissent peu d'indications sur les causes d'une efficacité médiocre et sur les modifications qui permettraient d'y remédier. C'est la raison pour laquelle se sont développées les recherches sur la modélisation des performances des architectures. Leur objectif est de dégager des principes de fonctionnement efficace et dans certains cas de prédire les conséquences sur les performances de modifications de l'architecture ou du comportement des charges. Ces modèles sont utilisés pour vérifier l'adéquation d'une configuration ou d'un mécanisme d'allocation de ressources aux charges qu'ils ont à traiter. Les résultats obtenus peuvent non seulement conduire à une mise en cause de l'architecture proposée, mais également montrer comment les performances peuvent être améliorées en agissant sur le comportement des programmes. Dans cette direction se situent les travaux effectués sur la restructuration des programmes pour améliorer l'efficacité des architectures à mémoire virtuelle.

Les études sur les performances d'architecture sont nombreuses et il est intéressant de noter que c'est pour répondre à la demande d'outils d'analyse nécessaires pour mener ces évaluations qu'ont été développées les recherches en modélisation évoquées plus haut. La conception et la mise en

oeuvre d'organisations aussi complexes que les réseaux d'ordinateurs (cf. Chap. III) , les systèmes multi-processeurs ou les architectures à hiérarchie de mémoires (cf. Chap. IV) n'est possible qu'à l'aide de cette approche, et le développement de ces grands projets est largement dépendant du progrès des méthodes d'analyse et de contrôle des performances.

Modélisation et Evaluation des performances

La modélisation mathématique n'existe pas séparée des autres techniques d'évaluation des performances. Au contraire, comme l'indiquent les remarques faites plus haut, la modélisation participe aux autres approches d'évaluation et représente un guide pour leur mise en oeuvre. Elle fournit les bases théorique qui permettent à la fois de simplifier les expériences d'évaluation et de garanti une certaine rigueur dans l'exploitation et l'interprétation des résultats obtenus.

La modélisation est aussi une approche intégrante, en ce sens qu'elle donne un moyen de relier entre eux des résultats isolés, qu'il s'agisse de mesures de performances d'architecture ou de caractérisation de charges, et de disposer ainsi d'une vision globale d'un comportement d'un système, et des interactions entre ses divers composants.

Une telle vision est indispensable dès que l'on cherche non plus seulement à assurer un fonctionnement satisfaisant du système, mais encore à tirer le meilleur parti des ressources disponibles. Contrôler les performances d'un système suppose que l'on ait su d'abord mettre en évidence les facteurs essentiel qui déterminent les performances et les indicateurs permettant de diagnostiquer les variations de ces performances.

Les exemples que nous présentons dans les chapitres qui suivent illustrent cette démarche. Qu'il s'agisse de l'allocation de l'unité centrale dans un système à temps partagé, du partage des mémoires tampons aux noeuds d'un réseau à commutation de paquets ou de la gestion de la multiprogrammation dans les ordinateurs à mémoire virtuelle, la solution au problème posé requiert

que le comportement du système étudié soit compris aussi complètement que possible, d'une part à partir de données expérimentales recueillies pendant son fonctionnement, d'autre part par la construction de modèles permettant de vérifier les hypothèses de comportement et d'en parvenir à une description fidèle.

Parmi les différentes représentations mathématiques utilisables, les files d'attente apparaissent particulièrement bien adaptées à la modélisation des systèmes informatiques. Des représentations plus générales sont certes possibles, mais l'on risque alors de s'éloigner fortement de la réalité de l'objet analysé et de perdre ainsi le bénéfice des propriétés de comportement liées à l'organisation même des architectures des ordinateurs et aux types des traitements exécutés.

Par ailleurs, la complexité des traitements, et l'impossibilité de connaître le plus souvent à priori le déroulement d'une exécution rend nécessaire leur description par des modèles probabilistes. Les programmes apparaissent comme des éléments d'une ou plusieurs populations, ces éléments étant indiscernables les uns des autres à l'intérieur d'une même population et leur comportement défini par plusieurs paramètres considérés comme des variables aléatoires.

La valeur des résultats obtenus à l'aide de cette représentation justifie l'intérêt de cette technique de modélisation. Elle montre que, du point de vue de leurs performances, les ordinateurs se comportent, en dépit de leur complexité interne, comme un système relativement simple défini par un nombre réduit de paramètres. Cette observation indique que les propriétés des modèles à file d'attente, du type de celles discutées dans le chapitre I, sont aussi celles des ordinateurs, avec l'intérêt que cette conséquence présente pour les expériences de mesure et de simulation comme nous l'avons signalé plus haut.

Toutefois ces conclusions ne sont valables que pour le type d'architectures de machine pour lesquelles ce mode de représentation a été utilisé,

et validé, et il serait certainement dangereux de les généraliser à d'autres type d'organisation. Comme dans les autres sciences, chaque problème suscite ses propres outils d'analyse, et l'évolution presque parallèle, depuis dix ans, des techniques de modélisation et des architectures de systèmes le montre très clairement.

De nouveaux problèmes sont posés par le développement des systèmes de gestion de bases de données, des réseaux d'ordinateurs, des systèmes répartis. Dans ces domaines, les résultats expérimentaux et les méthodes d'analyse disponibles sont, dans de nombreux cas, encore embryonnaires. Comme pour les systèmes multiprogrammés il y a dix ans, la maîtrise de ces nouvelles organisations passe par le développement des études expérimentales et le progrès des outils d'analyse.

MODELES A FILES D'ATTENTE

ET GESTION DE RESSOURCES

DANS LES SYSTEMES INFORMATIQUES

Thèse présentée par

Dominique POTIER

Chapitre I

PRECISION ET ROBUSTESSE DES MODELES A FILE D'ATTENTE

Chapitre I

Résumé

Le premier chapitre est consacré à une étude des propriétés de robustesse des modèles à files d'attente utilisés pour l'analyse et la prédiction des performances des systèmes informatiques, et à une présentation des différentes techniques de résolution de ces modèles.

Nous montrons en introduction que les écarts observés entre les résultats fournis par un modèle et les mesures effectuées sur le système réel ont deux causes distinctes. La première cause de ces écarts est liée aux simplifications qui sont introduites lors de la construction et de l'analyse du modèle. La sensibilité des résultats obtenus à ces simplifications définit les propriétés de robustesse du modèle. La seconde raison tient aux erreurs introduites par la technique de résolution lorsqu'il ne s'agit pas d'une résolution exacte, mais d'une technique approchée.

Dans la première partie du chapitre nous décrivons les principales techniques de résolution approchées utilisées pour ces modèles, depuis les techniques les plus générales, comme la simulation, jusqu'aux méthodes les plus dépendantes des propriétés du modèle analysé. A partir des résultats d'expériences et des études de précision publiés dans la littérature nous indiquons la précision obtenue pour chacune de ces approches.

Nous présentons dans la seconde partie du chapitre trois études de propriétés de robustesse de modèles à files d'attente par rapport aux hypothèses faites sur la nature de la source, la distribution des temps de service, et la capacité de la file d'attente. Dans le premier cas, il s'agit d'évaluer dans quelles conditions l'approximation qui consiste à remplacer une source finie, comme c'est le cas pour un système réel, par une source infinie est acceptable ;

dans le second cas d'examiner les conséquences sur la qualité des résultats obtenus des hypothèses exponentielles sur la distribution des temps de service ; dans le troisième cas d'analyser l'effet sur le fonctionnement d'un système à file d'attente des phénomènes de blocage provoqués par la capacité limitée de certaines files.

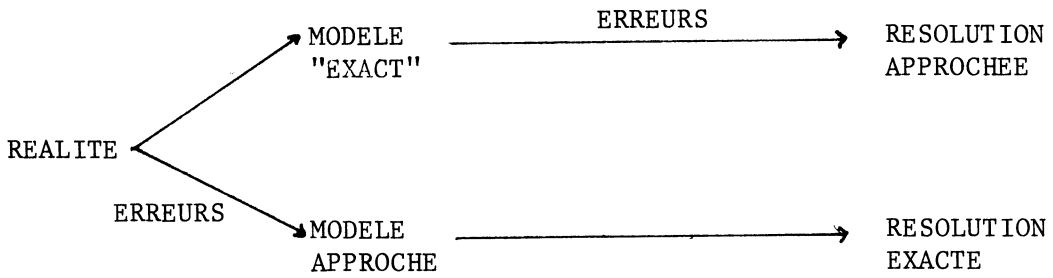
En conclusion, nous montrons quelles indications peuvent fournir les résultats obtenus pour la modélisation des systèmes informatiques, et nous indiquons à partir d'exemples comment l'effet de certaines hypothèses de modélisation très largement utilisées est encore mal connu et demande une étude à la fois théorique et expérimentale.

1. - INTRODUCTION

Toute démarche de modélisation est conduite en fonction de deux objectifs essentiels : comprendre et prédire. La compréhension du phénomène modélisé est obtenue au cours de la phase de construction et de mise au point du modèle pendant laquelle sont analysés les aspects principaux du phénomène et sont dégagées les lois qui décrivent son comportement. Une fois construit et validé le modèle est utilisé pour simuler le phénomène étudié et en prédire le comportement dans différentes situations.

Au cours de ces deux phases interviennent obligatoirement des simplifications et des approximations soit dans la définition du modèle, soit dans sa résolution. On peut schématiser cette situation par les deux cas extrêmes représentés sur la figure ci-dessous. Si l'on construit un modèle "exact", c'est à dire où un minimum de simplifications a été introduit, la résolution exacte de ce modèle est en général impossible et on est amené à utiliser une technique de résolution approchée (simulation, approximation par diffusion, méthode itérative, etc...)

Si en revanche le modèle construit est suffisamment simple, une résolution exacte est possible.Dans le premier cas l'écart que l'on pourra constater entre les prédictions du modèle et la réalité proviendra de la méthode de résolution, dans le second des simplifications introduites dans le modèle. Le problème posé est celui de la précision d'une résolution pour les écarts du premier type , de la précision des modèles ou leur robustesse, pour les écarts du second type. En général un bon modèle sera obtenu par un compromis entre la précision de la résolution et la précision du modèle, et il est donc important de distinguer ces deux niveaux d'erreurs afin de connaître le niveau de modélisation le plus satisfaisant en fonction des résultats recherchés.



Cette distinction entre ces deux types d'écarts est évidemment idéale, particulièrement parce que de nombreuses méthodes approchées sont fondées sur des propriétés de robustesse des modèles : c'est le cas, par exemple, des techniques de décomposition des réseaux de files d'attente dont l'application dépend directement de la structure du modèle utilisé. En revanche d'autres techniques de résolution approchée comme la simulation, la résolution numérique directe des modèles markoviens sont davantage indépendantes des propriétés intrinsèques des modèles considérés.

Dans la suite de ce chapitre, nous limitons l'étude des erreurs aux modèles probabilistes basés sur la théorie des files d'attente qui sont largement utilisées pour l'analyse des performances des systèmes informatiques Nous présentons d'abord les techniques de résolution approchée en débutant par les techniques les plus générales pour terminer par les techniques les plus dépendantes des propriétés intrinsèques des modèles, c'est à dire, dans l'ordre : la simulation à évènements discrets, les méthodes numériques d'analyse des systèmes markoviens, l'approximation par diffusion, les méthodes itératives, les méthodes de décomposition. Dans le paragraphe suivant nous présentons une étude des propriétés de robustesse des modèles pour les hypothèses de nature de la source,de distribution, de capacité des files. Nous concluons à partir des résultats de ces deux paragraphes.

2. - METHODES DE RESOLUTION APPROCHEE

2.1 Simulation

La méthode approchée la plus couramment utilisée pour résoudre un modèle de façon approchée est la simulation à évènements discrets [25] qui consiste à reproduire la séquence des évènements se produisant dans le système étudié telle qu'elle est décrite par le modèle et à prendre des mesures au cours de la simulation pour estimer les quantités recherchées. Dans le cas de simulation d'un modèle à files d'attente le problème est donc d'estimer les propriétés des processus stochastiques sous-jacents au modèle (nombre de clients dans une file d'attente, activité du serveur, etc...) à partir d'une réalisation unique de ces processus obtenue par la simulation. Pour chaque grandeur étudiée, on va donc construire un intervalle de

confiance -ce qui permettra de contrôler la précision de la simulation- et tenter de réduire la largeur de cet intervalle afin d'améliorer la précision, par exemple en augmentant la durée de la simulation ou en utilisant un estimateur de faible variance. L'ensemble de ces techniques est exposé dans [25] et nous n'irons pas plus avant dans leur description. Lorsque des techniques de contrôle et d'amélioration de la précision sont mises en oeuvre on peut atteindre une précision de l'ordre de 1 à 5 %, comme dans [1]. En l'absence de ces précautions la précision obtenue peut-être médiocre.

2.2 Résolution numérique des modèles markoviens

On sait que la solution stationnaire de modèles markoviens est obtenue à partir de l'information contenue dans les valeurs propres et les vecteurs propres de la matrice stochastique du modèle. Les méthodes de résolution numérique des modèles markoviens consistent donc à construire la matrice stochastique associée au modèle, à calculer les valeurs et vecteurs propres, et à les traiter pour en déduire le comportement du système étudié. L'intérêt de cette approche est sa généralité qui permet une automatisation du calcul à partir de la topologie du modèle et des caractéristiques de ses éléments [37,38]. Les difficultés sont rencontrées au niveau de la convergence numérique de la solution qui peut être longue, particulièrement si la matrice du système est presque décomposable. La mise en oeuvre d'algorithmes non standard comme dans l'analyseur de Stewart [37] permet de remédier en partie a cet inconvénient. Il convient de remarquer que l'utilisation de ces outils et l'interprétation des résultats requièrent une certaine familiarité avec les propriétés des matrices stochastiques et de leurs valeurs et vecteurs propres.

2.3. Approximation par diffusion.

L'utilisation de la théorie des chaînes de Markov à temps continu et à état continu pour étudier des chaînes de Markov à temps continu et à état discret est connue sous le nom d'approximation par diffusion. L'intérêt principal de cette technique vient de ce que les methodes analytiques sont plus développées pour les systèmes à domaine d'état continu que pour les systèmes à domaines d'état discret. Son inconvénient est qu'il s'agit

comme le nom l'indique d'une approximation. Comme noté dans [16] l'approximation d'un modèle par un processus de diffusion se fait en plusieurs étapes : (1) détermination des paramètres de la diffusion, (2) traitement des conditions aux limites, (3) choix de la discrétisation permettant de revenir du continu au discret. Les hypothèses retenues à chacune de ces étapes affectent la précision des résultats. En particulier, le calcul des paramètres de la diffusion qui repose sur l'application du théorème central limite suppose que le système étudié est fortement chargé.

Les premiers modèles de systèmes informatiques traités par l'approximation par diffusion ont été présentés par Gaver et Shedler à partir d'une étude de problèmes de congestion due à Gaver [10] où sont comparés les résultats obtenus sur un système M/G/1 soit par approximation, soit exactement. Dans [11] Gaver utilise l'approximation par diffusion pour analyser le comportement d'un ensemble de terminaux, et suggère que l'approximation par diffusion pourrait être utilisé comme complément, ou comme remplacement de la simulation. Un modèle cyclique de système multiprogrammé est étudié par Gaver et Shedler en [12] pour estimer l'utilisation de l'unité centrale. Les résultats de cette analyse et de l'analyse exacte du modèle [9] sont comparés, et le modèle approché est modifié dans [13] pour tenir compte des résultats de comparaison. Dans tous ces modèles le traitement des conditions aux limites est réalisé par la méthode de la barrière réfléchissante développée par Newell et Gaver [9]. C'est également l'approche suivie par Kobayashi [21,22] qui généralise l'application de l'approximation par diffusion à un réseau général de files d'attente. Le problème de la précision de cette technique d'approximation par diffusion a été examiné par Reiser et Kobayashi [34] en comparant les résultats approchés avec ceux obtenus soit de façon exacte, soit par simulatic lorsque la solution explicite n'est pas disponible. Leurs conclusions sont les suivantes :

- la précision est maximum lorsque la distribution du temps de service est exponentielle (c = 1), et les erreurs augmentent avec le coefficient de variation c de cette distribution,

- les erreurs tendent vers zéro lorsque les utilisations sont proches de l'unité. C'est une conséquence de l'utilisation du théorème central limite pour le calcul des paramètres de la diffusion,

- dans tous les exemples étudiés où les distributions ne sont pas exponentielles, l'approximation par diffusion donne une moyenne et une variance de la longueur de la file d'attente largement plus réaliste que celles obtenues par un modèle avec serveur exponentiel. La moyenne de la longueur de la file d'attente tend à être sous-estimée pour $c > 1$.

- pour des réseaux à files d'attente ouverts, le traitement conduit en considérant que chaque serveur peut être analysé séparément donne des résultats satisfaisants.

- pour des réseaux fermés avec un faible nombre de clients, l'utilisation des serveurs n'est pas estimée correctement, et l'approximation exponentielle donne de meilleurs résultats. Cette limitation provient essentiellement du traitement des conditions aux limites par la méthode de la barrière réfléchissante.

Afin de remédier à cette limitation Gelenbe introduit dans [16] les équations d'un processus de saut instantané pour décrire le comportement au point zéro. Cette méthode fournit exactement la probabilité que la file soit vide dans le cas du système M/G/1 et la même technique est utilisée dans [15] pour résoudre un système G1/G/1/ et G1/G/1/N. L'approximation obtenue sur l'utilisation des serveurs et la distribution de la longueur des files d'attente avec cette approche dans le cas du système GI/G/1/N a été étudiée par Badel et Shum [1] par simulation. Les résultats indiquent que la précision de l'approximation par diffusion est du même ordre de grandeur que la précision de la simulation, et décroît avec le coefficient de variation des distributions. En revanche, les résultats sont moins satisfaisants pour la longueur des files d'attente où la précision est largement dépendante du mode de discrétisation utilisé sans qu'il soit possible de dégager une technique de discrétisation généralisable à des réseaux de files d'attente.

2.4 Méthodes itératives

Le théorème de Norton indique que dans un réseau exponentiel fermé du type Jackson il existe, pour chaque station i, une station complémentaire B_i telle que la loi de probabilité du nombre de clients dans la station i est la même dans le réseau complet que dans le réseau à deux stations (i, B_i). B_i est le complément de la station i et son taux de service $r_i(n)$

peut être calculé simplement. Le calcul des stations complémentaires donne la solution du réseau complet. Dans le cas d'un réseau général, on procède par itération en ajustant à chaque itération les taux de service des stations i afin de satisfaire les équations de conservation de flux du réseau et la conservation du nombre total de clients. Cette méthode a été validée par Chandy [6] sur plusieurs exemples de réseaux de files d'attente avec distribution de service général et stations gérées suivant la politique PAPS, et des améliorations des algorithmes de Chandy sont présentées dans [4].

Les résultats cités dans les deux études référencées sont obtenus avec une précision de l'ordre de 5 %,aussi bien en ce qui concerne les taux d'utilisation des serveurs que les premiers moments des temps d'attente. Le nombre d'itérations nécessaire est faible avec un nombre moyen de quatre pour les résultats de [6]. Ces algorithmes de calcul se généralisent aux cas de plusieurs classes de clients et au cas où les taux de service dépendent de l'état du système.

2.5 Décomposition des réseaux de files d'attente

Le principe de décomposition des réseaux de file d'attente a son origine dans les techniques d'aggrégation largement utilisées en économie. Ces techniques sont fondées sur la remarque que dans la plupart des grands systèmes où interviennent de nombreuses variables, ces variables peuvent être rassemblées en un nombre limité de groupes tel que :

- les intéractions entre groupes peuvent être analysées indépendamment des intéractions au sein des groupes.

- les intéractions au sein des groupes peuvent être analysées indépendamment des intéractions entre groupes.

Si ces propriétés sont vérifiées le système est dit quasi-décomposable et Simon et Ando [36] ont montré que l'on pouvait distinguer le fonctionnement du système à court terme du fonctionnement à long terme de la façon suivante :

- dans le comportement à court terme, un équilibre local est atteint à l'intérieur de chaque sous-système indépendamment des autres sous-systèmes.

- dans le comportement à long terme, un équilibre global

Ces résultats ont été étendus et formalisés aux cas de modèles de systèmes informatiques à réseaux de files d'attente par Courtois [8]. Ils permettent d'appliquer les théorèmes de Jackson et de BCMP à des réseaux à files d'attente où existent des dépendances entre les stations si celles-ci sont faibles. C'est le cas dans les modèles de système multiprogrammé exploité à partir d'un ensemble de terminaux où le comportement des taches à l'intérieur du système peut être analysé indépendamment du processus de génération des taches par les terminaux. Cet exemple est étudié dans [8] ainsi que dans [17b] dans le cadre de l'analyse de politiques de régulation du degré de multiprogrammation.

2.6 Conclusion

Nous avons tenté de présenter dans ce paragraphe une description aussi complète que possible des méthodes de résolution approchée des modèles à file d'attente. Comme nous l'avons vu, ces techniques sont diverses, font appel à des principes très différents, et dans la grande majorité des cas demandent une mise en oeuvre et une mise au point particulière pour chaque application d'autant plus que la généralité de la méthode est grande et qu'elle s'appuie moins sur des propriétés intrinsèques du modèle.
Les résultats de précision sont encore fragmentaires, et il paraît indispensable pour la mise en oeuvre de ces méthodes que la précision soit contrôlée systématiquement. C'est dire la nécessité d'utiliser pour une même résolution plusieurs techniques concurremment.

3. Robustesse des modèles

La notion de robustesse et son étude sont récentes, et se sont développées après les premières expériences de modélisation par files d'attente de systèmes informatiques vers le milieu des années soixante. Ce n'est que dans ces dernières années où un nombre important de résultats originaux sur les propriétés des modèles à files d'attente ont été obtenus [2,3,7,21] et qu'une compréhension intime de leur comportement a été atteinte que les propriétés de robustesse ont été mises en évidence. Il convient de remarquer que parmi ces résultats plusieurs ont été obtenus pour répondre aux problèmes posés par l'analyse et la prédiction des performances des systèmes informatiques.

Les propriétés de robustesse d'un modèle s'apprécient par rapport à chacune des hypothèses intervenant dans le modèle : on dira ainsi qu'un modèle est robuste par rapport à une hypothèse de distribution si le fait de ne pas satisfaire l'hypothèse n'entraîne pas de modifications sensibles des résultats. De même la robustesse par rapport à une hypothèse n'a de sens que pour un type de résultat donné : comme nous le verrons, un modèle peut être robuste par rapport à une hypothèse donnée pour une certaine mesure de performance, mais beaucoup moins robuste pour une autre. C'est la connaissance de ces différences de comportement qui permettra d'utiliser un modèle dans de bonnes conditions.

Le mode de représentation d'un système par un réseau de files d'attente implique plusieurs hypothèses, liées soit à la structure même de ce type de modèle, soit aux contraintes imposées pour l'obtention d'une solution explicite aussi simple que possible. Nous examinerons dans ce paragraphe les propriétés de robustesse de différents modèles à files d'attente par rapport aux hypothèses faites sur les trois points suivants :

- nature de la source
- distribution des temps de service
- capacité des files d'attente

En l'absence de résultats théoriques, la méthode d'étude suivie consiste pour chaque cas à comparer les résultats obtenus à partir du modèle "exact", c'est à dire le modèle où l'hypothèse simplificatrice considérée n'est pas faite, à ceux fournis par le modèle approché correspondant.

3.1 Robustesse par rapport à la nature de la source

L'importance de cette hypothèse intervient principalement dans les modèles de système à temps partagé utilisés pour l'analyse de différents mécanisme d'allocations de l'unité centrale. Dans un tel système, la source des programmes est essentiellement de nature finie, puisque, si N désigne le nombre de télétypes connectés à l'ordinateur, au plus N programmes peuvent être présents dans l'ordinateur à un instant donné.

Toutefois, pour des raisons que nous allons préciser, l'étude de tels systèmes a été principalement conduite à l'aide de modèles à source infinie et typiquement à l'aide du modèle classique M/G/1. Il convient donc de préciser si les erreurs associées à cette approximation sont suffisamment petites pour permettre d'utiliser le modèle M/G/1 et, sinon, de calculer les conditions de validité de l'approximation. Cette étude a été faite par Buzen et Goldberg [5], et nous en rapportons ici le principe et les résultats.

Si nous représentons simplement comme sur la figure 1 le système étudié avec l'hypothèse que les temps de réflexion des usagers aux télétypes sont indépendants et distribués suivant une loi exponentielle de paramètre λ, le taux d'arrivée des programmes dépend du nombre de programmes n dans le système et s'annule pour n = N. Toutefois si N est grand, et si le nombre moyen de programmes en attente reste limité, le taux d'arrivée des programmes variera peu. Dans une telle situation on peut alors supposer que le taux d'arrivée moyen des clients reste constant, indépendamment du nombre de programmes dans le système.

C'est l'approximation qui est faite dans les modèles à source infinie. Le modèle à source infinie peut donc être considéré comme une limite du modèle à source finie lorsque N tend vers l'infini. En faisant l'hypothèse que les durées de calcul des programmes sont des variables aléatoires indépendantes et identiquement distribuées suivant une distribution donnée, le modèle à source infinie et le modèle à source finie correspondent aux modèles à file d'attente classiques M/G/1 et M/G/1/N.

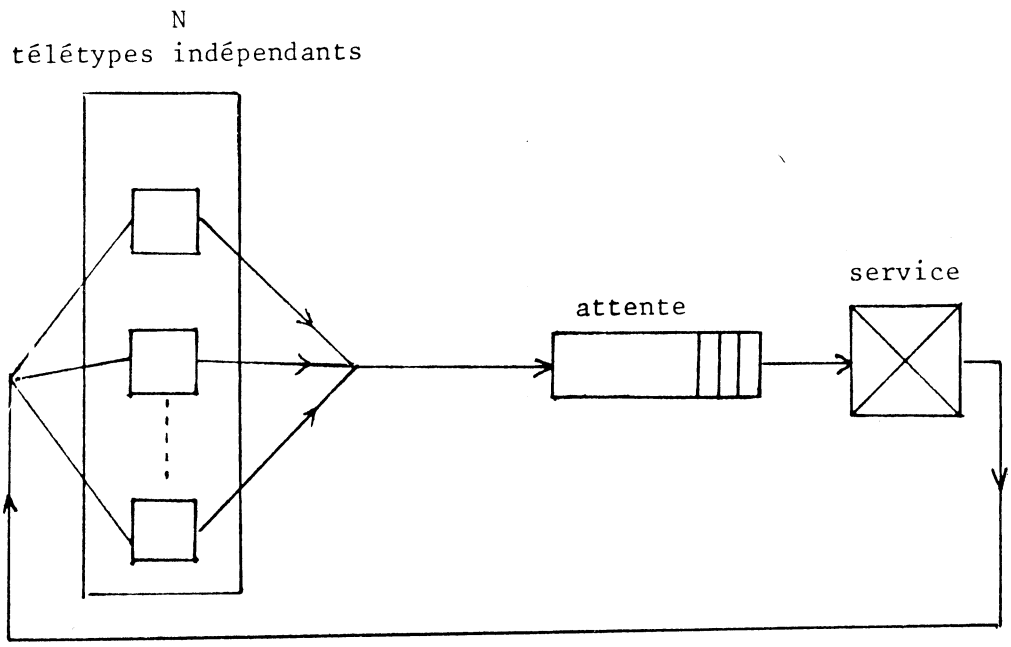


Figure 1 : Modèle à source finie

Bien que l'on puisse résoudre explicitement les modèles M/G/1/N dans un certain nombre de cas, et calculer les quantités recherchées, il y a plusieurs raisons pour préférer le modèle à source infinie M/G/1.

La première raison est la simplicité mathématique de tels modèles : il suffit de comparer les expressions du temps de réponse R et R_N pour les modèles M/G/1 et M/G/1/N données par les équations (1) et (2).

(1) $$R = \frac{1}{\mu} + \frac{\rho\ (1+c^2)}{2\mu\ (1-\rho)}$$

où

μ = inverse du temps moyen de calcul

ρ = facteur de charge, c'est à dire le rapport λ/μ,

c = coefficient de variation de la distribution de temps de calcul.

(2) $$R_N = \frac{N}{\mu} - \frac{1}{\lambda} + \left[\lambda \sum_{r=o}^{N-1} \binom{N-r}{r} \frac{1}{H_r} \right]^{-1}$$

où

μ = inverse du temps moyen de calcul,

λ = inverse du temps moyen de réflexion,

N = nombre de sources.

$$H_r = \begin{cases} 1 & \text{si} \quad r=o \\ \prod_{i=1}^{r} \varphi(i\lambda)/(1-\varphi(i\lambda)) & , \ r = 1,2 \ \ldots \ N-1 \end{cases}$$

$\varphi(s)$ = transformée de Laplace de la distribution de temps de calcul.

Cette complexité se retrouve dans la plus grande partie des modèles à source finie, et fait qu' un certain nombre de problèmes résolus pour une source infini n'ont pas encore trouvé de solution dans le cas fini : nous verrons que c'est le cas pour l'analyse du mécanisme d'adaptation.

Deux autres raisons, citées par Buzen[5] , interviennent également : il est plus facile de mesurer un taux moyen d'arrivée sur le système plutôt que

d'évaluer les temps de réflexion aux télétypes qui peuvent être éloignés du système et se comporter différemment ; le modèle à source fini introduit un paramètre supplémentaire N, qu'il est inutile de spécifier dans les modèles à source infinie.

Pour étudier la valeur de l'approximation considérons un système réel qui soit représenté exactement par le modèle M/G/1/N, et supposons que nous analysions ce système à l'aide d'un modèle M/G/1. En supposant la même distribution du temps de calcul et le même facteur de charge ρ sur chacun des modèles, on peut calculer les temps de réponse exacte R_N et approcher R et en déduire les erreurs $(R-R_N)/R_N$. Les résultats sont rassemblés sur les figures 2a, b, c, d pour les distributions de temps de calcul constantes, Erlangienne, exponentielles et hyperexponentielles. La figure 3 donne le nombre de sources minimum pour avoir une erreur inférieure à 5 %.

	ρ								
N	.1	.2	.3	.4	.5	.6	.7	.8	.9
1	.059	.127	.215	.332	.508	.742	1.172	2.031	4.375
2	.030	.068	.117	.188	.296	.456	.725	1.280	2.752
3	.020	.046	.082	.135	.213	.344	.541	.995	2.239
4	.014	.036	.066	.105	.171	.276	.455	.819	1.865
5	.011	.028	053	.090	.142	.228	.382	.684	1.590
6	.010	.024	.045	.076	.124	.199	.329	.597	1.480
7	.008	.021	.038	.065	.108	.173	.297	.569	1.293
8	.007	.018	.034	.059	.095	.162	.267	.502	1.226
9	.006	.016	.030	.051	.088	.141	.248	.466	1.132
10	.006	.015	.028	.047	.078	.133	.230	.430	1.021
20	.003	.008	.014	.025	.042	.074	.134	.262	.716
30	.002	.005	.009	.017	.028	.052	.094	.202	.587
40	.001	.004	.007	.013	.022	.039	.073	.156	.436
50	.001	.003	.006	.010	.018	.032	.062	.139	.392
60	.001	.003	.005	.008	.015	.026	.054	.116	.349
70	.001	.002	.004	.007	.013	.023	.045	.098	.339
80	.000	.002	.003	.006	.011	.020	.040	.090	.297
90	.000	.001	.003	.006	.010	.018	.036	.083	.258
100	.000	.001	.003	.005	.009	.016	.033	.077	.250
120	.000	.001	.002	.004	.008	.014	.028	.068	.235
140	.000	.000	.002	.003	.006	.012	.023	.055	.196
160	.000	.000	.002	.003	.005	.010	.020	.050	.185
180	.000	.000	.001	.002	.005	.009	.018	.045	.153
200	.000	.000	.001	.002	.004	.008	.016	.042	.145

Figure 2a : erreurs relatives pour des temps de service constants.

	ρ								
N	.1	.2	.3	.4	.5	.6	.7	.8	.9
1	.117	.254	.430	.664	1.016	1.484	2.344	4.063	8.750
2	.059	.135	.237	.379	.588	.894	1.348	2.387	5.339
3	.041	.093	.168	.274	.411	.641	1.007	1.764	3.794
4	.028	.073	.131	.217	.334	.524	.809	1.453	3.159
5	.023	.060	.105	.175	.274	.443	.696	1.240	2.672
6	.019	.048	.091	.149	.240	.382	.615	1.085	2.461
7	.017	.042	.077	.134	.212	.338	.566	1.006	2.145
8	.015	.037	.068	.116	.189	.308	.492	.919	1.935
9	.013	.033	.062	.106	.169	.273	.475	.829	1.860
10	.012	.030	.057	.098	.158	.259	.433	.782	1.762
20	.006	.016	.029	.050	.088	.148	.255	.473	1.157
30	.004	.011	.020	.035	.060	.102	.189	.370	.950
40	.003	.008	.015	.027	.045	.082	.151	.302	.786
50	.002	.006	.012	.022	.037	.065	.124	.260	.699
60	.002	.005	.010	.018	.031	.056	.104	.211	.608
70	.002	.005	.009	.015	.027	.049	.093	.195	.519
80	.001	.004	.008	.013	.024	.044	.084	.182	.510
90	.001	.004	.007	.012	.022	.038	.077	.159	.461
100	.001	.003	.006	.011	.020	.034	.066	.149	.415
120	.001	.003	.005	.009	.017	.029	.057	.124	.399
140	.001	.002	.004	.008	.014	.025	.050	.112	.348
160	.001	.002	.004	.007	.013	.022	.045	.102	.302
180	.001	.002	.003	.006	.011	.020	.041	.094	.290
200	.001	.002	.003	.006	.010	.018	.035	.087	.279

Figure 2c : erreurs relatives pour des temps de service exponentiels

	ρ							
Distribution	.1	.2	.3	.4	.5	.6	.7	.8
Constant	2	3	6	10	17	32	61	160
Erlang-2	2	5	9	16	27	49	101	200+
Exponential	3	6	12	21	38	70	142	200+
Hyperexponential	4	8	16	27	48	93	188	200+

Figure 3

N	ρ .1	.2	.3	.4	.5	.6	.7	.8	.9
1	.088	.190	.322	.498	.762	1.113	1.758	3.047	6.563
2	.044	.102	.180	.290	.442	.674	1.061	1.825	4.012
3	.031	.070	.127	.204	.321	.495	.789	1.345	3.014
4	.021	.054	.098	.160	.253	.402	.650	1.140	2.591
5	.017	.045	.079	.133	.213	.336	.556	.946	2.217
6	.014	.036	.068	.113	.180	.296	.487	.886	1.978
7	.012	.031	.057	.097	.164	.260	.421	.772	1.812
8	.011	.028	.051	.088	.140	.229	.383	.735	1.590
9	.010	.025	.046	.080	.129	.216	.358	.653	1.505
10	.009	.022	.042	.071	.121	.197	.334	.610	1.398
20	.004	.012	.021	.037	.063	.109	.196	.372	.945
30	.003	.008	.015	.026	.045	.078	.142	.298	.732
40	.002	.006	.011	.020	.033	.059	.112	.238	.657
50	.002	.005	.009	.015	.027	.049	.091	.189	.571
60	.002	.004	.007	.013	.023	.042	.080	.172	.483
70	.001	.003	.006	.011	.020	.035	.071	.148	.436
80	.001	.003	.006	.010	.018	.031	.060	.137	.390
90	.001	.003	.005	.009	.016	.028	.055	.127	.381
100	.001	.002	.005	.008	.014	.025	.050	.110	.338
120	.001	.002	.004	.007	.012	.021	.043	.097	.290
140	.001	.002	.003	.006	.010	.019	.038	.087	.275
160	.001	.001	.003	.005	.009	.016	.031	.080	.263
180	.001	.001	.003	.004	.008	.015	.028	.067	.224
200	.000	.001	.002	.004	.007	.013	.026	.062	.214

Figure 2b : erreurs relatives pour des temps de service exponentiels

N	ρ .1	.2	.3	.4	.5	.6	.7	.8	.9
1	.147	.317	.537	.830	1.270	1.856	2.930	5.078	10.938
2	.074	.170	.306	.469	.720	1.080	1.666	2.921	6.366
3	.052	.117	.210	.340	.526	.798	1.249	2.175	4.598
4	.036	.092	.164	.270	.420	.655	1.013	1.725	3.932
5	.029	.076	.139	.219	.356	.574	.880	1.483	3.443
6	.024	.060	.115	.194	.305	.472	.752	1.310	3.071
7	.021	.053	.102	.168	.262	.420	.694	1.220	2.570
8	.018	.047	.087	.146	.242	.373	.608	1.124	2.357
9	.016	.042	.079	.134	.218	.353	.558	1.025	2.289
10	.015	.038	.072	.124	.196	.315	.540	.924	2.194
20	.008	.020	.037	.065	.111	.183	.318	.581	1.394
30	.005	.014	.025	.045	.077	.134	.239	.444	1.088
40	.004	.010	.019	.035	.061	.103	.185	.367	.922
50	.003	.008	.016	.028	.048	.087	.153	.303	.835
60	.003	.007	.013	.023	.041	.072	.135	.280	.739
70	.002	.006	.011	.020	.035	.063	.115	.246	.643
80	.002	.005	.010	.017	.031	.056	.104	.216	.592
90	.002	.005	.009	.016	.028	.051	.096	.202	.542
100	.002	.004	.008	.014	.025	.044	.088	.191	.533
120	.001	.003	.007	.012	.021	.038	.072	.160	.437
140	.001	.003	.006	.010	.018	.033	.064	.135	.422
160	.001	.002	.005	.009	.016	.029	.057	.124	.371
180	.001	.002	.004	.008	.015	.026	.052	.115	.325
200	.001	.002	.004	.007	.013	.024	.047	.106	.313

Figure 2d : erreurs relatives pour des temps de service hyperexponentiels.

Les résultats mettent en évidence la sensibilité des erreurs commises en utilisant le modèle M/G/1 au facteur de charge ρ et on peut en déduire que l'hypothèse de source infinie est discutable quand ρ est grand. Dans la discussion des résultats obtenus à partir de modèle à source infinie, il convient donc d'être particulièrement prudent sur les conclusions qui peuvent être formulées pour les valeurs importantes de ρ.

3.2 Robustesse par rapport à la distribution de temps de service

En dépit de résultats récents sur la solution des réseaux de files d'attente avec des hypothèses élargies [2,7] , l'hypothèse exponentielle reste nécessaire pour le calcul exact de la solution dans de nombreux exemples, et en particulier chaque fois que la file d'attente considérée est gérée suivant la politique PAPS. Nous examinerons dans ce paragraphe la robustesse des modèles à files d'attente par rapport à cette hypothèse, d'une part pour un système fermé où les serveurs sont gérés suivant la règle PAPS, d'autre part pour un système ouvert :

i) lorsque la politique de service est la règle du "round-robin",

ii) lorsque la charge du serveur est proche de l'unité.

3.2.1. - Cas d'un système fermé

A - Résultats pour un réseau simple à deux serveurs

Nous étudions l'approximation qui est réalisée lorsqu'une file d'attente M/M/1 à capacité finie est utilisée pour prédire les paramètres de la distribution d'équilibre du temps de réponse d'une file d'attente M/G/1 à capacité finie.

a) Le modèle M/G/1 à capacité finie

La structure du modèle est représentée sur la figure 4. Il peut être considéré comme un modèle simple d'un ordinateur multi-programmé où un nombre fixe N de programmes sont exécutés successivement par l'unité centrale représentée par la station 1, et par une unité d'entrée-sortie représentée par la station 2. Plus généralement, un tel modèle peut être considéré comme un exemple simple de réseau fermé de files d'attente. Nous supposons que les temps de service à la station sont indépendants et identiquements distribués suivant une loi exponentielle de paramètre λ, et que les temps de service de la station 2 sont indépendants et identiquement distribués suivant une loi générale F(t) de moyenne E(T) et de variance Var(T). Etant donné ces hypothèses, la distribution d'équilibre du temps d'attente q(N) à la station 2 peut être calculée [23]. Les deux premiers moments de q(N) et le principe de la démonstration sont donnés dans l'Annexe A.

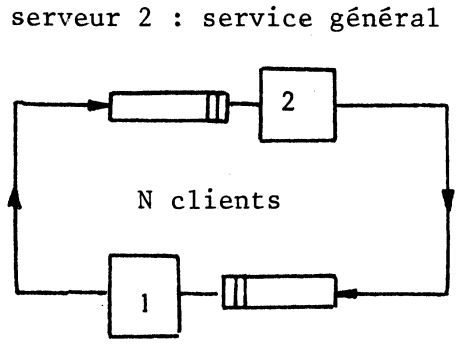


Figure 4 : le modèle M/G/1 à capacité limitée.

b) Le modèle approché M/M/1 à capacité finie

La structure du modèle M/M/1 à capacité finie est semblable à celle du modèle précédent, mais dans ce modèle la distribution de temps de service à la station 2 est supposée exponentielle. Ainsi, le modèle M/M/1 peut-il être considéré comme un modèle approché de la file d'attente M/G/1 qui représente plus exactement le système réel. En raison de l'hypothèse exponentielle, plusieurs avantages sont liés au modèle M/M/1. Le principal est la simplicité mathématique de solution qui autorise une compréhension et un calcul des résultats plus simple que dans le cas M/G/1.

Un autre avantage est qu'il suffit de caractériser la station 2 par un seul paramètre - son temps moyen de service - plutôt que par au moins deux paramètres - le temps moyen de service et sa variance. Cet aspect est particulièrement important lorsque la distribution réelle de temps de service est mal connu.

c) Valeur de l'approximation exponentielle

Afin de tirer au mieux parti du modèle M/M/1, il convient d'évaluer la valeur de l'approximation obtenue quand ce modèle est utilisé pour prédire les performances de systèmesréels où les distributions ne

sont pas exponentielles. Nous étudierons cette question en suivant l'approche utilisée dans l'étude précédente. Nous supposons qu'un système réel qui se comporte exactement comme la file d'attente M/G/1 à capacité finie est analysé à l'aide du modèle M/M/1 à capacité finie. Les deux modèles pouvant être résolus explicitement, il est possible de calculer à la fois les valeurs exactes de la moyenne E, de la variance V et du coefficient de variation C du temps de réponse r(N), et les valeurs approchées correspondantes $\bar{E}$, $\bar{V}$, et $\bar{C}$, d'obtenir ainsi les termes d'erreur $(\bar{E} - E)/E$, $(\bar{V} - V)/V$ et $(\bar{C} - C)/C$.

Toutefois, avant de conduire ce calcul, il est nécessair de préciser la façon dont les deux modèles sont calibrés l'un par rapport à l'autre. Deux approches sont possibles suivant le paramètre choisi pour réaliser la calibration. Une première calibration peut être faite de façon à ce que les utilisations du serveur 2 dans le modèle approché et le système réel soient identiques, et sera notée calibration A. Cette technique apparait réaliste car l'utilisation est un paramètre relativement simple à mesurer, et étroitement lié au comportement du système. Le second mode de calibration, noté calibration B, consiste à rendre égal dans les deux modèles les rapports $\lambda/E(T)$, et, dans ce cas, les termes d'erreur sur l'utilisation du serveur 2 $(\bar{U} - U)/U$ sont également calculés. Remarquons qu'il est important de savoir quelle technique de calibration conduit aux erreurs les plus petites afin de savoir quel paramètre il convient de mesurer sur le système réel pour obtenir le meilleur résultat de prédiction à partir du modèle.

d) Résultats numériques et conclusions

Les termes d'erreur sont tracés en fonction de la charge du système mesuré par l'utilisation U du serveur 2 sur les figures 5, 6, 7 et 8 pour N variant de 2 à 9.

Pour N = 1, les erreurs sur E [r(N)] sont évidemment nulles quelque soit U. Les distributions de service au serveur 2 considérées sont la distribution constante, l'Erlang-10, l'Erlang-2, et l'hyperexponentielle, si bien que le coefficient de variation $C(T) = Var(T)/E(T)^2$ prend les valeurs 0, 0.1, 0.5, 2. , 10. ,100. Tous les résultats ont été obtenus avec $\lambda = 1$

Les figures [5] et [6] présentent les termes d'erreur $(E-E)/E$ et $(\overline{C} - C)/C$ dans l'hypothèse de la calibration A. Les résultats montrent la sensibilité de l'erreur à la charge de la station, et mettent en évidence le fait que $(\overline{E} - E)/E$ est maximum pour les valeurs moyennes de U, et s'annule pour U = 0 et U = 1 comme on pouvait s'y attendre. Par ailleurs, les courbes montrent que l'erreur est minimum pour N = 2 et croit avec N vers un maximum atteint pour $N = \infty$ ce qui correspond au système M/G/1 ouvert. Dans le cas de $(\overline{C} - C)/C$, il faut remarquer que deux erreurs de signes opposés interviennent : l'erreur sur C(T) et l'erreur sur C[q(N)]

Les erreurs obtenues lorsque la calibration B est utilisée sont présentées sur les figures [7] et [8] . Pour des valeurs de U proches de 1. , le signe de $(\overline{E} - E)/E$ change ainsi qu'on peut le voir figure [7] , et, en conséquence, l'erreur est alors inférieure à celle obtenue précédemment.

La figure [8] illustre les erreurs obtenues sur U(N) avec la calibration B. Il est remarquable de noter que les erreurs obtenues restent faibles, même dans le cas d'une distribution constante où l'erreur maximum ne dépasse pas 10%, et dans le cas d'un fort coefficient de variation où l'erreur reste inférieure à 20%.

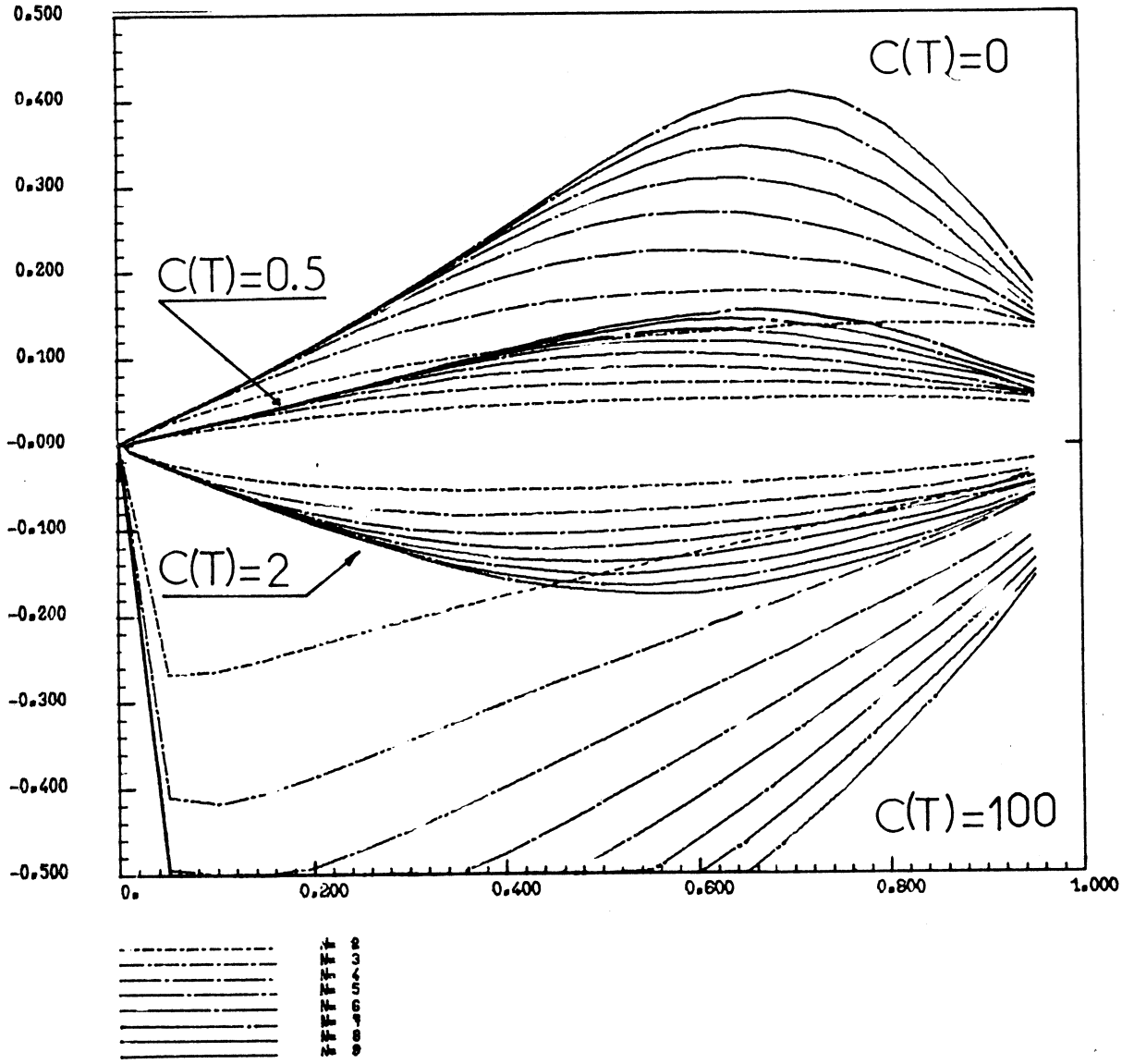


Figure 5 : erreurs relatives $(\bar{E} - E)/E$ (calibration A)

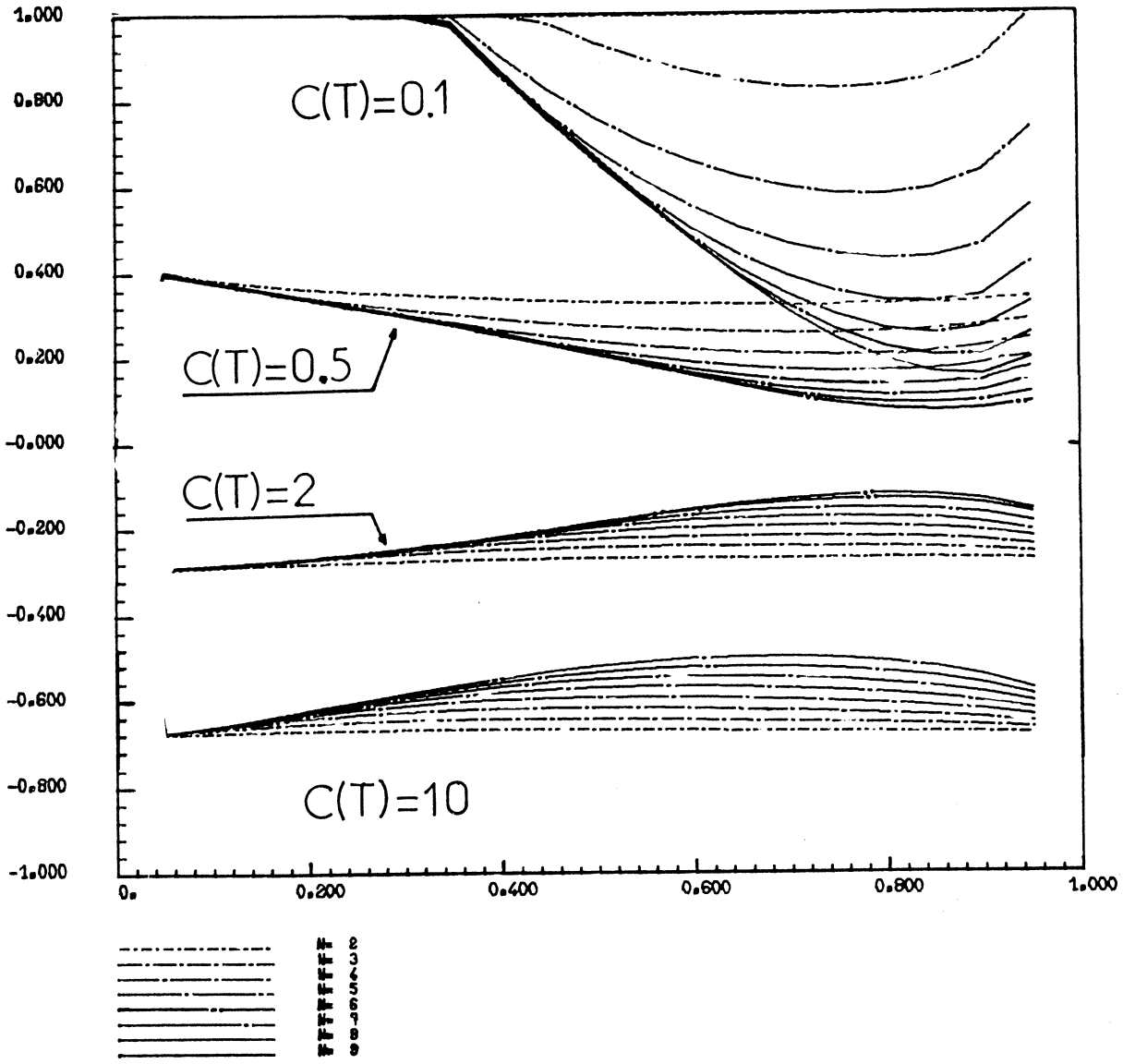


Figure 6 : erreurs relatives $(\bar{C} - C)/C$ (calibration A)

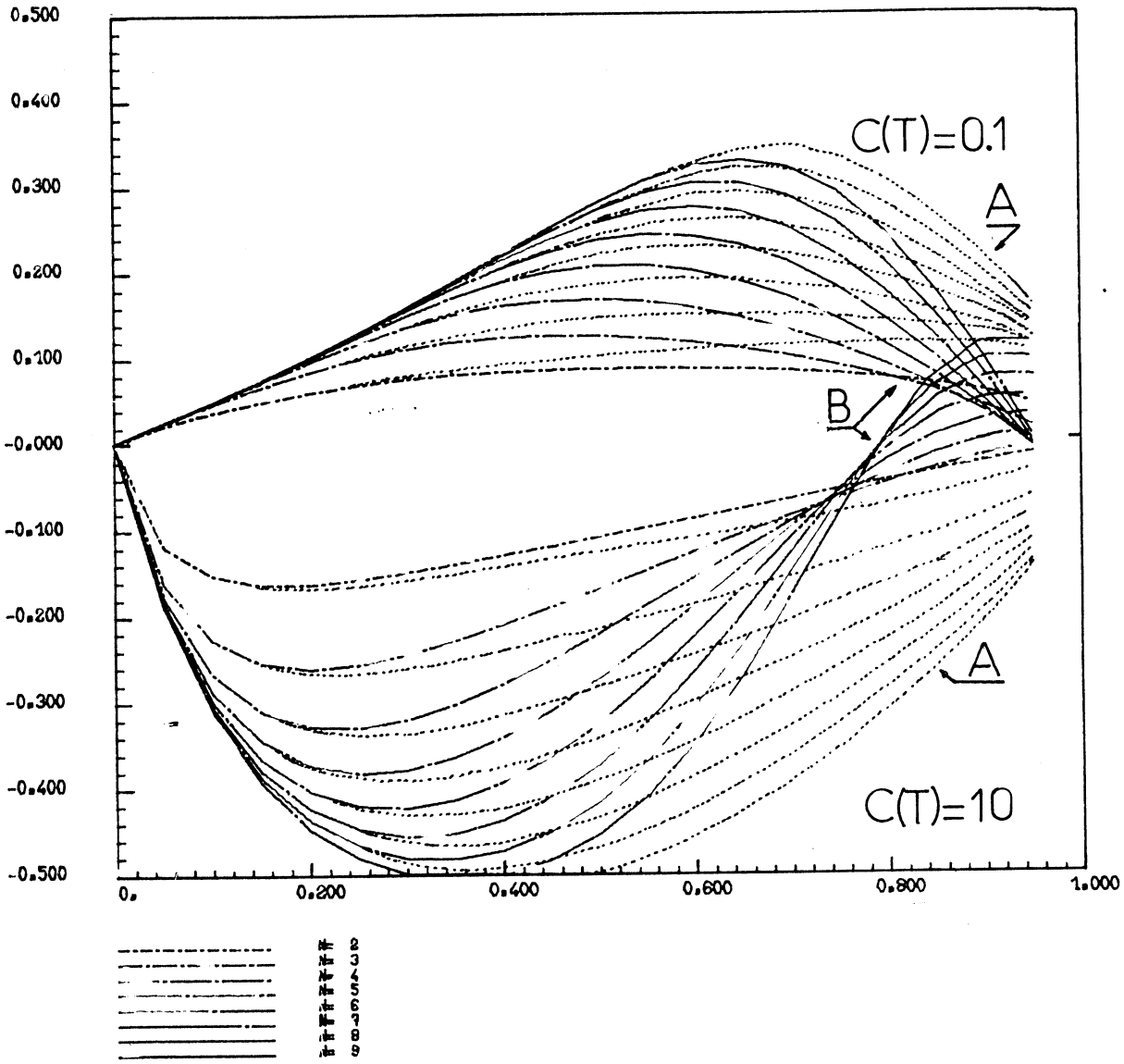


Figure 7 : erreurs relatives $(\bar{E} - E)/E$ (calibration A et B)

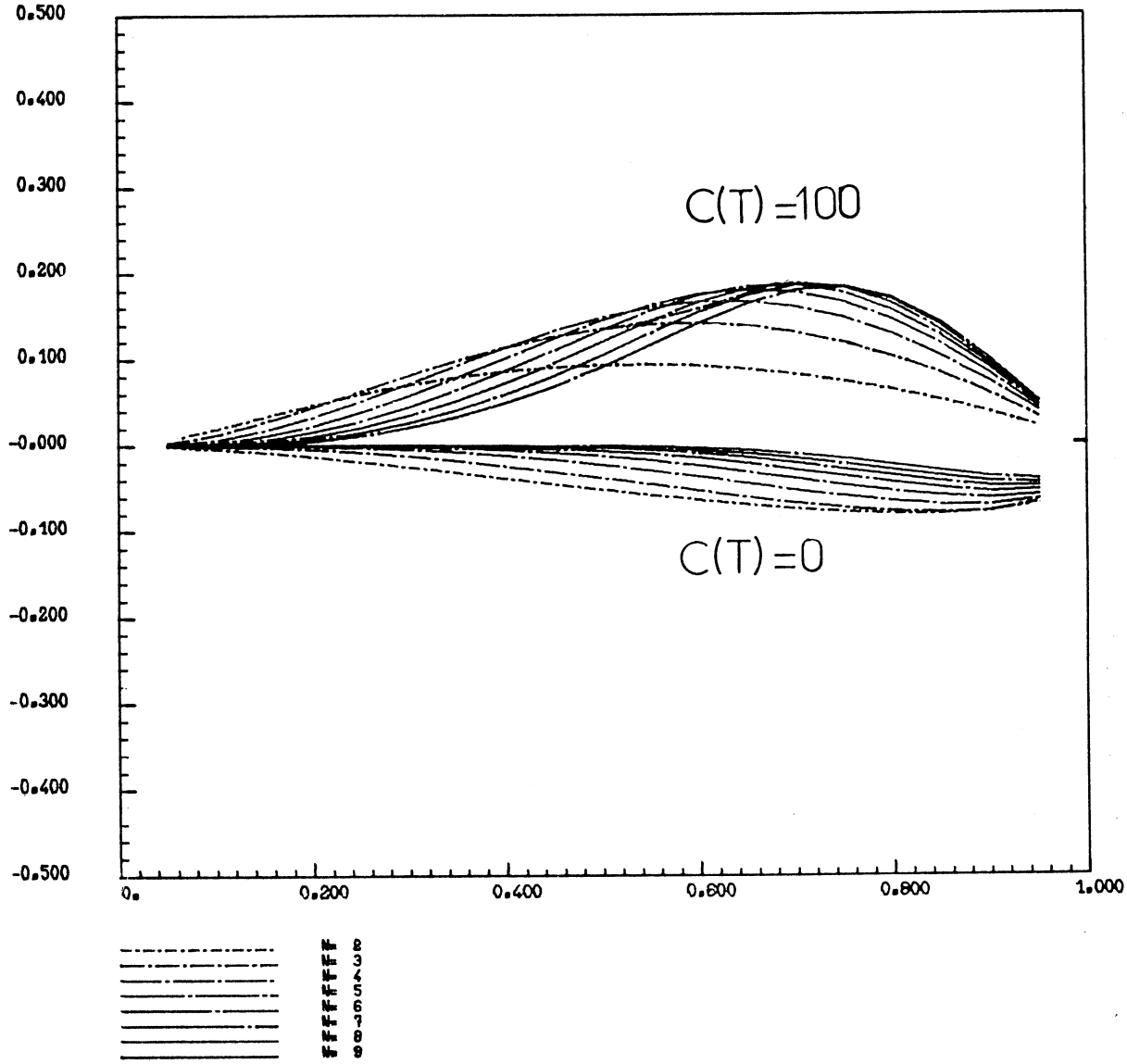


Figure 8 : erreurs relatives $(\bar{U} - U)/U$ (calibration B)

Nous pouvons tirer plusieurs conclusions de ces observations. La première porte sur les méthodes de calibration qui, nous l'avons vu, n'apportent pas de différences sensibles sur la valeur de l'approximation. Dans ces conditions, il semble que le choix devra se porter sur la méthode la plus simple à mettre en oeuvre en fonction du problème posé. La seconde conclusion, plus essentielle, porte sur les propriétés de robustesse des modèles à réseau de files d'attente fermé que l'on peut déduire de cette étude. Le point important nous parait être la différence des propriétés de robustesse suivant le type de mesure considéré. En ce qui concerne les temps de réponse, et le résultat serait semblable pour les temps d'attente, les erreurs peuvent être importantes, et dépendent fortement de la distribution réelle. En revanche, pour l'utilisation du serveur, les résultats montrent une bonne robustesse du modèle pour une très large plage de distribution.

B. - Résultats pour un réseau quelconque : théorèmes de Chang - Lavenberg

Les résultats précédents ont été obtenus pour un réseau simple à deux serveurs ce qui permet le calcul de la distribution d'équilibre du temps de réponse. Ce calcul n'est plus possible dans le cas d'un réseau général où plusieurs stations ont une distribution de service quelconque. Toutefois, en ce qui concerne les taux d'utilisation des serveurs du réseau, l'on dispose d'un résultat important dû à Chang et Lavenberg [7].

Considérons une réseau de files d'attente fermé avec N clients et M stations de service. Un client achevant son service à la station i se dirige vers la station j avec la probabilité p_{ij}. La matrice $P = [p_{ij}]$ est irréductible. Les temps de service à la station i sont indépendants et identiquement distribués avec une moyenne $0 < \mu_i^{-1} < \infty$ et une distribution générale $F_i = F_i(t)$. La station i contient m_i serveurs, avec $m_i \geq 1$, $1 \leq i \leq M$. Les temps de service sont indépendants d'une station à l'autre et la règle de service est PAPS.

Théorème

Soit $w_i(t)$ le service fourni par la station i durant l'intervalle de temps $[0, t]$ alors $U_i = \lim_{t\to\infty} \frac{w_i(t)}{t}$ existe avec probabilité un, et est indépendant de l'état du réseau à $t = 0$.

Théorème

Soit $\Pi = (\Pi_1, \ldots, \Pi_M)$ l'unique vecteur de probabilités satisfaisant $\Pi P = \Pi$, alors pour tout (i,j)

$$(9) \qquad \frac{U_i}{U_j} = \frac{\Pi_i \mu_i^{-1}}{\Pi_j \mu_j^{-1}}$$

Les résultats cités expriment que, quelles que soient les distributions qui interviennent aux serveurs, les taux d'utilisation de ceux-ci sont liés entre eux deux à deux par des relations de proportionalité ne dépendant que des moyennes des temps de service et des probabilités de passage par les différentes stations. En particulier, la relation (9) indique que si une résolution approchéeconduit à une erreur faible $\Delta U_i/U_i$ pour la station i du réseau, la même erreur sera commise pour toutes les autres stations.

3.2.2. - Cas d'un système ouvert

A. - Influence de la politique de service.

Les résultats du théorème BCMP [2] montrent que les distributions exponentielles et non-exponentielles produisent exactement la même distribution d'équilibre, et donc la même utilisation du serveur et le même temps d'attente lorsque la station de service répond à certaines conditions. C'est le cas en particulier lorsque le serveur est géré suivant la règle du processeur partagé c'est à dire lorsque les programmes présents dans la station s'exécutent en parallèle sur le serveur et en partageant également la vitesse de celui-ci. Le mode de gestion par processeur partagé est un mode idéal qui est approché pour la gestion de l'unité centrale dans les systèmes à temps partagé par la technique du quantum, le partage total du serveur étant obtenu en faisant tendre la durée du quantum vers 0.

On peut donc déduire de ces remarques qu'un modèle à files d'attente géré suivant la politique du "round-robin" est robuste par rapport à l'hypothèse exponentielle, d'autant plus que le quantum est petit, c'est à dire que l'on s'approche du cas idéal, et donc des hypothèses du théorème BCMP. Pour illustrer ce point nous avons représenté sur la figure 9 la variation du temps moyen d'attente w dans un système "round-robin" en fonction de la durée du quantum pour différentes distributions de service du type hyperexponentiel avec un coefficient de variation prenant les valeurs 1., 2., 5., et 10.

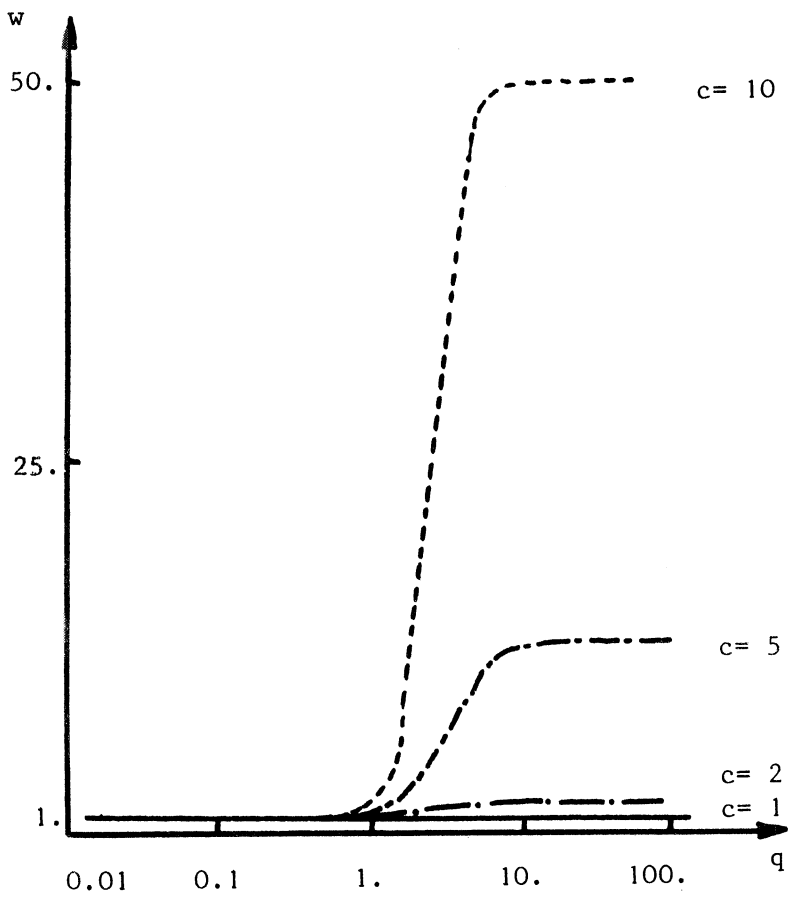


Figure 9

La figure 9 illustre bien comment, tant que la durée du quantum reste inférieure au temps moyen de service, ce temps d'attente est pratiquement indépendant du coefficient de variation de la distribution de temps de service, alors que, pour les valeurs de q supérieure, la dépendance apparaît fortement, et le système se comporte comme sur le modèle M/G/1/PAPS dès que q=10 sec.

B. - Comportement à forte charge : résultats asymptotiques

Il est aisé de voir que dans un système à files d'attente l'influence des distributions des intervalles de temps entre arrivées et des temps de service sur le comportement du système diminue quand la charge du serveur considéré tend vers l'unité. Cette intuition est vérifiée rigoureusement par Kingman [20] qui établit que pour une file d'attente GI/G/1 la distribution stationnaire du temps d'attente tend vers la distribution stationnaire du temps d'attente du système M/M/1 quand la charge du serveur tend vers 1.

Ce résultat indique que si la charge du serveur est proche de l'unité on pourra donc obtenir de bonnes indications ur le comportement d'une file GI/G/1 à partir du comportement de la file M/M/1 correspondante. Il est à rapprocher des observations faites sur l'approximation par diffusion qui est également d'autant plus satisfaisante que la charge est importante.

3.3 - Robustesse par rapport à la capacité des files d'attente.

Le calcul des solutions stationnaires des réseaux à file d'attente et l'obtention de la solution sous une forme produit simple suppose que les files sont indépendantes les unes des autres et, en particulier, que leur capacité sont illimitées afin qu'aucun phénomène de blocage ne puisse intervenir qui rendrait les files dépendantes les unes des autres. Toutefois, dans les systèmes réels les files d'attente qui entrent en jeu ont toujours évidemment des capacités finies ce qui amène des phénomènes de rejet ou de blocage. Aussi, l'utilisation de réseau de files d'attente pour l'analyse des performances de tels systèmes introduit-elle des erreurs liées aux hypothèses de capacité illimitée nécessaires au traitement mathématique. C'est l'étude de ces erreurs pour un système simple à deux serveurs, dans les cas du réseau ouvert et du réseau fermé que nous présentons dans ce paragraphe. Les résultats théoriques sur les files d'attente avec blocage proviennent de [33].

3.3.1. - Cas d'un réseau ouvert.

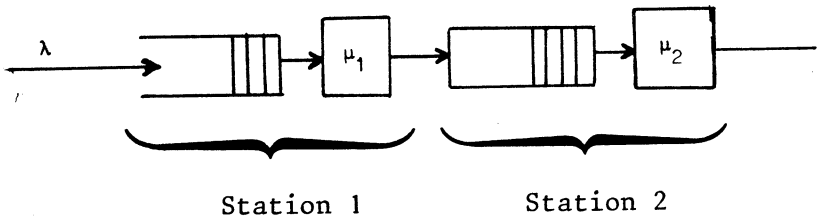


Figure 10

Le système à file d'attente étudié est représenté sur la figure 10 . Le processus d'arrivée est poissonnien avec pour paramètre

λ et la règle de service est PAPS aux deux serveurs. La station numéro 1 a une capacité illimitée $M_1 = \infty$; la station 2 a une capacité limitée à M_2. Quand un client achève son service à la station 1 il pénètre dans la station suivante si la file d'attente de cette station n'est pas pleine. Si non, le client reste dans la station 1 et en bloque le service jusqu'à ce qu'il puisse entrer dans la station 2. Les distributions de temps de service aux deux stations sont exponentielles de paramètres μ_1 et μ_2

Soit à un instant donné, n_1 le nombre de clients dans la station 1 y compris le client bloqué si c'est le cas, n_2 le nombre de clients dans la station 2, augmenté de 1 si un client est bloqué dans la station 1. Nous avons donc :

$$n_1 \geq 0$$

$$M_2 + 1 \geq n_2 \geq 0$$

Le processus (n_1, n_2) est une chaîne de Markov apériodique et irréductible. La condition d'équilibre du système et les temps moyens d'attente à chacune des stations sont obtenus suivant la méthode présentée en [32]. Les résultats principaux sont rappelés en Annexe B.

3.3.2 - Cas d'un réseau fermé.

Le modèle fermé est représenté sur la Figure 11. Les hypothèses sont identiques à celles prises précédemment et, de plus, nous supposons que la capacité de la station 1 est limitée à M_1. Soit N le nombre total de clients dans le système en supposant ;

$$M_1 + M_2 > N > \sup [M_1, M_2]$$

qui assure que le système n'est pas en état de blocage permanent et qu'un blocage est possible.

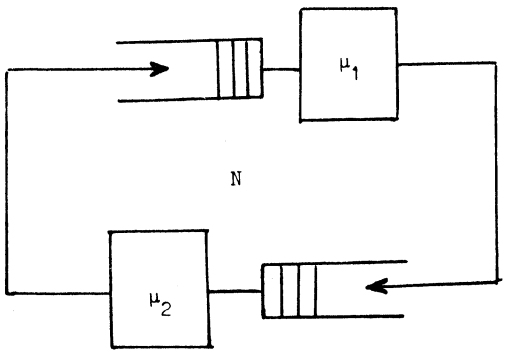


Figure 11

Soit $p(n_1,n_2)$ la probabilité d'équilibre d'avoir n_1 clients dans station 1, n_2 dans la station 2, en supposant qu'un client bloqué est compté uniquement dans la station qui est la cause du blocage. Nous avons donc :

$$\begin{cases} n_1 + n_2 = N \\ N - M_2 - 1 \leq n_1 \leq M_1 + 1 \\ N - M_1 - 1 \leq n_2 \leq M_2 + 1 \end{cases}$$

et d'après [32] , les résultats suivants :

Soit $\nu_1 = 1/\mu_1$, $\nu_2 = 1/\mu_2$, alors

i - La distribution d'équilibre $p(n_1, n_2)$ est donnée par :

$$p(n_1,n_2) = C\nu_1^{n_1} \nu_2^{n_2}$$

où C est une constante de normalisation.

ii - Le temps moyen de passage dans le système W est donné par :

$$W = \frac{N\nu_1}{1-p(N-M_2-1,M_2+1)} = \frac{N\nu_2}{1-p(M_1+1,N-M_1-1)}$$

Remarque :

Si $\mu_1 = \mu_2 = \mu$ alors

$p(n_1,n_2) = 1/(M_1+M_2-N+3)$, pour tout (n_1,n_2),

$$W = \frac{N}{\mu} \frac{M_1+M_2-N+3}{M_1+M_2-N+2}$$

4.3. - Valeur de l'approximation de capacité illimitée

Connaissant la solution de deux systèmes à file d'attente simples faisant intervenir des files d'attente à capacité limitée, nous pouvons évaluer l'erreur commise lorsque l'on fait l'approximation de capacité illimitée. La méthode appliquée est semblable à celle utilisée précédemment et consiste à supposer qu'un système se conformant exactement au modèle à capacité limitée est analysé avec l'hypothèse de capacité illimitée. Pour chacun des modèles le temps de réponse moyen à la station 2 est calculé, la calibration des deux modèles étant faite sur la charge ρ_2 de cette station.

La figure 12 présente les erreurs relatives obtenues sur le temps de réponse moyen de la station 2 en fonction de ρ_2 et pour différentes valeurs de M_2. Les résultats sont sensibles à la charge de la station, puisque plus celle-ci augmente, plus la capacité limite risque d'être atteinte. Ainsi, l'erreur est inférieure à 1% pour ρ_2 =0.3 si M_2=3, alors que pour atteindre le même niveau d'erreur avec ρ_2=0.9 nous devons avoir $M_2 \geq 8$. Si nous prenons un niveau d'erreur de 10%, plus cohérent avec la précision des mesures et des estimations qui peuvent être faites lors de l'évaluation d'un système, nous obtenons une zone où l'approximation raisonnable est beaucoup plus importante.

Les erreurs obtenues pour le modèle fermé sont données dans la figure 13 . Le calcul est fait avec $M_1 = N-1$ de telle sorte qu'un blocage ne puisse pas se produire du fait de la station 1, M_2 variant de 2 à N-2, et en égalant pour le modèle à capacité limitée et celui à capacité illimitée l'utilisation ρ_2 de la station 2. Le terme d'erreur porte le temps de réponse moyen défini ici comme le temps moyen de passage par les deux stations. Les observations faites sur les résultats sont semblables à celles présentées au des avec, en plus, la remarque que l'erreur relative augmente avec N pour M_2 fixé ce qui tend à montrer que l'erreur croit lorsque l'on passe du réseau fermé au réseau ouvert. Ce point est vérifié en comparant les deux tableaux, où l'erreur du modèle ouvert est toujours supérieure à celle du modèle fermé.

Les résultats que nous avons présentés ont été obtenus pour des systèmes à files d'attente très simples, et ils ne peuvent être étendus directement à des systèmes plus complexes. Toutefois, ils illustrent l'effet du

M_2	taux d'utilisation de la station 2								
	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
1	0.017	0.058	0.116	0.191	0.285	0.413	0.603	0.952	*
2	0.002	0.010	0.028	0.055	0.093	0.147	0.231	0.328	0.800
3	-	0.001	0.007	0.017	0.036	0.057	0.096	0.165	0.356
4	-	-	0.002	0.005	0.011	0.023	0.041	0.075	0.165
5	-	-	0.001	0.002	0.004	0.009	0.018	0.034	0.079
6	-	-	-	0.001	0.002	0.004	0.008	0.016	0.038
7	-	-	-	-	0.001	0.002	0.004	0.008	0.016
8	-	-	-	-	-	0.001	0.002	0.004	0.008
9	-	-	-	-	-	-	0.001	0.002	0.004

Figure 12 : erreur relative sur le temps de réponse moyen à la station 2.

N	M_2	taux d'utilisation de la station 2								
		0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
5	2	.001	.006	.019	.039	.069	.109	.163	.243	.381
	3	-	0.001	.004	.010	.020	.034	.052	.080	.125
10	2	.001	.007	.021	.046	.087	.150	.245	.398	.694
	3	-	.001	.006	.017	.037	.072	.126	.215	.379
	5	-	-	.001	.002	.008	.019	.040	.075	.139
	7	-	-	-	-	.002	.005	.011	.024	.047
	8	-	-	-	-	-	.002	.004	.010	.020
15	3	-	.001	.006	.017	.033	.074	.135	.241	.450
	5	-	-	.001	.003	.008	.022	.048	.098	.198
	7	-	-	-	-	.002	.007	.019	.046	.100
	10	-	-	-	-	-	.001	.005	.014	.036
	13	-	-	-	-	-	-	.001	.002	.006

Figure 13 : erreur relative sur le temps de réponse moyen à la station 2.

phénomène de blocage sur les performances et peuvent fournir une indication sur le comportement d'un système comportant dans une de ses parties une file limitée.

3.4. - Conclusion

Les études de robustesse présentées ont porté sur trois hypothèses en considérant deux mesures de performances différentes : l'utilisation des serveurs et les temps de réponse moyen. Les résultats montrent une bonne robustesse des modèles à file d'attente en ce qui concerne les taux d'utilisation des serveurs vis à vis de la distribution des temps de service. Ce résultat s'explique par l'existence dans un réseau de files d'attente de relations simples entre les taux d'utilisation des serveurs, indépendantes des distributions de temps de service, qui imposent une certaine "rigidité" à ces quantités et limitent leur dépendance aux hypothèses de distribution. Il resterait à vérifier que cette conclusion est encore valable lorsqu'on relâche d'autres hypothèses, comme l'indépendance des temps de service à un serveur : l'autocorrélation d'une suite de variables aléatoires étant un phénomène du même ordre que leur variance, on peut penser que les propriétés de robustesse demeurent dans ce cas. A notre connaissance, on ne dispose pas actuellement de résultats publiés sur ce point. Des expériences réalisées par P.A.W. Lewis [26] sur une file M/G/I/N par simulation en utilisant des générateurs de variables aléatoires dont on peut contrôler l'autocorrélation [27] indiquent que le taux d'utilisation du serveur est peu affecté par l'autocorrélation des intervalles de temps entre arrivées ou des taux de service. En revanche, pour certaines dépendances particulières entre les arrivées [17] cet effet est plus sensible.

Ajoutons un dernier argument en faveur des propriétés de robustesse des modèles fermés à file d'attente en ce qui concerne les taux d'utilisation des serveurs en présentant quelques résultats d'expériences de prédictions de performance réalisées sur des ordinateurs importants. La qualité des résultats obtenus, alors que de toute évidence les hypothèses de distribution, d'indépendance, etc..., ne sont pas satisfaites prouve à posteriori la robustesse de ces modèles. Ces études de prédiction de performances consistent à évaluer les conséquences d'une modification de la configuration du système étudié en construisant un modèle du système et en validant ce modèle sur la première configuration pour un benchmark de programmes donnés. En modifiant ensuite les paramètres du modèle, par exemple la vitesse de l'unité centrale, la taille de la mémoire, ou le nombre des canaux, on peut alors prédire les modifications de performances correspondantes. La valeur de la prédiction obtenue est alors vérifiée en exécutant le benchmark sur la nouvelle configuration.

La figure 14 présente des résultats d'une étude [19] de prédiction de performances faites sur un UNIVAC 1108 pour évaluer les conséquences d'une augmentation d" la taille mémoire de 128 K mots à 192 K mots. Les données figurant dans le tableau concernent la configuration à 192 K mots.

	Utilisation UC	Utilisation disque 1	Utilisation disque 2	Utilisation disque 3	Utilisation bande
PREDICTION	.77	.32	.30	.71	.22
MESURE	.74	.29	.27	.68	.22

Figure 14 : UNIVAC 1108 - Résultats de prédictions de peformances

Un exemple de prédiction de performances par modèles dans le cas d'une modification du nombre de canaux dans [35]. Le but de l'étude était d'évaluer les conséquences de l'addition d'un canal sélecteur supplémentaire à un IBM 370/155 sous OS/MUT possédant déjà un canal sélecteur et un canal multiplexeur. La figure 15 donne les résultats prédits et mesurés pour la nouvelle configuration.

	Utilisation UC	Utilisation canal 1	Utilisation canal 2	Utilisation canal 3
PREDICTION	.472	.540	.214	.151
MESURE	.473	.535	.214	.152

Figure 15 : IBM/370/155 - Résultats de prédictions de performances

D'autres expériences semblables sont en cours à la Compagnie CII - Honeywell-Bull [29] et à la CISI [18] et les premiers résultats obtenus indiquent une qualité de prédiction très satisfaisante.

L'ensemble des observations faites sur les taux d'utilisation des serveurs ne se généralise malheureusement pas en ce qui concerne le temps de réponse : comme on peut voir d'après les résultats obtenus, cette quantité est très sensible aux hypothèses de modélisation et il ne semble pas que l'approximation obtenue soit acceptable. La première étude montre combien l'hypothèse de source infinie est discutable pour la prédiction du temps de réponse si le nombre de consoles n'est pas assez élevé. Il convient toutefois de noter, pour mieux apprécier les conséquences de ce résultat, que ce type de modèle à source infinie est le plus souvent utilisé non pour obtenir une valeur absolue d'une grandeur donnée, dans notre cas le temps de réponse, mais plutôt pour obtenir un écart relatif sur cette grandeur, par exemple en comparant à l'aide du modèle les performances de deux algorithmes différents : c'est ce qui est fait dans le chapitre 2 où nous utilisons un modèle à scurce finie pour comparer les performances de deux politiques d'allocation U.C. Dans le cas du système à source finie la comparaison est faite par simulation et les résultats montrent que les écarts relatifs observés sont comparables à ceux calculés pour le modèle à source infinie, bien que les grandeurs elles-mêmes soient très différentes. Nous pouvons donc tirer une première conclusion positive de cette remarque, en notant que l'approximation par source infinie respecte, au moins pour les expériences que nous avons pu faire, les écarts relatifs.

Dans le cas des hypothèses de distribution, il semble en revanche difficile de tirer une conclusion favorable sur les propriétés de robustesse vis à vis du temps moyen de réponse, sauf dans le cas d'un serveur géré suivant la politique du processeur partagé. Cette politique d'allocation est sensiblement réalisée pour l'unité centrale d'un ordinateur, mais représente mal les mécanismes d'allocation au niveau des périphériques, sources de la plus grande partie des temps d'attente, et la prédiction des temps de réponse dans un ordinateur complexe paraît donc délicate. Il faut ajouter à ces difficultés l'influence de l'autocorrélation des intervalles de temps entre arrivées ou des temps de service.

Cet effet a été observé sur un modèle à files d'attente du sous-système de gestion de données DL/1 du système IMS [24] qui s'est révélé inapte,malgré sa complexité, à prédire de façon satisfaisante les temps de réponse en raison de l'autocorrélation entre les requêtes.

En plus des qualités de robustesse, les propriétés de sensibilité sont également importantes pour permettre l'utilisation des modèles à files d'attente. La sensibilité $\sigma_{E,S}$ de S par rapport à E se définit comme le rapport entre la variation Δ E d'une grandeur d'entrée E du modèle et la variation ΔS correspondante d'une grandeur de sortie S.

$$\sigma_{S,E} = \Delta S/\Delta E$$

Il est équivalent de considérer l'élasticité $\zeta_{E,S}$ de S par rapport à E qui se définit par :

$$\zeta_{S,E} = \frac{\Delta S}{\Delta E}\frac{E}{S},$$

c'est à dire le rapport des variations relatives de S et E.

Une propriété remarquable des réseaux à files d'attente fermés jacksoniens mise en évidence par Williams [39] est que les élasticités des mesures de performances considérées -utilisation des serveurs, temps de réponse moyen, débit par rapport aux paramètres du modèle, temps moyen de service, probabilité de branchement-sont toujours inférieures à 1. Plus précisément, en désignant par U_i, R et T l'utilisation du serveur i, le temps de réponse moyen et le débit du réseau, et en notant $X_j = \pi_j/\mu_j$ le produit de la fréquence de passage π_j des clients par le serveur j par le temps moyen de service $1/\mu_j$ de ce serveur, nous avons

$$\left| \frac{X_j}{U_i} \frac{\partial U_i}{\partial X_j} \right| < 1 \quad , \qquad i=j$$

$$\left| \frac{X_j}{U_i} \frac{\partial R}{\partial X_j} \right| < 1 \quad ,$$

$$\left| \frac{X_j}{R} \frac{\partial T}{\partial X_j} \right| < 1 \quad ,$$

et

$$-u_j \leq \frac{X_j}{U_i}\frac{\partial U_i}{\partial X_j} \leq 0, \qquad i \neq j,$$

$$-U_j \leq \frac{X_j}{R}\frac{\partial R}{\partial X_j} \leq 0,$$

$$-U_j \leq \frac{X_j}{T}\frac{\partial T}{\partial X_j} \leq 0.$$

Ces résultats montrent qu'une erreur de p % sur X_j entraîne une erreur sur U_i, R ou T inférieure à p % et, en fait, inférieure à U_j p %. On vérifie bien aussi ce que l'intuition suggère, c'est à dire que l'erreur faite sur un paramètre d'un serveur a d'autant moins d'effet que ce serveur est peu chargé, et on montre en plus que cette erreur est toujours limitée, et n'est pas amplifiée par le modèle.

La conclusion de ces observations et remarques nous paraît être que les résultats sur les propriétés de robustesse des modèles à files d'attente sont encore fragmentaires et insuffisamment explorées . Ce fait constitue un frein important à la généralisation de l'utilisation des modèles à files d'attente en la limitant à des spécialistes avertis des différentes particularités de ces modèles. L'effort de recherche dans cette direction est donc indispensable et devrait être poursuivi à la fois au niveau de l'analyse mathématique des modèles pour tenter d'exhiber des lois de comportement et au niveau expérimental pour évaluer les implications des différentes hypothèses suivant la démarche que nous avons suivie dans ces trois études présentées.

Annexe A

Les paramètres de la distribution d'équilibre du temps d'attente q(N) sont donnés par :

$$(3) \qquad E[q(N)] = (N-1)\,E(t) - \sum_{k-1}^{N-1} k\,\Pi_k(N) \;/\lambda$$

$$(4) \qquad Var[q(N)] = (N-1)\,Var(T) + \sum_{k-1}^{N-1} \Pi_k(N) \;/\lambda$$

$$- \Big\{ \sum_{k-1}^{N-1} k\,\Pi_k(N)/\lambda \Big\}^2 - N(N-1)\,E(T)\,\Pi_N(N) \;/\lambda$$

où $\Pi_k(N)$, K=1,...N est la distribution stationnaire du nombre de programmes dans la station 1 immédiatement après un départ de la station 2. Le calcul de $\Pi_k(N)$ est obtenu comme suit.

Soit α_k, $k \geq 1$, la probabilité qu'il y ait au moins k départs de la station 1 durant la durée d'un service à la station 2. Il vient

$$(5) \qquad \alpha_k = \int_o^\infty \big[1-F(t)\big] \frac{\lambda^k\, t^{k-1}\, e^{-\lambda t}}{(k-1)!}\, dt \;, \quad k \geq 1$$

L'expression des $\Pi_k(N)$ s'obtient à partir des α_k, $k \geq 1$ suivant [23] par

$$\Pi_{N-1}(N) = 1/A_{N-2},$$
$$\Pi_k(N-1) = A_{N-1-k}/A_{N-2}, \quad 1 \leq k \leq N-1,$$
$$\Pi_k(N) = \Pi_{k-1}(N-1)\,A_{N-2}/A_{N-1}, \quad 2 \leq k \leq N,$$

avec

$$A_k = \sum_{j=o}^{k} a_j, \quad k \geq o,$$

et

$$a_o = 1,$$
$$a_1 = \alpha_1 / (1-\alpha_1),$$
$$a_k = \sum_{j=1}^{k-1} (a_j\,\alpha_{k+1-j} + \alpha_k) / (1-\alpha_1), \quad k \geq 2$$

A partir des équations (3) et (4) les paramètres de la distribution stationnaire du temps de réponse r(N) - temps d'attente augmenté du temps de service - à la station 2 s'obtiennent immédiatement.

$$(6) \qquad E\left[r(N)\right] = E\left[q(N)\right] + E(T),$$

$$(7) \qquad Var\left[r(N)\right] = Var\left[q(N)\right] + Var(T).$$

Par la suite, nous aurons également besoin de calculer l'utilisation d'équilibre u(N) du serveur de la station 2. D'après Gaver [9] nous avons :

$$(8) \qquad u(N) = \frac{b(N)}{b(N) + \lambda^{-1}},$$

où b(N) représente la durée moyenne de la période active au serveur 2 et est donné par :

$$b(N) = E(T) + \sum_{k=1}^{N-2} b(N-k)\ \alpha_{k+1} \ /\ (1-\alpha_1)\ ,\ N > 2$$

$$b(2) = E(T)/E\ (e^{-\lambda T})$$

Annexe B

L'analyse du processus (n_1, n_2) conduit aux résultats suivants :

Soit $\rho_1 = \lambda/\mu_1$ $\rho_2 = \lambda/\mu_2$. Alors, si

$$\frac{\rho_2^{M_2+2} - \rho_1^{M_1+2}}{\rho_2^{M_2+1} - \rho_1^{M_1+1}} < 1$$

nous avons :

i - la chaine de Markov $\mathcal{C}$ est ergodique

ii - la probabilité d'équilibre lorsque le système est vide est donnée par :

$$p_{oo} = \frac{1}{D}\left[\frac{1}{\rho_1^{M_2+1}} - \frac{1}{\rho_2^{M_2+1}}\right]\left[1 - \frac{\rho_2^{M_2+2} - \rho_1^{M_1+1}}{\rho_2^{M_1+1} - \rho_1^{M_1+1}}\right]$$

iii - la distribution d'équilibre $F_p, p = o, M_2 + 1$ du nombre de clients dans la seconde station est approchée par $\hat{F}_p, p = o, M_2 + 1$ donné par

$$\hat{F}_o = 1 - \rho_2,$$

$$\hat{F}_p = \frac{1}{D}\rho_2^p \frac{1}{1-\rho_1}\left[\frac{1-\rho_2}{\rho_2^{M_2+1}} - \frac{1-\rho_1}{\rho_2^{M_2+1}} - \frac{\rho_1-\rho_2}{\rho_1^p}\right], \quad p = 1, \ldots M_2$$

$$\hat{F}_{M_2+1} = \frac{1}{D}\left[\left(\frac{\rho_2}{\rho_1}\right)^{M_2+1} - 1\right]$$

avec

$$D = \frac{1}{\rho_1^{M_2+1}} \frac{1-\rho_1}{1-\rho_1} - \frac{1}{\rho_2^{M_2+1}} \frac{1-\rho_2^{M_2+1}}{1-\rho_2}$$

Remarque :

Dans le cas $\rho_1 = \rho_2 = \rho$, la condition d'équilibre devient

$$\rho < \frac{M_1+1}{M_2+2},$$

et les F_p sont **donnés** par

$$\hat{F}_p = (1-\rho)\,\frac{\rho^{M_2+2} + \rho^{p}\left[M_2+1 - p(M_2+2)\right]}{\rho^{M_2+2} + M_2 + 1 - \rho(M_2+2)}, \quad p = 0,\ldots, M_2+1$$

Soit n_2 le nombre de clients dans la seconde station, non compris le client éventuellement bloqué dans la station 1. Si la condition d'équilibre est satisfaite, alors nous avons

i - le nombre moyen de clients $E(n_1)$ dans la station 1 est approché par $\hat{E}(n_1)$ donné par :

$$\hat{E}(n_1) = \frac{\rho_1^{M_2+2} - \rho_2^{M_2+2} - (\rho_1-\rho_2)\left[\rho_1^{M_2+1}\hat{F}_{M_2+1} - A(1-\rho_2)\right]}{\rho_1^{M_2+1}(1-\rho_1) - \rho_2^{M_2+1}(1-\rho_2)},$$

avec $$A = \sum_{i=1}^{M_2} \rho_2^{M_2-i+1} F_i z_i + \rho_2 F_{M_2+1} z_{M_2},$$

et $$z_i = \frac{1}{(\rho_1-\rho_2)^2}\left[\rho_1^{i+1} - \rho_2^{i}\left[\rho_1(i+1) - i\rho_2\right]\right], i = 1,\ldots M_2.$$

ii = Le nombre moyen de clients dans la station 2, $E(n_2^*)$, est approché par $\hat{E}(n_2^*)$ donné par

$$\hat{E}(n_2^*) = \sum_{i=1}^{M_2} i\,F_i + M_2\,F_{M_2+1}$$

Remarque :

Si $\rho_1 = \rho_2 = \rho$, nous avons :

$$\hat{E}(n_1) = \frac{\rho(M_2+1) - \hat{F}_{M_2+1}}{M_2+1 - \rho(M_2+2)}$$

$$A = \sum_{i=1}^{M_2} \frac{i(i+1)}{2} \hat{F}_i + \frac{M_2(M_2+1)}{2} \hat{F}_{M_2+1}.$$

L'intérêt des résultats approchés cités est qu'ils s'expriment sous une forme simple, aisèment calculable. L'approximation a été étudiée dans [33] numériquement par comparaison avec le calcul de la solution exacte. Dans tous les exemples étudiés l'approximation conduit à une erreur inférieure à 3 % sur les quantités $E(n_1)$ et $E(n_2)$

Références

[1] M. BADEL, A. SHUM — "Accuracy of an approximate computer system model" Proc. 2nd International Workshop on modelling and performance evaluation of computer systems, E.GELENBE ed. North-Holland Publishing Company, 1976

[2] F. BASKETT et al — "Open, closed and mixed network of queues with differe classes of customers" J. ACM 22, 2, April 1975.

[3] F. BASKETT, F. PALACIOS — "Processor sharing in a central server queueing model of multiprogramming with applications" Proc. of the 6th Annual Princeton Conference on Information Sciences and Systems, Princeton University, March 72.

[4] C. BIGUEREAU, B. GARAGNON — "Approximation de réseaux de files d'attente" Projet de fin d'études, Année 1975, 1976, INSA de Renn

[5] P.J. BUZEN, P.S. GOLDERG — "Guidelines for the use of infinite source queueing models in the analysis of computer system performance" Proc. National Computer Conference, 1974.

[6] K.M. CHANDY, V. HERZOG, L. WOO — "Approximate analysis of general queueing networks" IBM I. Res. Develop., Jan. 1975.

[7] A. CHANG, S.S. LAVENBERG — "Work-rates in closed queueing systems" Oper. Res. 22, 4, 1974

[8] P.J. COURTOIS — "Decomposability, instabilities, and saturation in multiprogramming systems" C. ACM 18, 7, July 1975.

[9] D.P. GAVER — "Probability models for multiprogramming computer sytems" J. ACM 14, 3, July 1967.

[10] D.P. GAVER — "Diffusion Approximations and models for certain congestion problems" J.A.P. 5, 1968

[11] D.P. GAVER — "Analysis of remote terminal backlogs under heavy demand conditions" J. ACM 18, 3, July 1971.

[12] D.P. GAVER, G.S. SHEDLER — "Processor utilization in multiprogramming systems via diffusion approximations" Oper. Res. 21, 2, March-April 1973

[13] D.P. GAVER, G.S. SHEDLER — "Approximate models for processor utilization in multiprogrammed computer systems" SIAM J. Computing, 2, 3, Sept. 73

[14] D.P. GAVER, P A.W. LEWIS — "First order autoregressive Gamma sequences and point processes" To appear.

[15] E. GELENBE — "A non-markovian diffusion process and its application to the approximation of queueing and computer system behaviour" Res. Report 74-1, Chaire de Systèmes Informatiques, Université de Liège.

[16] E. GELENBE — "On approximate computer system models" J. ACM 22, 2, April 1975.

[17a] E. GELENBE

[17b] E. GELENBE, A. KURINCKX — "Random injection control of multiprogramming in virtual memory" Proc. 2nd International Workshop on Modelling and performance evaluation of Computer Systems, E.GELENBE ed., North-Holland Publishing Company, 1976

[18] A. GUILLON — Communication orale.

[19] P.H. HUGHES, G. NOE — "A structural approach to computer performance analysis" Proc. National Computer Conference, 1973

[20] J.F.C. KINGMAN — "The heavy traffic approximation in the theory of queues" Proc. Symp. on Congestion Theory, Eds. W.L.SMITH and W.E.WILKINSON, University of North Carolina Press, Chapel Hill, N.C. 1965

[21] H. KOBAYASHI — "Application of the diffusion approximation to queueing networks I : Equilibrium distributions" J. ACM 21, 2, April 1974.

[22] H. KOBAYASHI — "Application of the diffusion approximation to queueing networks II : non-equilibrium distributions and applications to computer system modelling" J. ACM 21, 3, July 1974.

[23] S.S. LAVENBERG — "The steady-state queueing time distribution for the M/G/1 finite capacity queue" Management Science 21, 5, Jan. 1975.

[24] S.S. LAVENBERG, G.S. SHEDLER — "A queueing model of the DL/1 component of IMS" IBM Research RJ 1561, April 1975.

[25] J. LEROUDIER, M. PARENT — "Discrete event simulation of computer system performance" Rapport de Recherche n°177, IRIA-LABORIA, Juin 1976

[26] P.A.W. LEWIS — Communication orale.

[27] P.A.W. LEWIS — "Generation of Gamma and mixed exponential time series with controlled dependence" To appear.

[28] P.A.W. LEWIS, G.S. SHEDLER — "Statistical analysis of non stationnary series of ever in a data base system" To appear in IBM J. Res. and Develop.

[29] D. MERLE — "Modèles mathématiques de système" Rapport de Stage, CII-CHB, Juin 1976.

[30] R.R. MUNTZ — "Asymptomatic properties of closed queueing network models" Proc. 8th Annual Princeton Conf. Information Sciences and Systems, March 1974.

[31] R.R. MUNTZ — "Poisson departure processes and queueing networks" Proc. 7th Annual Princeton Conf. Information Sciences and Systems, March 1973.

[32] G. PUJOLLE, D. POTIER — "Accuracy of the infinite waiting room approximation in open and closed queueing systems" Proc. Intern. Seminar and models and measures for Computer systems, Bologne, Feb.-March 1975.

[33] G. PUJOLLE, D. POTIER — "Réseaux de files d'attente à capacité limitée avec des applications aux systèmes informatiques" à paraître dans RAIRO.

[34] M. REISER, H. KOBAYASHI — "Accuracy of the diffusion approximation for some queueing systems" IBM J. Res. Develop. 18, 2, March 1974.

[35] C. ROSE — "Validation of a queueing model with classes of customers" Proc. Int. Symposium on Computer performance, modelling measurement and Evaluation, ACM/SIGMETRICS, IFIPS WG 7 Cambridge, Mass. March 1976.

[36] H.A. SIMON, A. ANDO — "Aggregation of variables in Dynamic Systems" Econometrica 29, 2, April 1961.

[37] W.J. STEWART — "MARCA, Markov Chain Analyser" Rapport de Recherche IRISA, Université de Rennes, 1976

[38] V.L. WALLACE, R.S. ROSENBERG — "RQA-1, The recursive Queue Analyser" Technical Report n°2, 1966, Dept. of El. Eng., University of Michigan, Ann Arbor.

[39] A.C. WILLIAMS, R.A. BHANDIWAD — "A generating function approach to queuing network analysis of multiprogrammed computers" Networks 6, 1, Jan. 1976.

Chapitre II

ALLOCATION ADAPTATIVE DE L'UNITE CENTRALE PAR QUANTUM

CHAPITRE II

Résumé

En introduction au second chapitre nous rappelons les problèmes posés par l'allocation de l'unité centrale dans les système à temps partagé. Nous proposons un algorithme d'allocation par quantum adaptatif qui a pour effet de permettre l'allocation de quanta supplémentaires aux programmes en fonction de la charge du système et de réduire ainsi l'overhead du système. Les hypothèses du modèle utilisé pour analyser cette règle d'allocation sont présentées et discutées et le modèle est résolu dans l'hypothèse du modèle ouvert pour des lois de distributions de service exponentielles et hyperexponentielles. Dans le cas du système fermé, la résolution est conduite par simulation. Les résultats numériques montrent les performances du mécanisme d'allocation suivant les paramètres choisis, et mettent en évidence la sensibilité des améliorations obtenues à la distribution de temps de service.

1. - INTRODUCTION

1.1. - Situation d'un programme vis à vis de l'allocation de ressources.

Le problème posé par l'allocation de l'unité centrale est le suivant : à un instant donné, seuls peuvent s'exécuter simultanément des programmes en nombre au plus égal au nombre d'unités centrales (U.C) ou processeurs du système considéré. De plus, l'activation d'un programme, c'est à dire l'allocation de l'U.C. à ce programme, n'est possible que si ce programme n'est pas bloqué en attente d'autres ressources (p.e. par une entrée-sortie) et dispose, on sera assuré de disposer au moment de son activation, de la ressource de mémoire centrale nécessaire à son exécution. Ces différents états d'un programme vont caractériser sa situation vis-à-vis de l'allocation de la ressource U.C. Classiquement ces états sont dénommés comme suit :

- état BLOQUE
- état PRET
- état ACTIF.

Un programme "bloqué" est un programme en attente de ressource autre que la ressource U.C. : l'activation d'un tel programme est donc impossible et l'allocation de l'U.C. ne se pose donc pas tant que l'état du programme reste bloqué. Un programme est "actif" lorsqu'il s'exécute sur une unité centrale. L'activation d'un programme donne lieu à un certain nombre d'opérations pour charger le contexte du programme activé et amener en mémoire centrale le code et les données nécessaires à son exécution; opérations qui se répètent lorsque le programme est désactivé et qui introduisent une charge supplémentaire sur l'unité centrale en "gestion propre" ou "overhead".

Parmi les programmes débloqués, certains ne peuvent toutefois être activés si toutes les U.C sont occupées. Ces programmes sont dans l'état "prêt", et c'est dans cet ensemble de programmes que l'on choisit les programmes à activer. Nous appelons allocateur -ou scheduler en anglais- l'algorithme chargé de déterminer l'allocation des U.C. aux programmes prêts. Le rôle de l'allocateur est de décider à la fois de l'ordre d'activation des programmes prêts et de l'allocation U.C. qui est faite à chaque activation.

Nous limiterons les analyses qui suivent à l'allocation de l'unité centrale dans un contexte monoprocesseur. Dans la mesure où les unités centrales sont banalisées, les résultats obtenus se généralisent au cas de systèmes multiprocesseurs.

1.2. - Fonction de l'allocateur

Tel que nous l'avons défini, l'allocateur contrôle l'accès des programmes à la ressource U.C., et donc l'exécution de ces programmes. Il s'agit d'une fonction essentielle qui a pour rôle de traduire, au niveau interne, les objectifs externes de gestion du système. La structure de l'allocateur , les règles de priorité et d'allocation des ressources qui y sont introduites n'ont de sens que par rapport à ces objectifs externes, eux-mêmes liés au système envisagé et à l'utilisation qui en est faite.

La conception de l'allocateur dépend également fortement de l'information qu'il peut obtenir sur les besoins en ressource U.C. des programmes. Si ces besoins sont connus complètement, comme c'est le cas par exemple dans un système temps réel où les traitements se répètent et ont des caractéristiques fixes, le problème se pose de façon différente que lorsque les demandes U.C. des programmes sont inconnues et largement variables, comme dans un environnement de temps partagé. Dans le premier cas l'allocation de l'U.C. pourra être déterminée complètement à priori par des techniques d'ordonnancement [6,10] ou en partie à l'arrivée du programme en lui affectant par exemple une priorité suivant sa nature [17] ; dans la seconde situation, le mécanisme de contrôle aura pour tâche de réaliser l'allocation en tenant compte davantage des besoins passés que des besoins futurs qui sont inconnus. En reprenant une terminologie classique en théorie du contrôle, on parlera de "contrôle en boucle ouverte" dans la première éventualité, de "contrôle en boucle fermée" dans la seconde. Remarquons que les deux types de contrôle peuvent être utilisés conjointement dans le cas d'un connaissance partielle du comportement du programme : si les programmes sont partagés en classes, la classe fournit une information pour affecter une priorité à un programme, l'allocation de l'U.C. à l'intérieur des programmes d'une même classe étant réglée par un contrôle en boucle fermée.

De ces remarques, il apparait que le problème posé n'a de solution que dans un contexte donné. Les résultats que nous présentons dans ce chapitre portent sur l'analyse de l'allocation U.C. à des programmes dans un contexte de temps partagé. Dans les paragraphes suivants nous présentons les objectifs imposés à l'allocateur dans un système de temps partagé et un bref historique des allocateurs réalisés avant d'aborder l'analyse par modèle à file d'attente des performances des allocateurs.

1.3. - Objectifs de l'allocateur dans un système à temps partagé

Dans un système à temps partagé l'allocateur a donc pour fonction principale de réaliser le partage du temps U.C. entre les programmes en assurant une priorité implicite aux programmes dont le temps d'exécution est court afin d'assurer à ces programmes un temps de réponse satisfaisant. La qualité du service est ainsi mesurée par le temps de réponse obtenu suivant la durée d'exécution du programme. Un second objectif est de conduire à l'utilisation optimale des ressources du système, c'est-à-dire de consacrer la plus grande part des ressources disponibles à l'exécution des programmes en limitant la part prise par la gestion propre du système. Il s'en suit que l'allocateur doit être simple, de façon à ce que sa mise en oeuvre n'alourdisse pas la gestion du système. Enfin, un dernier critère de performance de l'allocateur est sa robustesse, c'est-à-dire sa capacité à assurer un contrôle de l'allocation U.C. homogène et stable, aussi indépendant que possible des conditions de fonctionnement du système.

Les qualités d'un allocateur peuvent donc s'apprécier par les trois critères quantitatifs suivants :

- Temps moyen de réponse pour tous les programmes ;
- Temps moyen de réponse pour les programmes dont l'exécution à une durée t ;
- Taux de gestion propre, c'est-à-dire, le rapport de l'utilisation de l'U.C. en gestion propre sur l'utilisation U.C. consacrée aux programmes et des critères qualitatifs :

- robustesse ;

- simplicité.

Les critères quantitatifs représentent des mesures de performance élémentaires de l'allocateur, et il peut être parfois intéressant d'agréger ces mesures en faisant intervenir des coûts de façon à obtenir un critère unique. Ainsi, Rash [21]donne une mesure globale du temps de réponse moyen en affectant le temps de réponse R(t) d'un programme demandant t secondes d'exécution par une pondération e^{-at} qui fournit une mesure globale du comportement de l'allocateur puisque le terme e^{-at}, pour a > o, peut s'interpréter comme une priorité d'autant plus forte que le temps d'exécution t est faible. Toutefois le choix de la structure de coût retenue est largement dépendant du sytème particulier étudié, et une analyse du coût de fonctionnement d'un système de temps partagé à partir des critères définis plus haut n'a de sens que dans un contexte bien défini. Aussi nous limiterons nous,dans les analyses qui suivent, au calcul des mesures proposées et à leur discussion sans tenter d'y affecter des coûts.

1.4. - Conception de l'allocateur dans un système à temps partagé

Comme nous l'avons vu, le problème de l'allocation de l'U.C. dans un système à temps partagé peut se poser comme un problème de contrôle : suivant l'état du système caractérisé par les besoins futurs et passés des programmes, déterminer l'allocation U.C. qui maximise un certain critère choisi parmi les mesures citées. Une telle approche à l'inconvénient dans un système aussi mouvant qu'un système informatique où la charge et le comportement des utilisateurs sont très variables, de ne conduire qu'à une solution ponctuelle, valable dans un environnement donné, pouvant conduire à des performances médiocres dans tout autre situation. A notre connaissance, la seule expérience dans cette direction est celle de KASAYAP [14] sur le système CTSS [11]. En d'autre terme, le calcul direct d'une loi de contrôle risque de conduire à un mécanisme de contrôle peu robuste, alors qu'il s'agit d'une qualité essentielle. Aussi l'approche suivie consiste à se donner une structure de contrôle robuste, dépendant d'un certain nombre de paramètres, et à analyser les performances de cette structure de contrôle suivant les paramètre

Les structures de contrôle utilisées pour l'allocation de l'U.C. dans les systèmes à temps partagé s'inspirent toutes du principe de l'allocation par quantum mise en oeuvre pour la première fois sur le système expérimental BBN construit à Cambridge en 1962. Ce système, implanté sur un ordinateur DEC-PDP1 de 8 K mots de mémoire permettait à cinq utilisateurs de travailler "simultanément" à partir de cinq téléscripteurs grâce au mécanisme suivant. Les programmes prêts sont regroupés en une file d'attente (figure 1) , et servis par l'U.C. dans l'ordre de leur arrivée dans la file. Le programme en tête de la file obtient l'U.C. et la mémoire centrale pour une durée égale au plus à q unités de temps (q=140ms dans le système BBN).A la fin de ce quantum de temps ou dès que le vidage ou le remplissage d'un tampon est nécessaire pour une entrée-sortie, le programme actif est replacé en fin de la file d'attente et un autre programme est activé. Ce mode de gestion de l'U.C. par quantum est appelé "tourniquet" ou "Round-Robin". Remarquons que dans le système BBN chaque commutation d'un programme à l'autre s'accompagnait du déchargement du premier programme de la M.C. sur le tambour et du chargement du second, suivant la méthode du "swapping" total, et qu'il n'y avait pas de recouvrement entre l'exécution d'un programme et les opérations de chargement et de déchargement.

Si la méthode employée dans le système BBN réalise bien un partage de l'U.C. entre les programmes qui favorise, comme le montrera l'analyse rapportée dans les paragraphes suivants, les programmes courts, elle a l'inconvénient de conduire à un temps de gestion propre constant, quel que soit la durée d'exécution des programmes. D'où l'idée, appliquée à la conception de l'allocateur du système CTSS [11] , d'un quantum de temps et d'une priorité variable suivant la durée d'exécution et la taille de l'espace d'adressage du programme. Dans ce système l'allocateur gère N files d'attente ou niveaux, numérotés de 1 à N, 1 représentant la plus forte priorité. Un programme qui n'a pas achevé son exécution à l'expiration de son allocation U.C., ou qui est interrompu par une demande d'entrée-sortie, est alors transféré dans la file d'attente du niveau immédiatement supérieur, à moins que le dernier niveau n'ait été atteint. Chaque fois qu'un programme change de niveau sa priorité diminue et le nombre de quanta accordés à ce programme double. A l'entrée dans le système, c'est-à-dire à sa première activation, le niveau U affecté à un programme est déterminé en fonction de la taille de son espace d'adressage suivant la formule :

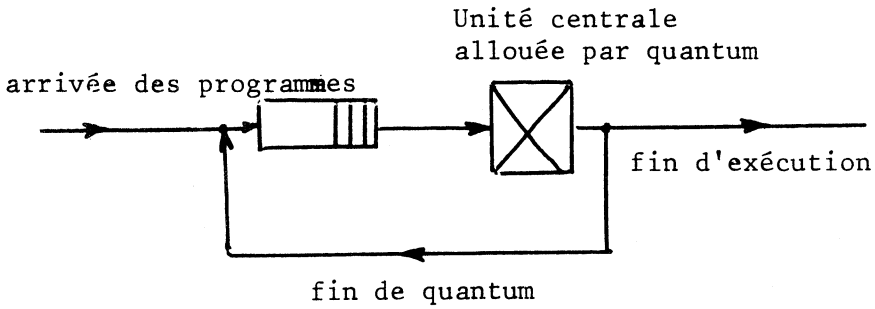


Figure 1 : Modèle d'allocation par quantum

$$U = \text{Min}\left[N, \text{Max}\ (1, \left[\text{Log}_2 \left\lfloor \frac{W_p}{W_q} \right\rfloor + 1 \right])\right]$$

où W_q est le nombre de mots qui peuvent être transférés de la mémoire secondaire à la M.C. durant le quantum élémentaire q, avec q = 200 ms ;

W_p est la taille, évaluée en mots, de l'espace d'adressage.

Ce mécanisme réalise bien une allocation par quantum variable, du fait de l'allocation de plusieurs consécutifs, ce qui conduit à une réduction du nombre de commutations de programmes et du temps de gestion propre

Une autre façon de réduire le temps de gestion est de rendre la durée du quantum variable non pas suivant la durée d'exécution des programmes, mais suivant la charge du système en réalisant un mécanisme d'allocation c quantum adaptatif [20]. L'intérêt est d'avoir un temps de gestion d'autant plus réduit que la charge imposée au système est lourde, et d'obtenir ainsi un fonctionnement plus stable du système et de retarder sa saturation. Le mécanisme que nous proposons et qui sera analysé en détail dans la suite du chapitre est le suivant. Le principe est celui du système à "tourniquet" classique, mais un programme en cours d'exécution peut recevoir un quantum supplémentaire à l'expiration du quantum en cours si les deux conditions suivantes sont satisfaites :

1 - à la fin du quantum en cours ce programme n'a pas achevé son exécution ;

2 - durant le quantum en cours au moins r nouveaux programmes sont entrés dans le système.

Le seuil r est le paramètre du mécanisme d'adaptation et définit la sévérité de la règle d'allocation supplémentaire. Au mécanisme classique de contrôle de l'allocation U.C. par quantum où le paramètre de la loi de contrôle est la durée du quantum, on a donc ajouté un mécanisme d'adaptation qui rend ce paramètre de contrôle dépendant de la charge du système. Un tel système n'est pas difficile à implémenter en hardware : si chaque nouvelle arrivée se manifeste par une interruption, un compteur initialisé à r produira un signal masquant l'interruption de fin de quantum si le nombre d'arrivées durant le quantum est au moins égal à r.

Dans les paragraphes suivants nous présentons un bref rappel des méthodes d'analyse et des principaux résultats pour les mécanismes d'allocation U.C. par "tourniquet" avant de présenter en détail l'étude du mécanisme proposé.

2. - LES MODELES ANALYTIQUES D'ALLOCATION DE L'U.C.

2.1. - Principes généraux

L'étude des modèles analytiques d'allocation de l'U.C. a pour objectif de fournir en fonction d'un certain nombre d'hypothèses sur l'architecture du système analysé, les caractéristiques des usagers et les règles d'ordonnancement, des résultats qualitatifs et quantitatifs sur le comportement et les performances du mécanisme d'allocation considéré.

Deux approches sont possibles. La première consiste à étudier la chronologie des événements se produisant au cours du fonctionnement du système en fonction d'hypothèses déterministes sur les usagers et les règles d'allocation : c'est l'approche déterministe, et dans une telle représentation, la suite des événements est donc entièrement déterminée par les hypothèses initiales. Cette approche, qui suppose connus de façon complète les besoins des usagers, n'est guère applicable dans le contexte de temps partagé que nous étudions, et a davantage son intérêt dans l'étude de l'allocation de l'U.C. dans les systèmes temps réels [6,17].

La seconde approche est celle des modèles stochastiques construits à partir d'hypothèses probabilistes sur le comportement des usagers, ou des différentes populations d'usagers. L'hypothèse principale est qu'à l'intérieur d'une population les usagers sont indiscernables et se comportent de façon identique. Une population est complètement caractérisée par le processus d'arrivée de ses membres au système, et la loi de distributions des demandes de calcul sur l'U.C.

L'intérêt de ces modèles est de permettre simplement, à condition de fonctionnement identique, la comparaison des performances de différentes règles d'allocation U.C., et les résultats fournis par de tels modèles doivent être appréciés ainsi de façon relative plutôt que de façon absolue. Le calcul des performances est réalisé sous l'hypothèse de fonctionnement stationnaire, c'est-à-dire en supposant que le système a atteint un état d'équilibre.

Les modèles stochastiques d'allocation de l'U.C. ont donné lieu à une littérature considérable. L'étude bibliographique du domaine réalisée en 1969 par MC KINNEY [15] rassemblait trente cinq références. Une étude plus récente due à WYSZEWIANSKI et DISNEY [25] en comporte cent seize. L'intérêt suscité par ces modèles s'expliquent par leur relative simplicité. Le nombre des paramètres est réduit et ceux-ci sont facilement accessibles par l'expérimentation, et la qualité des résultats obtenus qui fournissent une image réaliste du comportement du système réel. Dès 1965, les mesures et le modèle réalisés par SCHERR [23] montraient qu'un modèle analytique simple possédant une bonne généralité et fournissant des résultats précis pouvait être construit.

2.2. - Caractérisation des modèles analytiques d'allocation U.C.

Nous nous limitons ici aux modèles stochastiques d'allocation U.C. par tourniquet. En reprenant la classification de MC KINNEY [15] , les hypothèses qui caractérisent ce type de modèle interviennent sur les points suivants :

- nature de la source,
- processus d'arrivée,
- distribution de la durée de calcul.

Les hypothèses qui sont faites sur ces différents éléments conditionnent en partie la validité du modèle, et il est nécessaire de pouvoir évaluer leurs conséquences et de les situer par rapport aux comportements mesurés sur les systèmes. Les éléments présentés dans ce paragraphe permettront d'apprécier les résultats obtenus dans les paragraphes consacrés à l'analyse du mécanisme d'allocation adaptative.

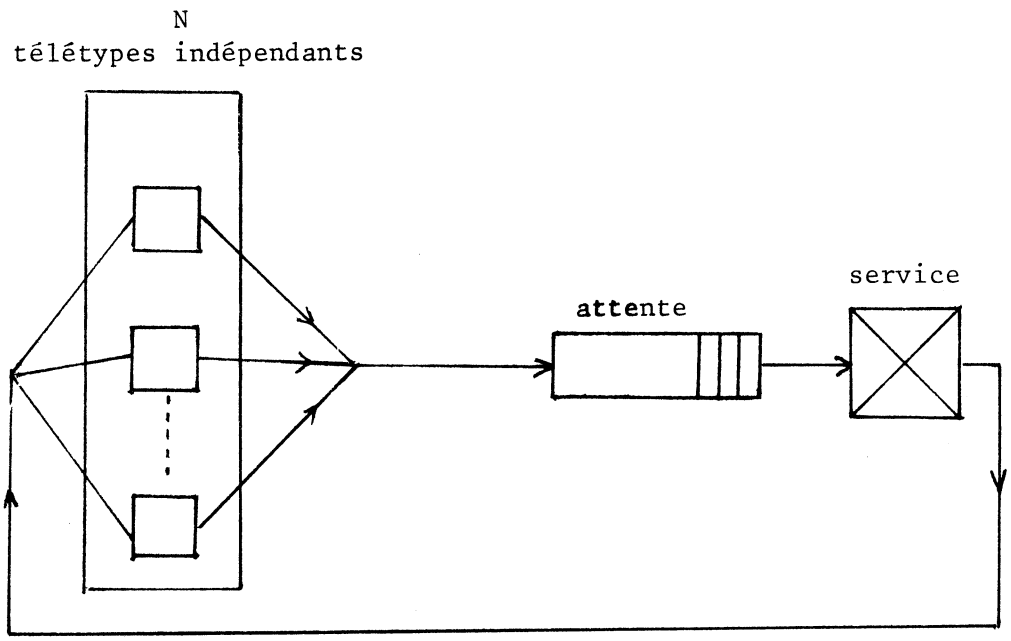


Figure 2 : Modèle à source finie

a) Nature de la source

Dans un système de temps partagé, la source des programmes est de nature finie, puisque, si N désigne le nombre de télétypes connectés au système, le nombre maximum de programmes présents dans le système à un instant donné est N. Si nous représentons simplement comme sur la figure 2 le système étudié avec l'hypothèse, qui sera discutée dans les paragraphes suivants, que les temps de réflexion des usagers aux consoles sont indépendants et identiquement distribués suivant une loi exponentielle de paramètre λ, le taux d'arrivée des programmes dépend du nombre de programmes n dans le système et j annule donc pour n = N. Toutefois si N est grand, et si le nombre moyen de programmes en attente reste limité, le taux d'arrivée des programmes variera peu. Dans une telle situation on peut alors supposer que le taux d'arrivée moyen des clients reste constant indépendamment du nombre de programmes dans le système.

C'est l'approximation qui est faite dans les modèles à source infinie. Le modèle à source infinie peut donc être considéré comme la limite du modèle à source finie lorsque N tend vers l'infini. L'erreur obtenue lorsqu'un système à source finie est approché par un modèle à source infinie a été étudiée dans le chapitre consacré à l'analyse des propriétés de robustesse des modèles à files d'attente. Nous en redonnons uniquement ici les résultats sur les figures 3, 4 et 5 qui illustrent l'erreur commise sur le temps de réponse moyen en fonction du nombre N de télétypes et de la charge ρ du système pour différentes distributions, ainsi que le nombre minimum de consoles nécessaire pour atteindre une erreur inférieure à 5 %.

Distribution	ρ .1	.2	.3	.4	.5	.6	.7	.8
Constant	2	3	6	10	17	32	61	160
Erlang-2	2	5	9	16	27	49	101	200+
Exponential	3	6	12	21	38	70	142	200+
Hyperexponential	4	8	16	27	48	93	188	200+

Figure 3

N	ρ .1	.2	.3	.4	.5	.6	.7	.8	.9
1	.147	.317	.537	.830	1.270	1.856	2.930	5.078	10.938
2	.074	.170	.306	.469	.720	1.080	1.666	2.921	6.366
3	.052	.117	.210	.340	.526	.798	1.249	2.175	4.598
4	.036	.092	.164	.270	.420	.655	1.013	1.725	3.932
5	.029	.076	.139	.219	.356	.574	.880	1.483	3.443
6	.024	.060	.115	.194	.305	.472	.752	1.310	3.071
7	.021	.053	.102	.168	.262	.420	.694	1.220	2.570
8	.018	.047	.087	.146	.242	.373	.608	1.124	2.357
9	.016	.042	.079	.134	.218	.353	.558	1.025	2.289
10	.015	.038	.072	.124	.196	.315	.540	.924	2.194
20	.008	.020	.037	.065	.111	.183	.318	.581	1.394
30	.005	.014	.025	.045	.077	.134	.239	.444	1.088
40	.004	.010	.019	.035	.061	.103	.185	.367	.922
50	.003	.008	.016	.028	.048	.087	.153	.303	.835
60	.003	.007	.013	.023	.041	.072	.135	.280	.739
70	.002	.006	.011	.020	.035	.063	.115	.246	.643
80	.002	.005	.010	.017	.031	.056	.104	.216	.592
90	.002	.005	.009	.016	.028	.051	.096	.202	.542
100	.002	.004	.008	.014	.025	.044	.088	.191	.533
120	.001	.003	.007	.012	.021	.038	.072	.160	.437
140	.001	.003	.006	.010	.018	.033	.064	.135	.422
160	.001	.002	.005	.009	.016	.029	.057	.124	.371
180	.001	.002	.004	.008	.015	.026	.052	.115	.325
200	.001	.002	.004	.007	.013	.024	.047	.106	.313

Figure 4 : distribution hyperexponentielle

N	ρ .1	.2	.3	.4	.5	.6	.7	.8	.9
1	.117	.254	.430	.664	1.016	1.484	2.344	4.063	8.750
2	.059	.135	.237	.379	.588	.894	1.348	2.387	5.339
3	.041	.093	.168	.274	.411	.641	1.007	1.764	3.794
4	.028	.073	.131	.217	.334	.524	.809	1.453	3.159
5	.023	.060	.105	.175	.274	.443	.696	1.240	2.672
6	.019	.048	.091	.149	.240	.382	.615	1.085	2.461
7	.017	.042	.077	.134	.212	.338	.566	1.006	2.145
8	.015	.037	.068	.116	.189	.308	.492	.919	1.935
9	.013	.033	.062	.106	.169	.273	.475	.829	1.860
10	.012	.030	.057	.098	.158	.259	.433	.782	1.762
20	.006	.016	.029	.050	.088	.148	.255	.473	1.157
30	.004	.011	.020	.035	.060	.102	.189	.370	.950
40	.003	.008	.015	.027	.045	.082	.151	.302	.786
50	.002	.006	.012	.022	.037	.065	.124	.260	.699
60	.002	.005	.010	.018	.031	.056	.104	.211	.608
70	.002	.005	.009	.015	.027	.049	.093	.195	.519
80	.001	.004	.008	.013	.024	.044	.084	.182	.510
90	.001	.004	.007	.012	.022	.038	.077	.159	.461
100	.001	.003	.006	.011	.020	.034	.066	.149	.415
120	.001	.003	.005	.009	.017	.029	.057	.124	.399
140	.001	.002	.004	.008	.014	.025	.050	.112	.348
160	.001	.002	.004	.007	.013	.022	.045	.102	.302
180	.001	.002	.003	.006	.011	.020	.041	.094	.290
200	.001	.002	.003	.006	.010	.018	.035	.087	.279

Figure 5 : distribution exponentielle

b) Processus d'arrivée des programmes

L'hypothèse couramment admise est que les processus d'arrivée des programmes est poissonnien. Cette hypothèse s'appuie sur le théorème de PALM-KHINTCHINE [12] qui montre que le processus obtenu par superposition de n processus de renouvellement indépendants tend vers un processus de Poisson. On en déduit que dans le cas d'un système à temps partagé le processus d'arrivée des programmes des différents télétypes fonctionnant indépendamment s'approche d'un processus de Poisson quand le nombre de télétypes est suffisamment grand. Cette hypothèse a été étudiée expérimentalement par COFFMAN et WOOD [9] sur le système TSS [24]. Les tests portent à la fois sur l'indépendance entre les intervalles de temps entre deux arrivées, et sur la distribution de ces intervalles de temps. Les résultats obtenus montrent la validité

de l'hypothèse d'indépendance, mais ne confirment pas complètement l'hypothèse Poissonnienne : la distribution des temps entre deux arrivées est plus exactement approchée par une distribution hyper- exponentielle de la forme :

$$f(x) = a\ \lambda_1\ e^{-\lambda_1 x} + (1-a)\lambda_2\ e^{-\lambda_2 x}$$

avec a = 0.615, λ_1 = 0.03 et λ_2 = 0.148, ce qui donne un coefficient de variation (rapport de l'écart type à la moyenne) de 1.2 peu éloigné de 1, coefficient de variation de la distribution exponentielle. Remarquons que les mesures effectuées portent sur un groupe de 22 télétypes, et que les conclusions seraient sensiblement modifiées pour un ensemble plus large de télétypes.

Une conséquence de l'hypothèse poissonnienne est qu'elle tend à négliger l'importance des longs intervalles de temps entre arrivées, et par là à sur-estimer la saturation du système. Toutefois, la faible valeur du coefficient de variation estimé montre que l'approximation exponentielle reste satisfaisante.

c) Distribution de temps de calcul

Un nombre important de résultats de mesures sur la distribution du temps U.C. pris par les programmes dans un système de temps partagé est disponible[3,18]et permet de caractériser le comportement des programmes. L'hypothèse d'indépendance a été testée par ANDERSON et SARGENT [2] en estimant le coefficient de corrélation pour 32 échantillons différents. Dans tous les cas les coefficients de corrélation ont été trouvés proche de zéro. D'autre part, les mesures rapportées dans[3] ainsi que dans LEROUDIER [18]. montrent que la distribution de temps de calcul a un coefficient de variation important allant jusqu'à 9 et 10.

Remarquons que ces chiffres concernent l'ensemble des programmes soumis au système, et qu'une caractérisation plus fine faite en distinguant le traitement effectué fait apparaître des différences importantes comme le montre la figure 6 établie à partir des résultats de [18].

	EXECUTION	EDITION	ASSEMBLEUR	COMP. FORT.	COMP. PL1
% U.C.	60	20	20		
Moyenne	3.4	.52	11.	5.2	12.7
Ecart-type	30.3	1.26	11.5	7.6	15.7
coefficient de variation	9	2.5	1	1.5	1.2

Figure 6

Pour les résultats cités, la distribution observée est approchée de façon satisfaisante par une distribution hyper-exponentielle à deux termes. Bien que de traitement mathématique moins simple que la distribution exponentielle, ce type de distribution défini comme la combinaison convexe de deux exponentielles permet dans certains cas une extension relativement directe des résultats obtenus dans le cas exponentiels, et c'est dans cette hypothèse que sera analysé le mécanisme d'adaptation.

3. - LE MODELE D'ALLOCATION ADAPTATIVE DE QUANTUM U.C.

Afin d'évaluer les performances d'une allocation adaptative de quantum U.C. aux programmes, le mécanisme d'allocation est analysé à l'aide de modèles analytiques étudiés sous différentes hypothèses. Les hypothèses les plus importantes, comme nous l'avons vu au dessus,concernent la nature de la source et la distribution de temps de calcul. Le premier modèle présenté suppose la source infinie et la distribution de temps U.C. exponentiel : il s'agit donc d'un modèle de type M/M/1. Dans le second modèle - du type M/G/1 - la distribution de temps de calcul est supposée être hyper-exponentielle, en accord avec les observations expérimentales citées au dessus. Le troisième modèle est le modèle M/G/1/N, c'est-à-dire le modèle fermé,qui sera étudié par simulation.

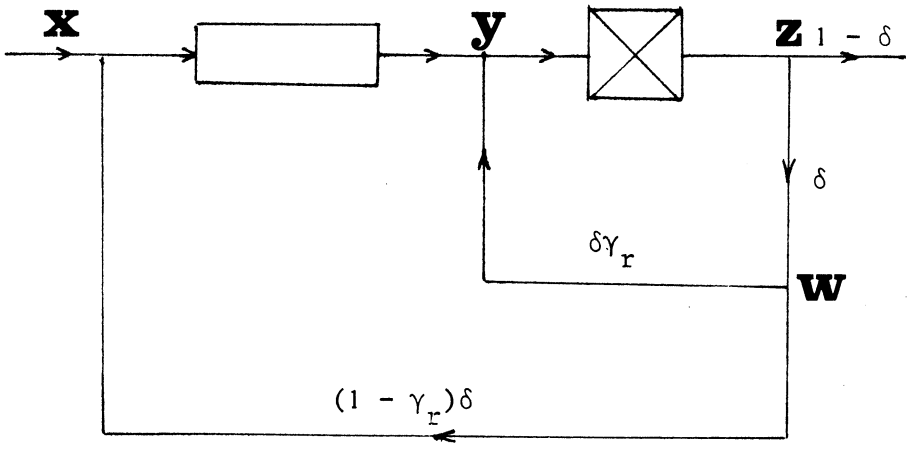


Figure 7

3.1. - Analyse du modèle M/M/1

Le modèle est représenté sur la figure 7 . Les programmes arrivent de l'extérieur au point x suivant un processus de Poisson de taux λ. Ils sont placés dans l'ordre PAPS dans la file d'attente. Les programmes arrivent également en x venant de w et sont également placés dans la file. Le nombre de programmes en attente ne comprend pas le programme en traitement sur l'U.C. ; le nombre de programmes dans le système comprend les programmes en attente, et celui éventuellement en cours de traitement. Soit γ_r, donnée par :

$$(1) \qquad \gamma_r = \sum_{i=r}^{\infty} e^{-\lambda q}(\lambda q)^i / i!,$$

la probabilité qu'il y ait au moins r arrivées durant la durée q du quantum. La durée de temps de calcul de chaque programme sur l'U.C. est une variable aléatoire t distribuée suivant une loi exponentielle F(t) de moyenne $1/\mu$. Remarquons que t ne comprend pas l'overhead de chargement qui est supposé constant égal à s et compté quand un nouveau programme est activé. La probabilité δ qu'un programme n'achève pas sont exécution durant un quantum est alors :

$$(2) \qquad \delta = \int_{q}^{\infty} e^{-\mu t} dt = e^{-\mu q}$$

Durant un quantum, un programme quitte donc l'U.C. et le système par le point z avec la probabilité $1 - \delta$. Le temps τ utilisé par un programme durant un quantum - en excluant le temps de commutation - est une variable aléatoire distribuée suivant $E(\tau)$ avec :

$$(3) \qquad E(\tau) = \begin{cases} 0 & \text{si } \tau < 0 \\ 1 - e^{-\mu\tau} & \text{si } 0 \leq \tau < q \\ 1 & \text{sinon} \end{cases}$$

La démonstration conduit aux expressions du temps moyen d'attente $e_r(k)$ pour les programmes demandant k quanta, le temps moyen d'attente pour l'ensemble des programmes E_r, et le taux de gestion propre θ_r en fonction des paramètres s, q, r, μ et λ.

$$e_r(k) = z_o + \frac{\rho}{\lambda(1-\rho)} \left[m_1' + m_1 + \lambda \frac{q \frac{\gamma_{r-1}}{\gamma_r} - Q}{1-\alpha} \right.$$

$$- \left[m_1' + \alpha(m_1 + \lambda \frac{q \frac{\gamma_{r-1}}{\gamma_r} - Q}{1-\alpha}) \right] \left[\gamma_r + \alpha(1-\gamma_r) \right]^{k-1}$$

$$- \frac{1 - \left[\gamma_r + \alpha(1-\gamma_r) \right]^k}{(1-\alpha)\ (1-\gamma_r)} \ \lambda q \frac{\gamma_{r-1}}{\gamma_r}$$

$$\left. + (k-1)\ \lambda q \gamma_{r-1} + \lambda Q \left[\gamma_r + k(1-\gamma_r) \right] \right],$$

avec :

$$z_o = \frac{\lambda s}{1-\xi} \left[\frac{s}{2} + \tau_1 \right] + \frac{\lambda}{\mu^2} \left(1 - \frac{\mu q \delta}{1-\delta} \right) + \frac{\lambda q \delta}{1-\delta} \tau_1 \frac{\gamma_r}{\gamma_1} \quad ,$$

$$\rho = \frac{\lambda}{\mu} \left(1 + \frac{\mu s}{1-\xi} \right),$$

$$\tau_1 = (1-\xi)/\mu,$$

$$m_1' = \eta + \lambda z_o,$$

$$m_1 = \lambda\ E\left[e_r(k) \right] = \lambda(1-\delta) \sum_{k=1}^{\infty} e_r(k) \delta^{k-1}$$

$$Q = s + q \frac{1 - \gamma_{r-1}}{1-\gamma_r}$$

$$\alpha = \xi + \rho(1-\xi),$$

$$\xi = \delta \frac{1-\gamma_r}{1-\delta\gamma_r} \quad ,$$

$$\eta = \lambda s \frac{\xi}{1-\xi} + \lambda q \frac{\delta}{1-\delta} \left[1 - \frac{\gamma_r}{\gamma_1} (1 - \xi) \right].$$

avec θ_r donné par :

$$\theta_r = \frac{s/(1 - \xi)}{s/(1-\xi) + 1/\mu}$$

La démonstration est développée en détail dans l'Annexe A.

3.2. - Analyse du modèle M/G/1

L'analyse du modèle M/G/1 dans le cas classique où il n'y a pas de mécanisme d'adaptation, ce qui revient à poser $r = \infty$, a été faite par SAKATA [22] puis reprise par COCHI [4] . La technique utilisée consiste à attribuer une classe à chaque programme en attente dans la file, la classe d'un programme étant déterminée par le temps de calcul déjà reçu par ce programme mesuré par le nombre de quanta qui lui ont été alloués. La méthode de calcul reste par ailleurs la même que dans le cas M/M/1, et revient à écrire d'une part les équations de transitions d'une classe à une autre, et à évaluer d'autre part les temps d'attente causés à un programme particulier P par les programmes de chaque classe.

Toutefois, la généralité de cette méthode est limitée par le calcul numérique des grandeurs obtenues au terme des calculs. En effet, dans le cas général, le nombre de classes est infini, ce qui rend difficile sinon impossible le calcul de la solution qui fait alors intervenir des sommations à l'infini, et l'on est amené à tronquer la distribution de façon à avoir un nombre de classe fini.

Cependant pour certaines distributions particulières comme la distribution hyper-exponentielle ou la distribution d'Erlang la méthode précédente se simplifie considérablement et permet d'obtenir des expressions ne faisant intervenir que des sommes finies comme l'a montré SAKATA [22].

C'est suivant le même principe que nous calculons la solution du modèle M/G/1 avec le mécanisme d'adaptation pour la distribution hyper-exponentielle.

Une distribution hyper-exponentielle F(t) est définie comme la combinaison convexe de deux distributions exponentielles, c'est-à-dire :

$$F(t) = \beta_1(1-e^{-\mu_1 t}) + \beta_2(1-e^{-\mu_2 t}),\ \beta_1 + \beta_2 = 1, \tag{29}$$

et sa moyenne $1/\mu$ et son coefficient de variation C sont donnés par :

$$1/\mu = \beta_1/\mu_1 + \beta_2/\mu_2 \tag{30}$$

$$C^2 = 1 + 2\,\mu^2\,\beta_1\beta_2(1/\mu_1 - 1/\mu_2)^2. \tag{31}$$

Réciproquement, connaissant C et $1/\mu$, les paramètres de la distribution sont calculés de la façon suivante :

$$1/\mu_2 = b/\mu, \tag{32}$$

$$1/\mu_1 = 1/\mu + 1/\mu\ \frac{C^2 - 1}{2(1-b)}, \tag{33}$$

$$\beta = 2(1-b)^2\ \frac{1}{C^2-1 + 2(1-b)^2}, \tag{34}$$

où b est un paramètre choisi entre 0 et 1.

Si nous considérons une population de programmes dont la distribution F(t) est donnée par (29), nous pouvons remarquer que tout se passe comme si cette population était composée de deux classes de programme . La première classe C_1 en proportion β_1 avec une distribution exponentielle de moyenne $1/\mu_1$; la seconde classe C_2 en proportion β_2 avec une distribution exponentielle $1/\mu_2$.

A partir de cette remarque, nous pouvons simplement étendre la démonstration donnée pour le cas M/M/1 au cas M/G/1 avec distribution hyper-exponentielle en considérant les deux classes de programmes C_1 et C_2, et en reprenant les arguments précédents.

Les calculs et les résultats sont présentés dans l'Annexe B.

3.3. - Analyse du modèle fermé M/M/1/N

L'analyse du modèle fermé dans le cas M/M/1/N sans adaptation a été faite complètement par ADIVI et AVI-ITZHAK [1] après avoir été abordée par COFFMAN et KVISHNAMOORTHI [8] et ensuite KRISHNAMOORTHI et WOOD [16] qui en donnèrent une solution approchée. Le calcul du temps moyen de réponse R_N est relativement simple, et utilise le résultat déjà cité en [1] . Soit q et s la durée du quantum et de l'overhead, $1/\lambda$ le temps moyen de réflexion aux télétypes, N le nombre de télétypes et $1/\mu$ le temps moyen de calcul.
Le temps U.C. total X pris par un programme à chaque activation est distribué suivant :

$$X = \begin{cases} q + s & \text{avec la probabilité } \delta = e^{-\mu q}, \\ D & \text{avec la probabilité } 1 - \delta \end{cases}$$

où D a pour densité $f_D(x)$ avec :

$$(57) \qquad f_D(x) = \frac{1}{1-\delta}\,\mu e^{-\mu(x-s)} \quad , \; s \leqslant x \leqslant q+s.$$

Le temps total U.C. S pris par un programme pour son exécution s'écrit alors :

$$(58) \qquad S = (R-1)(s+q) + D$$

où R est le nombre de quanta reçus par le programme distribué géométriquement suivant :

$$(59) \qquad P(R=r) = \delta^{r-1}(1-\delta), \; r = 1,2,\ldots$$

Nous en déduisons la transformée de Laplace $\varphi_s(z)$ de S à partir des équations (57) et (59).

$$\varphi_S(z) = \sum_{r=1}^{\infty} \delta^{r-1}(1-\delta)e^{-z(q+s)(r-1)}\varphi_D(z),$$

$$\varphi_S(z) = \frac{\mu e^{-zs}(1-\delta e^{-qz})}{(\mu+z)(1-\delta e^{-(q+s)z}} \; ,$$

avec :

$$E(S) = 1/\mu + \frac{s}{1-\delta} \; .$$

Connaissant $\varphi_S(z)$, on en déduit R_N à partir de l'équation(2,Chap. Le calcul du temps de réponse moyen conditionné par la durée du calcul du programme est beaucoup plus complexe.

Nous n'avons pu généraliser le calcul de R_N, ni a fortiori celui des temps d'attente conditionnés dans le cas où le mécanisme d'adaptation entre en jeu. Aussi, l'étude du système M/M/1/N a-t-elle été conduite par simulation.

Le programme de simulation écrit en langage FORTRAN décrit à la fois le modèle fermé et le modèle ouvert. Sa validation a été réalisée en comparant les résultats obtenus à ceux qui ont pu être calculés analytiquement dans le cas M/G/1 pour toutes les mesures de performances et dans le cas M/M/1/N pour R_N dans l'hypothèse $r = \infty$. La méthode de réduction de variance [19] consistant à grouper les mesures par blocs est utilisé au cours des simulations pour fournir la variance de l'estimateur de la mesure calculée et en déduire un intervalle de confiance. Les résultats obtenus ont été calculés en simulant le traitement d'environ 50 000 programmes et le calcul des intervalles de confiance estimés ont fourni un résultat comportant une erreur inférieure dans tous les cas à 5%.

3.4. - Résultats numériques

A partir des expressions obtenues pour les modèles M/M/1 et M/G/1, et à l'aide des résultats de simulation pour le modèle M/M/1/N, nous pouvons calculer l'effet sur les mesures de performances E_r, $e_r(k)$ et θ_r du mécanisme d'adaptation. Le calcul est fait en évaluant les améliorations relatives :

$$\frac{E_\infty - E_r}{E_\infty} \quad , \frac{e_\infty(k) - e_r(k)}{e_\infty(k)} \quad , \frac{\theta_\infty - \theta_r}{\theta_\infty} ,$$

puisque pour $r = \infty$ le mécanisme d'adaptation n'entre pas en jeu. Avant de faire ces calculs, il convient de déterminer dans quelles conditions les deux systèmes avec et sans adaptation sont comparés, c'est-à-dire quel paramètre est retenu pour mesurer la charge du système. Une première solution consiste à faire les comparaisons de telle sorte que le taux d'arrivée λ des programmes soit le même dans les deux cas : l'effet du mécanisme est alors évalué à taux d'arrivée constant ce qui correspond au point de vue de l'utilisateur ; une autre façon de procéder est d'ajuster les taux d'arrivées dans les deux situations afin que la charge de l'U.C. soit la même dans les deux cas : l'effet du mécanisme d'adaptation est alors évalué à charge constante, ce qui correspond au point de vue de la gestion des ressources du système. La distinction entre ces deux approches est importante puisque, si une comparaison devait être faite sur un système réel, le résultat serait différent suivant que la charge du système serait mesurée par le taux d'arrivée des programmes ou le taux d'utilisation de l'U.C.

La calibration des deux modèles avec et sans mécanisme sera notée calibration A dans le premier cas, calibration B dans le second.

Les résultats sont présentés en faisant varier les paramètres λ ou ρ de 0.3 à 0.9 et en supposant dans toutes les expériences le temps de calcul moyen d'un programme égal à 1 seconde et la durée du quantum égale à 0.25 seconde. Le coefficient de variation du temps de calcul est pris égal à 1., 2., 5., 10. successivement. Dans le cas du modèle fermé le nombre N de télétypes est choisi égal à 5, 10 et 25. Remarquons que dans ce dernier cas il n'est pas possible de faire la calibration sur le taux d'utilisation de l'U.C. étant donné la technique de résolution du modèle utilisée.

a) Modèle ouvert

Les figures 9 et 10 montrent l'influence des paramètres r et s sur les résultats. Nous pouvons observer, à partir de la figure 9 que l'influence du seuil r décroît très vite dès que r augmente : importantes pour r = 1, les améliorations apportées par le mécanisme d'adaptation se réduisent pour r = 2 et deviennent négligeables pour r = 3. Aussi, dans la suite des expériences présentées, seules les valeurs r = 1 et r = 2 seront prises en compte.

Le paramètre s a un rôle important puisque c'est le but du mécanisme d'adaptation de réduire le temps de gestion. Aussi l'effet du seuil r d'allocation de quantum supplémentaire est-il d'autant plus important que s est grand, μ et q étant fixés par ailleurs. C'est ce qui apparaît sur la figure 10 où sont reportées les améliorations obtenues sur le temps d'attente moyen E. Nous retiendrons la valeur s = 0.03 pour tous les résultats numériques présentés.

Les résultats sur les performances du mécanisme d'adaptation, mesurées par les écarts relatifs $(E_\infty - E_r)/E_\infty$, $(e_\infty(k) - e_r(k))/e_\infty(k)$, $k = 1,2,\ldots$, $(\theta_\infty - \theta_r)/\theta_\infty$, où un écart positif signifie une réduction du temps d'attente, ou du taux de gestion, et donc une amélioration apportée par l'allocation adaptative, sont présentés sur les figures 11a, 11b et 11c pour la calibration A, et sur les figures 12a, 12b et 12c pour la calibration B.

s = 0.05 sec, q = 0.25 sec, μ = 1.0 sec^{-1}

k \ λ		0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
1	r = 1	−0.894	−0.714	−0.560	−0.425	−0.301	−0.177	−0.019	0.533
	r = 2	−0.011	−0.016	−0.018	−0.017	−0.013	−0.006	0.007	0.114
	r = 3	−0.000	−0.000	−0.001	−0.001	−0.001	0.000	0.001	0.007
3	r = 1	0.022	0.025	0.031	0.041	0.059	0.091	0.165	0.596
	r = 2	0.002	0.004	0.005	0.007	0.010	0.014	0.023	0.126
	r = 3	0.000	0.000	0.000	0.000	0.001	0.001	0.002	0.009
5	r = 1	0.090	0.091	0.093	0.098	0.109	0.133	0.197	0.608
	r = 2	0.003	0.005	0.007	0.010	0.013	0.017	0.026	0.128
	r = 3	0.000	0.000	0.000	0.000	0.001	0.001	0.002	0.009

Figure 9 : $\dfrac{e_\infty(k) - er(k)}{e_\infty(k)}$

q = 0.25 sec, μ = 1.0 sec^{-1}

s (sec) \ λ		0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
0.03	r = 1	0.025	0.031	0.037	0.045	0.055	0.071	0.101	0.189
	r = 2	0.002	0.003	0.005	0.007	0.009	0.012	0.016	0.029
0.05	r = 1	0.037	0.046	0.055	0.068	0.087	0.117	0.187	0.605
	r = 2	0.002	0.003	0.005	0.008	0.011	0.015	0.025	0.128
0.07	r = 1	0.046	0.057	0.071	0.089	0.116	0.168	0.330	—
	r = 2	0.002	0.003	0.006	0.008	0.012	0.019	0.043	—

Figure 10 : $\dfrac{E_\infty - E_r}{E_\infty}$

La première observation qui peut être faite est la réduction croissante du taux de gestion propre lorsque la charge du système, mesurée par ou ρ, augmente (figures 11b, 12b). Pour r = 1, la réduction passe d'environ 5 % pour ρ = 0.3 à plus de 10 % pour ρ = 0.9. Cet effet est évidemment indépendant de la distribution de temps de service, et sensiblement le même pour les deux modes de calibration.

La seconde observation porte sur les écarts $(E_\infty - E_r)/E_\infty$ figurant sur les figures 11a et 12a . Dans le cas d'une distribution de service exponentielle, l'amélioration est positive, quelle que soit la calibration réalisée, mais croît pour la calibration A avec la charge du système et reste sensiblement constante suivant la calibration B. Cette différence entre ces deux modes de calibrations s'explique en remarquant que l'amélioration sur E provient essentiellement de la réduction des temps de gestion qui produit une réduction de la charge ρ du système, et que la calibration B, en égalant les charges des deux modèles, à tendance à absorber en partie l'effet de cette réduction de charge. Lorsque la distribution de temps de service n'est pas exponentielle, mais hyper-exponentielle, l'effet du mécanisme d'adaptation sur le temps moyen d'attente s'inverse, et produit des écarts négatifs, en dépit d'une réduction du taux de gestion. Le résultat, qui peut sembler contraire à l'intuition peut s'expliquer en notant que l'allocation de quanta supplémentaires a pour effet d'augmenter la durée moyenne du quantum, et donc, suivant les courbes présentées sur la figure 9, Chap de rendre plus sensible sur le temps d'attente moyen l'effet de la distribution de temps de service. Dans ces conditions, la politique d'allocation adaptative provoque une augmentation du temps d'attente moyen qui n'est pas compensée par la réduction du taux de gestion. Ces effets apparaissent plus fortement sur le mode de calibration B comme on peut le constater en comparant les figures 11a et 12a puisque, comme nous l'avons vu, la réduction du taux de gestion est alors en partie annulée par la technique de calibration

Il reste à évaluer les conséquences du mécanisme d'adaptation sur la priorité implicitement accordée par la règle d'allocation par quantum aux programmes courts. Les résultats portant sur les améliorations relatives $\left[e_\infty(k) - e_r(k) \right] / e_\infty(k)$ pour k = 1,3,5 sont portés sur les figures 11c et 12c

.	λ	.3	.5	.7	.9
1.	r=1	.04	.06	0.10	*
	r=2	-	.01	.02	*
2.	r=1	-.03	-	.06	*
	r=2	-	-	.01	*
5.	r=1	-.05	-.03	.03	*
	r=2	-	-	-	*
10.	r=1	-.06	-.03	.03	*
	r=2	-	-	-	*

Figure 11a : $\frac{E_\infty - E_r}{E_r}$ (calibration A)

C	λ	.3	.5	.7	.9
1.	r=1	.05	.08	.11	*
	r=2	-	-	.01	*
2.	r=1	.05	.08	.11	*
	r=2	-	-	.01	*
5.	r=1	.05	.08	.11	*
	r=2	-	-	.01	*
10.	r=1	.05	.08	.11	*
	r=2	-	-	.01	*

Figure 11b : $\frac{\sigma_\infty - \sigma_r}{\sigma_\infty}$ (calibration A)

C	λ	.3			.5			.7			.9		
	k	1	3	5	1	3	5	1	3	5	1	3	5
1.	r=1	-.70	-	.08	-.43	.02	.08	-.19	.07	.11	*	*	*
	r=2	-.02	-	-	-.02	-	.01	-	.01	.02	*	*	*
2.	r=1	-.74	-.05	.02	-.47	-.03	.03	-.23	.03	.07	*	*	*
	r=2	-.02	-	-	-.02	-	-	-.01	-	.01	*	*	*
5.	r=1	-.75	-.06	-	-.49	-.05	-	-.25	-	.05	*	*	*
	r=2	-.02	-	-	-.03	-	-	-.01	-	.01	*	*	*
10.	r=1	-.76	-.07	-	-.49	-.05	-	-.25	-	.05	*	*	*
	r=2	-.02	-	-	-.03	-	-	-.02	-	.01	*	*	*

Figure 11c : $\dfrac{\ell_\infty(k) - \ell_r(k)}{\ell_\infty(k)}$ (calibration A)

C \ ρ		.3	.5	.7	.9
1.	r=1	.03	.03	.03	.03
	r=2	-	-	-	.01
2.	r=1	-.04	-.03	-.02	-.01
	r=2	-	-	-	-
5.	r=1	-.07	-.06	-.04	-.02
	r=2	-	-	-	-
10.	r=1	-.07	-.06	-.05	-.03
	r=2	-	-	-	-

Figure 12a : $\frac{E_\infty - E_r}{E_\infty}$ (calibration B)

C \ ρ		.3	.5	.7	.9
1.	r=1	.04	.07	.10	.13
	r=2	-	-	-	.01
2.	r=1 / r=2	idem			
5.	r=1 / r=2	idem			
10.	r=1 / r=2	idem			

Figure 12b : $\frac{\mathcal{O}_\infty - \mathcal{O}_r}{\mathcal{O}_\infty}$ (calibration B)

C	ρ	.3			.5			.7			.9		
	k	1	3	5	1	3	5	1	3	5	1	3	5
1.	r=1	-.77	-	-	-.53	-	.06	-.35	-	.05	-.21	-	.04
	r=2	-.02	-	-	-.02	-	-	-.02	-	-	-	.01	.02
2.	r=1	-.80	-.05	-.02	-.58	-.05	-	-.39	-.05	-	-.25	-.04	-
	r=2	-.02	-	-	-.02	-	-	-.02	-	-	-.01	-	-
5.	r=1	-.82	-.07	-	-.59	-.08	-.01	-.41	-.07	-.02	-.26	-.05	-.02
	r=2	-.02	-	-	-.03	-	-	-.02	-	-	-	-	-
10.	r=1	-.82	-.08	-	-.59	-.08	-.02	-.42	-.08	-.02	-.27	-.06	-.02
	r=2	-.02	-	-	-.02	-	-	-.02	-	-	-.02	-	-

Figure 12c : $\frac{\ell_\infty(k) - \ell_r(k)}{\ell_\infty(k)}$ (calibration B)

Comme il pouvait être prévu, le mécanisme d'adaptation tend à réduire le traitement prioritaire donné aux programmes ne demandant qu'un quantum (k=1), réduction d'autant plus forte que le coefficient de variation de la distribution de temps de service est élevé. En revanche pour k = 3 et 5 on observe, principalement pour le mode de calibration A, pour les raisons exposées plus haut, une amélioration de 5 à 10 %. Par ailleurs, l'effet du coefficient de variation et du mode de calibration sur ces résultats est semblable à ceux observés pour les temps d'attente moyen.

Nous pouvons résumer ces observations en notant que l'effet du mécanisme d'adaptation augmente avec la charge du système, mais diminue lorsque le coefficient de variation de la distribution de temps de service augmente. Les deux modes de calibration mettent en évidence la modification des priorités accordées aux programmes suivant leur durée apportée par la règle d'allocation de quantum supplémentaire.

b) Modèle fermé

La différence entre le système ouvert et le système fermé tient essentiellement dans le processus d'arrivée des programmes vers la file d'attente de l'U.C., les deux systèmes se comportant de façon identique pour le reste. Aussi, l'effet de la distribution de temps de service sur les performances du mécanisme d'adaptation ayant été étudié dans le paragraphe précédent, nous contenterons-nous d'observer les performances de ce mécanisme dans le cadre du système fermé M/M/1/N.

Les résultats obtenus par simulation sont rassemblés sur la figure 13 pour r=1, q=0.25 sec., $\mu = 1.\text{sec.}^{-1}$, et s=0.03 sec. Les résultats d'améliorations pour r=2 étant de l'ordre de grandeur des erreurs de simulation n'ont pas été reportés. La comparaison avec les résultats obtenus pour le modèle ouvert montre une similitude de comportement, et les ordres de grandeurs des écarts sur les performances causés par l'allocation adaptative sont conservés, même pour les valeurs élevées de la charge du système pour lesquelles, comme nous l'avons montré, l'approximation par source infinie cesse d'être valable lorsque les valeurs absolues des temps de réponse sont considérées. Cette observation indique que le modèle à source infinie est apte à prédire les améliorations relatives apportée par une règle d'allocation, même si les valeurs absolues fournies sont incorrectes.

N	5			10			25		
$1/\lambda$	4.	8.	16.	8.	16.	32.	32.	64.	128.
$(E_\infty - E_r)/E_\infty$	.04	.05	-	.01	.05	-.03	.05	.03	.06
$(\mathcal{O}_\infty - \mathcal{O}_r)/\mathcal{O}_\infty$	.11	.07	.05	.12	.09	.04	.11	.06	.03
$\frac{\ell_\infty(k) - \ell_r(k)}{\ell_\infty(k)}$ k=1	-.52	-.76	-.80	-.38	-.50	-.94	-.31	-.73	-1.08
k=3	.01	.03	-.14	-.02	-	-.05	.01	.04	.13
k=5	.06	.09	-	.04	.04	-.01	.10	.09	.06
ρ_∞	.85	.57	.32	.42	.62	.33	.79	.43	.22
ρ_r	.84	.58	.33	.92	.62	.34	.78	.43	.21

Figure 13 : modèle M/M/1/N r=1

5. - CONCLUSIONS

Deux types de conclusions peuvent être dégagées à la suite de l'étude présentée. Les premières conclusions portent sur la technique d'analyse utilisée et sur les extensions possibles de cette approche ; le deuxième type de conclusion intéresse le problème concret analysé et les résultats que la modélisation a permis de dégager.

Comme nous l'avons vu en introduction, les modèles d'allocation de l'U.C. dans le contexte d'un système à partage de temps ont donné lieu a un nombre considérable d'études. Elles consistent à examiner diverses variantes du modèle de base en élargissant les hypothèses ou en modifiant certains mécanismes, mais, dans tous les cas, la démarche d'analyse reste sensiblement la même : le cheminement d'un programme dans le système est examiné depuis son entrée jusqu'à la fin de son exécution, et les temps d'attente causés à ce programme par les autres programmes à chaque étape de son traitement sont calculés. L'analyse n'est donc pas globale, mais plutôt locale, et il faut attendre la fin du calcul pour avoir un résultat utilisable. Les solutions montrent également que le résultat obtenu ne peut être interprété simplement, et qu'il faut avoir recours au calcul numérique des formules pour connaître le comportement du système modélisé. Cette difficulté est évidemment d'autant plus sensible que le modèle est complexe. On peut donc considérer que ce type de modèles n'apporte qu'une partie de ce que l'on attend en général d'une démarche de modélisation -une compréhension globale quantitative des phénomènes ; un résultat explicite- et il sera souvent plus profitable de considérer des modèles plus simples qui donnent des lois de comportement aisément interprétables. En d'autres termes, si l'on considère l'évolution des modèles d'allocation U.C., la connaissance supplémentaire apportée par chaque nouveau modèle suit une loi de rendements décroissants, et il ne nous paraît donc pas essentiel de poursuivre l'étude de nouveaux modèles de ce type autrement que dans un but de formation.

Les principales conclusions sur les performances du mécanisme d'allocation ont été présentées lors de la discussion des exemples numériques Le point important nous paraît être la sensibilité des résultats à l'hypothè

de distribution puisque, pratiquement, deux effets opposés sont obtenus suivant que la distribution des temps de service est exponentielle ou hyperexponentielle. Dans leur ensemble les améliorations observées sont relativement limitées, et l'on peut penser qu'un mécanisme moins sévère d'allocation supplémentaire de quantum serait plus efficace : une variante pourrait consister à allouer un quantum supplémentaire de durée $q' > q$, auquel cas l'analyse présentée reste encore valable sans d'importantes modifications.

Annexe A

Démonstration (modèle M/M/1)

La méthode d'analyse consiste à suivre le cheminement dans le système étudié d'un programme particulier noté P qui demande k quanta de temps de calcul, et qui arrive dans le système lorsque celui ci est en état d'équilibre. Le cheminement de P dans le système est en particulier caractérisé par le nombre de passes qu'il effectue, c'est-à-dire le nombre de fois où l'U.C. lui est alloué. Soit n, $1 \leqslant n \leqslant k$, le nombre de passes d'un programme demandant k quanta, et $p_n(k)$ la probabilité qu'un tel programme fasse n passes

En raison de l'hypothèse faite sur le processus d'arrivée, la probabilité γ_r qu'à la fin d'un quantum, un quantum supplémentaire soit alloué au programme est indépendante du quantum considéré, et est donnée par (1) .

Nous en déduisons les probabilités $p_n(k)$

$$(2) \qquad p_n(k) = C_{k-1}^{n-1} \gamma_r^{k-n} (1-\gamma_r)^{n-1}, \quad 1 \leqslant n \leqslant k.$$

Nous définissons $w_n(k)$ comme le temps moyen d'attente d'un programme demandant k quanta et effectuant exactement n passes. D'après la définition de $p_n(k)$ nous avons alors :

$$(3) \qquad e_r(k) = \sum_{n=1}^{k} p_n(k) w_n(k).$$

Afin de calculer $w_n(k)$, les programmes causant une attente au programme P sont partitionnés en deux sous-ensembles disjoints $\mathcal{P}_1$ et $\mathcal{P}_2$ définis de la façon suivante :

D1 - Un programme appartient à $\mathcal{P}_1$ s'il était en cours d'exécution ou de chargement à l'arrivée du programme P, ou s'il arrive durant l'exécution ou le chargement d'un programme appartenant à $\mathcal{P}_1$;

D2 - Un programme appartient à $\mathcal{P}_2$ s'il était en attente à l'arrivée de P, ou s'il arrive durant l'exécution ou le chargement d'un programme appartenant à $\mathcal{P}_2$.

Il reste à évaluer les temps d'attente moyens $T1_{k,n}$ et $T2_{k,n}$ dus aux programmes appartenant respectivement à $\mathcal{P}_1$ et $\mathcal{P}_2$, puisque

$$w_n(k) = T1_{k,n} + T2_{k,n} \qquad (4)$$

Evaluation de $T2_{k,n}$

Soit y_i le temps moyen d'attente de P à sa $i^{ième}$ passe, ce temps excluant, dans le cas de la 1ère passe, le temps d'attente dû au programme éventuellement en cours de chargement ou d'exécution lors de l'arrivée de P, ceci d'après D2. Nous avons alors :

$$T2_{k,n} = \sum_{i=1}^{n} y_i .$$

Si m_i est le nombre moyen de programmes en attente devant le programme à la $i^{ième}$ passe, et en notant τ le temps d'exécution moyen, par programme et par passe, alors :

$$y_i = m_i(\tau + s),$$

et

$$T2_{k,n} = \sum_{i=1}^{n} m_i(\tau+s). \qquad (5)$$

Nous définissons ξ comme la probabilité qu'un programme, ayant achevé une passe, retourne dans la file d'attente A. Soit $k_{n,i-1}$ le nombre de quanta reçu par P à sa $(i-1)^{ième}$ passe, ce qui entraîne qu'il s'est produit au moins r arrivées dans chacune des $k_{n,i-1} - 1$ quanta précédents, et soit N_r le nombre moyen d'arrivées durant chacun de ces $k_{n,i-1} - 1$ quanta. Il vient :

$$N_r = E\left\{\text{nombre d'arrivées durant un quantum} \,/\, \text{il y a au moins r arrivées}\right\},$$

d'où :

$$N_r = \frac{1}{\gamma_r} \sum_{p=r}^{\infty} p\, e^{-\lambda q} \frac{(\lambda q)^p}{p!},$$

et :

$$N_r = \lambda q \frac{\gamma_{r-1}}{\gamma_r} \tag{6}$$

Pendant le dernier quantum des $k_{n,i-1}$ quanta, le nombre d'arrivées est inférieur à r. Nous pouvons calculer de même que précédemment le nombre moyen d'arrivées N'_r donné par :

$$N'_r = \frac{1}{1-\gamma_r} \sum_{p=o}^{r-1} p\, e^{-\lambda q} \frac{(\lambda q)^p}{p!} = \lambda q \frac{1-\gamma_{r-1}}{1-\gamma_r} \tag{7}$$

De plus, λs programmes en moyenne vont arriver pendant le chargement de P à la passe précédente, et $\lambda\, m_{i-1}(\tau+s)$ programmes en moyenne pendant l'exécution des programmes en attente devant P à la $i^{ième}$ passe. Il s'en déduit le nombre moyen m_i de programmes en attente devant P à la $i^{ième}$ passe.

$$m_i = \xi m_{i-1} + \lambda m_{i-1}(\tau+s) + \lambda s + N_r E\left\{k_{n,i-1}\right\} + N'_r.$$

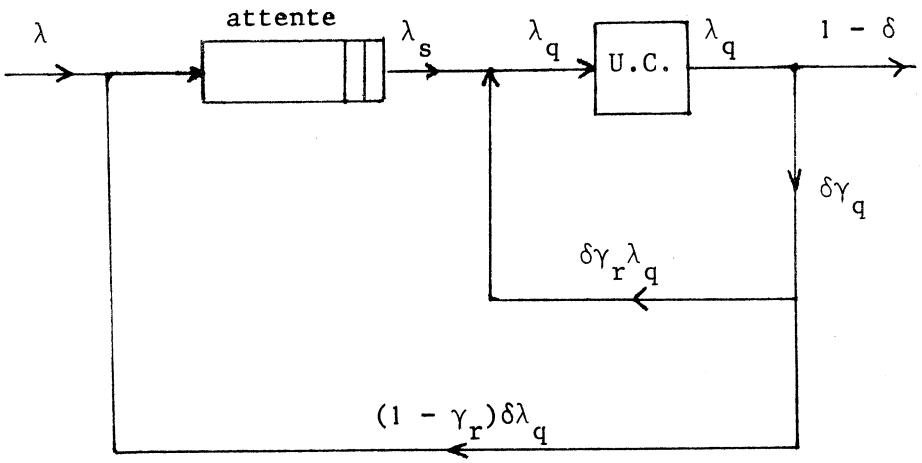


Figure 8 : Flux des quanta et des opérations de chargements

Suivant COFFMAN [5] nous pouvons faire l'hypothèse que $E[k_{n,i}] = \frac{k}{n}$, quel que soit i. En posant :

$$\alpha = \xi + \lambda(\tau+s), \tag{8}$$

$$Q = s + \frac{1 - \gamma_{r-1}}{1 - \gamma_r} q, \tag{9}$$

nous obtenons ,

$$m_i = \alpha\, m_{i-1} + (\frac{k}{n} - 1)N_r + \lambda Q,$$

d'où, par récurrence ,

$$m_i = \alpha^{i-1}\, m_1 + \left[(\frac{k}{n} - 1)N_r + \lambda Q \right] \frac{1 - \alpha^{i-1}}{1 - \alpha} . \tag{10}$$

L'évaluation de m_1 sera faite à la fin de la démonstration. Pour calculer ξ considérons le schéma représenté sur la figure 8 où λ_s et λ_q désignent le taux d'arrivée des overheads et des quanta respectivement. Nous en déduisons, à partir de la définition de δ et γ_r les relations suivantes qui expriment la conservation des flux de quanta et d'overheads :

$$\lambda_q = \lambda_s + \delta\gamma_r\, \lambda_q \ ;$$

$$\xi\lambda_s = \lambda_q\, \delta(1-\gamma_r).$$

D'où l'expression de ξ :

$$\xi = \delta\, \frac{1 - \gamma_r}{1 - \delta\gamma_r} \tag{11}$$

Le calcul de τ peut être fait de plusieurs façons. En remarquant que le nombre moyen de passes effectuées par un programme est donné par $1/(1-\xi)$ puisque la probabilité qu'à la fin d'une passe l'exécution du programme ne soit pas terminée est ξ, le temps d'exécution moyen par passe est obtenu simplement en divisant le temps d'exécution moyen total $1/\mu$ par le nombre moyen de passes $1/(1-\xi)$, d'où :

$$\tau = \frac{1}{\mu}(1-\xi) \tag{12}$$

Le même résultat peut être obtenu par un calcul détaillé en calculant le nombre moyen de quanta par passe. La probabilité qu'un programme obtienne j quanta dans une seule passe s'écrit, à partir des définitions de γ_r et δ, $\gamma_r^{j-1}(1-\gamma_r)\delta^j$, dans le cas où l'exécution ne se termine pas à la fin de la passe, et $\gamma_r^{j-1}(1-\delta)\delta^{j-1}$ dans le cas contraire. Nous avons donc :

$$\tau = \sum_{j=1}^{\infty} \left[jq\ \gamma_r^{j-1}(1-\gamma_r)\delta^j + \gamma_r^{j-1}(1-\delta)\delta^{j-1}\left[(j-1)q + \varepsilon_1 \right] \right],$$

où ε_1 représente le temps moyen pris par un programme auquel un quantum q a été alloué sachant que le programme achève son exécution avant la fin du quantum. Il vient :

$$\varepsilon_1 = \frac{1}{1-\delta} \int_0^q dE\ (t) = 1/\mu - \frac{\delta}{1-\delta} q, \tag{13}$$

d'où à partir du calcul de la sommation intervenant dans l'expression de τ :

$$\tau = \frac{1}{\mu} \frac{1-\delta}{1-\delta\gamma_r} = \frac{1}{\mu}(1-\xi).$$

Pour la suite des calculs, il sera utile de considérer le facteur de charge du système désigné par ρ, et défini comme le rapport de l'intervalle moyen entre deux arrivées consécutives à la durée moyenne de traitement T d'un programme, chargements et exécution compris. Comme précédemment, le calcul peut se faire de deux façons, suivant que le temps moyen de traitement d'un programme est calculé directement à partir de la loi de distribution F(t) et des probabilités donnant le nombre de passes suivant le nombre de quanta du programme, ce qui donne alors :

$$T = \sum_{k=1}^{\infty} \int_{(k-1)q}^{kq} \sum_{i=1}^{k} (t+is)\, p_i(k)\, dF(t),$$

ou en écrivant que ce temps traitement moyen est égal au temps d'exécution moyen augmenté de l'overhead total moyen. L'overhead total moyen est égal au nombre de passes moyen multiplié par s, et il vient :

$$T = \frac{1}{\mu} + \frac{s}{1-\xi},$$

d'où :

$$\rho = \lambda T = \frac{\lambda}{\mu}\left(1 + \frac{\mu s}{1-\xi}\right). \qquad (14)$$

On peut vérifier que le premier mode de calcul conduit au même résultat. Remarquons que, comme dans le cas précédent, les deux modes de calcul sont équivalents à cause de l'hypothèse faite sur F(t) qui entraîne que le comportement d'un programme à l'intérieur d'une passe est toujours le même, quel que soit le numéro de la passe.

A partir de (8) et (12), nous avons :

$$\alpha = \xi + \rho(1-\xi) \qquad (15)$$

et en subsitutant (10) dans (5), il vient :

$$T2_{k,n} = (\tau+s) \sum_{i=1}^{n} \left\{ \alpha^{i-1} m_1 + \left[\left(\frac{k}{n} - 1\right) N_r + \lambda Q \right] \frac{1 - \alpha^{i-1}}{1 - \alpha} \right\},$$

et :

$$(16) \qquad T2_{k,n} = \frac{\tau+s}{1-\alpha} \left\{ (k-n) N_r + n\lambda Q + (1-\alpha^n) \left[m_1 - \frac{\left(\frac{k}{n} - 1\right) N_r + \lambda Q}{1 - \alpha} \right] \right\}$$

Calcul de $T1_{k,n}$

Le temps $T1_{k,n}$ représente le temps moyen d'attente du au programme en cours de traitement à l'arrivée de P. Considérons d'abord le temps moyen z_o nécessaire à l'achèvement du traitement en cours à l'arrivée de P. Trois cas sont possibles suivant que P arrive pendant un chargement, pendant un quantum partiel, c'est-à-dire un quantum pendant lequel le programme en cours d'exécution achève son calcul, et pendant un quantum complet. Dans ce dernier cas, il y aura eu alors moins de r-1 autres arrivées que P pendant le quantum avec la probabilité $1 - \gamma_r/\gamma_1$ et au moins r-1 autres arrivées avec la probabilité γ_r/γ_1. Notons Π_s, Π_{pq} et Π_{cq} les probabilités d'arrivée de P durant un chargement, un quantum partiel et un quantum complet, respectivement. Nous pouvons alors écrire :

$$(17) \qquad z_o = \Pi_s(\bar{s}+\tau) + \Pi_{pq} \bar{q} + \Pi_{cq}\left(1-\frac{\gamma_r}{\gamma_1}\right) \frac{q}{2} + \Pi_{cq} \frac{\gamma_r}{\gamma_1} \left(\frac{q}{2} + \tau\right),$$

où $\bar{s}$ et $\bar{q}$ représentent le temps moyen de chargement et de quantum restant dans le premier et le second cas, respectivement.

Calcul de Π_s et $\bar{s}$

La probabilité Π_s est égal au flux d'arrivée d'overheads λ_s multiplié par la durée d'un overhead s, d'où, d'après (11) :

$$\Pi_s = \lambda_s s = \frac{\lambda s}{1-\xi}$$

La durée de l'overhead étant constante, et l'arrivée des programmes de l'extérieur et les déchargements des événements indépendants, nous avons $\bar{s} = \frac{s}{2}$.

Calcul de Π_{pq} et $\bar{q}$

La probabilité Π_{pq} est égal au flux d'arrivée des quanta partiels, c'est-à-dire au flux d'arrivée des programmes λ multiplié par la durée moyenne ε_1 d'un quantum partiel donnée par (13), d'où :

$$\Pi_{pq} = \lambda\varepsilon_1 = \frac{\lambda}{\mu}\left(1 - \frac{\mu\delta q}{1-\delta}\right) \tag{18}$$

Pour le calcul de $\bar{q}$ nous utilisons le lemme suivant :

Lemme

Soit une file d'attente avec un processus d'arrivée poissonnien et une loi de service de distribution H(t) dont les deux premiers moments sont notés x_1 et x_2. Alors la densité de probabilité $g(\tau)$ du temps τ restant dans le service à l'arrivée d'un nouveau client et sa moyenne m sont données par :

$$g(\tau) = \frac{1 - H(\tau)}{x_1}$$

$$m = \frac{x_2}{2\,x_1} \tag{19}$$

Dans le cas qui nous intéresse, la distribution de service est celle de la durée d'un quantum partiel dont le premier moment ε_1 a déjà été calculé par l'équation (13) et dont le second moment ε_2 est donné par :

$$\varepsilon_2 = \frac{1}{1-\delta}\int_0^{q-} \tau^2\, dF(\tau) = \frac{2}{\mu}\,\varepsilon_1 - \frac{\delta}{1-\delta}\,q^2. \tag{20}$$

Nous avons donc finalement d'après (18) et (19) :

$$\Pi_{pq}\ \bar{q} = \lambda\ \frac{\varepsilon_2}{2}$$

Calcul de Π_{cq}

Comme précédemment Π_{cq} est égal au produit du taux d'arrivée des quanta complets par la durée de ces quanta, c'est-à-dire q. Le taux d'arrivée des quanta λ_q est donné à partir de (11) par :

$$\lambda_q = \frac{\lambda}{1-\delta},$$

et le taux d'arrivée des quanta complets λ_{pq} s'en déduit directement par définition de δ d'où :

$$\lambda_{pq} = \lambda\ \frac{\delta}{1-\delta}\ ,$$

et finalement :

(21) $$\Pi_{cq} = \lambda q\ \frac{\delta}{1-\delta}\ .$$

Nous pouvons maintenant écrire l'expression de z_o. Il vient :

(22) $$z_o = \frac{\lambda s}{1-\xi}\ \left(\frac{s}{2} + \tau\right) + \frac{\lambda}{\mu^2}\ \left(1 - \frac{\mu\delta q}{1-\delta}\right) + \frac{\lambda q\delta}{1-\delta}\ \tau\ \frac{\gamma r}{\gamma_1}\ .$$

En notant η la probabilité que le programme en cours de traitement à l'arrivée de P fera une nouvelle passe, nous pouvons calculer le nombre moyen m_1' de programmes derrière P à sa première passe, et donc devant lui à la seconde passe. Il vient simplement :

$$m_1' = \eta + \lambda z_o$$

Pour le calcul de η, nous procédons comme pour z_o en remarquant que P peut arriver durant un chargement avec la probabilité Π_s ou durant un quantum complet avec la probabilité Π_{cq}. Dans le premier cas, le programme en cours de chargement fera une nouvelle passe avec la probabilité ξ ; dans de second cas il reviendra directement dans la file pour une nouvelle passe avec la probabilité $1 - \gamma_r/\gamma_1$ ou obtiendra un quantum supplémentaire avec la probabilité γ_r/γ_1 et fera ensuite une nouvelle passe avec la probabilité ξ. Nous avons donc :

$$\eta = \Pi_s \xi + \Pi_{cq} (1 - \gamma_r/\gamma_1) + \Pi_{cq} \gamma_r/\gamma_1 \xi,$$

soit :

$$\eta = \lambda s \frac{\xi}{1-\xi} + \lambda q \frac{\delta}{1-\delta} \left[1 - \frac{\gamma_r}{\gamma_1} (1-\xi) \right]. \tag{23}$$

Notons m'_i le nombre moyen de programmes de type $\mathcal{P}_1$ en attente devant P à la $i^{ième}$ passe. Comme pour le calcul de $T2_{k,n}$, nous avons, en remarquant que ces programmes reçoivent au plus n-1 passes :

$$T1_{k,n} = z_o + \sum_{i=1}^{n-1} m'_i(\tau+s) ,$$

et nous pouvons écrire :

$$m'_i = \xi\, m'_{i-1} + \lambda m'_{i-1}(\tau+s),$$

ou

$$m'_i = \alpha\, m'_{i-1} = \alpha^{i-1} m'_1. \tag{24}$$

Nous en déduisons :

$$T1_{k,n} = z_o + m'_1(\tau+s) \frac{1 - \alpha^{n-1}}{1 - \alpha} \tag{25}$$

Il reste à calculer $w_n(k)$ à partir de (4), (16) et (25) ce qui donne, en remarquant que $(\tau+s) = \frac{\rho}{\lambda}(1-\xi)$,

$$(26) \qquad w_n(k) = z_o + \frac{\rho}{\lambda}\frac{1-\xi}{1-\alpha}\Big\{(1-\alpha^{n-1})m_1' + (1-\alpha^n)m_1 + (k-n)N_r + n\lambda Q - \Big\{(k/n-1)N_r + \lambda Q\Big\}\frac{1-\alpha^n}{1-\alpha}\Big\}$$

En multipliant (26) par $p_n(k)$ donné par (2) et en substituant dans (3), il vient, après avoir effectué la sommation :

$$(27) \qquad e_r(k) = z_o + \frac{\rho}{\lambda}\frac{1-\xi}{1-\alpha}\Big\{m_1\Big[1-\alpha(\gamma_r+\alpha(1-\gamma_r))^{k-1}\Big] + m_1'\Big[1-(\gamma_r+\alpha(1-\gamma_r))^{k-1}\Big] + \lambda q\frac{r-1}{(1-\alpha)(1-\gamma_r)}\Big[(\gamma_r+\alpha(1-\gamma_r))^{k-1} + (k-1)(1-\alpha)(1-\gamma_r) - 1\Big] + \lambda\frac{Q}{1-\alpha}\Big[\alpha\Big[\gamma_r+\alpha_r(1-\gamma_r)\Big]^{k-1} + \Big[\gamma_r + k(1-\gamma_r)\Big](1-\alpha) - 1\Big]\Big\}.$$

Il reste à évaluer m_1. A partir de la formule de LITTLE [12] , nous pouvons écrire :

$$m_1 = \lambda E_r,$$

avec :

$$E_r = \sum_{k=1}^{\infty} e_r(k)\,\Pi(k),$$

où $\Pi(k)$ représente la probabilité qu'un programme demande k quanta.

$$(28) \qquad \Pi(k) = \int_{(k-1)q}^{kq} dF(t) = \delta^{k-1}(1-\delta).$$

Nous obtenons alors :

$$E_r = z_o + \frac{\rho}{\lambda} \frac{(1-\xi)}{1-\delta(\gamma_r+\alpha(1-\gamma_r))} \Big\{ (1-\delta\gamma_r)m_1 + \delta(1-\gamma_r)m_1' + \frac{1-\gamma_r}{1-\delta} \lambda\delta \left[q\delta\gamma_{r-1} + Q(1-\delta\gamma_r) \right] \Big\},$$

ce qui permet de calculer m_1 en écrivant :

$$E_r = m_1/\lambda.$$

Afin d'achever le calcul des mesures de performances retenues, il reste à évaluer le taux d'overhead θ_r. D'après la définition de θ_r, et les équations (12) et (14), il vient :

$$\theta_r = \frac{s/(1-\xi)}{s/(1-\xi) + 1/\mu}$$

Annexe B

Démonstration (modèle M/G/1)

Nous considérons de nouveau un programme P arrivant lorsque le système est à l'équilibre, et la partition des programmes en attente en deux ensembles $\mathcal{P}_1$ et $\mathcal{P}_2$. Les quantités définies dans la démonstration précédente sont affectées d'un indice supérieur j, j = 1,2, lorsqu'elles se rapportent aux programmes de classe j. En reprenant la démonstration à l'évaluation de $T2_{k,n}$, il vient :

$$T2_{k,n} = \sum_{j=1}^{2} \sum_{i=1}^{n} m_i^j \, (\tau_j + s), \tag{35}$$

De même l'équation liant les m_i^j aux m_{i-1}^j , j=1,2, s'obtient en considérant les arrivées de chacune des classes pendant le temps de traitement des programmes, soit :

$$m_i^1 = \xi_1 \, m_{i-1}^1 + \beta_1 \lambda \left\{ m_{i-1}^1 (\tau_1 + s) + m_{i-1}^2 (\tau_2 + s) + q \, \frac{\gamma_{r-1}}{\gamma_r} \, E\left[k_{n,i-1} - 1\right] + s + q \, \frac{1 - \gamma_{r-1}}{1 - \gamma_r} \right\},$$

$$m_i^2 = \xi_2 \, m_{i-1}^2 + \beta_2 \left\{ m_{i-1}^1 (\tau_1 + s) + m_{i-1}^2 (\tau_2 + s) + q \, \frac{\gamma_{r-1}}{\gamma_r} \, E\left[k_{n,i-1} - 1\right] + s + q \, \frac{1 - \gamma_{r-1}}{1 - \gamma_r} \right\}.$$

Posons :

$$\nu = \lambda \left[q \frac{\gamma_{r-1}}{\gamma_r}(k/n-1) + Q \right],$$

$$b_1 = \lambda \beta_1(\tau_2+s),$$

$$b_2 = \lambda \beta_2(\tau_1+s),$$

$$a_j = \xi_j + \beta_j \lambda (\tau_j+s) \quad , \; j = 1, 2,$$

alors, en reprenant les notations précédentes,

$$m_i^1 = a_1 m_{i-1}^1 + b_1 m_{i-1}^2 + \nu\beta_1, \tag{36}$$

$$m_i^2 = a_2 m_{i-1}^2 + b_2 m_{i-1}^1 + \nu\beta_2, \tag{37}$$

ou, sous forme matricielle,

$$\bar{m}_i = \bar{\bar{\alpha}}\, \bar{m}_{i-1} + \nu\bar{\beta} \tag{38}$$

avec :

$$\bar{m}_i = \begin{pmatrix} m_i^1 \\ m_i^2 \end{pmatrix}; \quad \bar{\beta} = \begin{pmatrix} \beta_1 \\ \beta_2 \end{pmatrix}; \quad \bar{\bar{\alpha}} = \begin{pmatrix} a_1 & b_1 \\ b_2 & a_2 \end{pmatrix}$$

Soit α_1 et α_2 les valeurs propres de $\bar{\bar{\alpha}}$, et Q la matrice de passage, avec :

$$\begin{cases} \alpha_1 = \dfrac{a_1 + a_2 + \sqrt{\Delta}}{2}, \\ \alpha_2 = \dfrac{a_1 + a_2 - \sqrt{\Delta}}{2}, \\ \Delta = (a_1+a_2)^2 + 4b_1b_2 \end{cases} \tag{39}$$

$$(40) \qquad Q = \begin{pmatrix} 1 & 1 \\ \dfrac{a_2 - a_1 + \sqrt{\Delta}}{2b_1} & \dfrac{a_2 - a_1 - \sqrt{\Delta}}{2b_1} \end{pmatrix},$$

nous pouvons alors écrire, à partir de (38) :

$$(41) \qquad \bar{m}_i = Q \begin{pmatrix} \alpha_1^{i-1} & \\ & \alpha_2^{i-1} \end{pmatrix} Q^{-1} \bar{m}_1 + Q \begin{pmatrix} \dfrac{1-\alpha_1^{i-1}}{1-\alpha_1} & \\ & \dfrac{1-\alpha_2^{i-1}}{1-\alpha_2} \end{pmatrix} Q^{-1} \bar{\beta}$$

Comme précédemment l'évaluation de $\bar{m}_1$ est reportée à la fin de la démonstration. Le calcul des quantités ξ_j, τ_j, ρ_j, $j = 1,2$, se fait directement, d'où, en posant :

$$(42) \qquad \delta_j = e^{-\mu_j q},$$

nous avons :

$$(43) \qquad \xi_j = \delta_j \frac{1 - \gamma_r}{1 - \delta_j \gamma_r},$$

$$(44) \qquad \tau_j = (1-\xi_j)/\mu_j,$$

$$(45) \qquad \rho_j = \lambda(1+\mu_j s/(1-\xi_j))/\mu_j \ , \ j = 1,2.$$

Nous pouvons maintenant calculer $T2_{k,n}$ d'après (35). Il vient :

$$T2_{k,n} = \left[\tau_1+s,\ \tau_2+s\right] \left[Q \begin{pmatrix} \frac{1-\alpha_1^n}{1-\alpha_1} & 0 \\ 0 & \frac{1-\alpha_2^n}{1-\alpha_2} \end{pmatrix} Q^{-1}\ \bar{m}_1 \right.$$

$$\left. + \nu Q \begin{pmatrix} \frac{n(1-\alpha_1) - (1-\alpha_1^n)}{(1-\alpha_1)^2} & 0 \\ 0 & \frac{n(1-\alpha_2) - (1-\alpha_2^n}{(1-\alpha_2)^2} \end{pmatrix} Q^{-1}\ \bar{\beta} \right]$$

Pour le calcul de z_o, nous reprenons le raisonnement utilisé dans la première démonstration, en distinguant à chaque étape la part due aux programmes de chaque classe. Ainsi, dans le cas où P arrive durant le chargement d'un programme, la probabilité sera de $\beta_1 \lambda s/(1-\xi_1)$ si le programme en chargement est de classe C_1, et de $\beta_2 \lambda s/(1-\xi_2)$ si le programme est de classe C_2. On en déduit le temps d'attente dans ce cas qui s'écrit :

$$\beta_1 \frac{\lambda s}{1-\xi_1} \left(\frac{s}{2} + \tau_1\right) + \beta_2 \frac{\lambda s}{1-\xi_2} \left(\frac{s}{2} + \tau_2\right),$$

soit :

$$\sum_{j=1}^{2} \beta_j \frac{\lambda s}{1-\xi_j} \left(\frac{s}{2} + \tau_j\right).$$

Le calcul serait identique dans les autres cas et conduit à une expression de z_o de la forme :

$$z_o = \sum_{j=1}^{2} \beta_j\ z_o^j, \tag{46}$$

où z_o^j est donné par une équation identique à (22) dans laquelle les quantités ξ, τ, μ et δ sont indicées par j.

Notons η la probabilité qu'un programme en cours de traitement à l'arrivée de P fasse une nouvelle passe. En raisonnant comme au dessus, il vient :

$$\eta = \sum_{j=1}^{2} \beta_j \eta^j ,$$

où η^j est donné par une équation identique à (23). Les nombres moyens $m_1'^j$ étant définis comme précédemment m_1', nous en déduisons :

$$m_1'^j = \beta_j(\eta^j + \lambda z_o), \quad j=1,2. \tag{47}$$

Il reste à écrire les relations entre $m_1'^j$ et $m_{i-1}'^j$, $j=1,2$. Nous avons :

$$m_i'^1 = \xi_1 \, m_{i-1}'^1 + \lambda\beta_1 \left[m_{i-1}'^1(\tau_1+s) + m_{i-1}'^2(\tau_2+s) \right] ,$$

$$m_i'^2 = \xi_2 \, m_{i-1}'^2 + \lambda\beta_2 \left[m_{i-1}'^1(\tau_1+s) + m_{i-1}'^2(\tau_2+s) \right] .$$

soit, en notation matricielle :

$$\bar{m}_i' = \bar{\bar{\alpha}} . \bar{m}_{i-1}' , \tag{48}$$

et :

$$\bar{m}_i' = Q \begin{pmatrix} \alpha_1^{i-1} & 0 \\ 0 & \alpha_2^{i-1} \end{pmatrix} Q^{-1} \, \bar{m}_1' . \tag{49}$$

Nous avons alors :

$$Tl_{k,n} = z_o + \left[\tau_1+s, \ \tau_2+s \right] \left[Q \begin{pmatrix} \frac{1-\alpha_1^{n-1}}{1-\alpha_1} & 0 \\ 0 & \frac{1-\alpha_2^{n-1}}{1-\alpha_2} \end{pmatrix} Q^{-1} \, m_1' \right] . \tag{50}$$

Le calcul de $e_r(k)$ se fait à partir de $w_n(k) = T1_{k,n} + T2_{k,n}$ et des probabilités $p_n(k)$ données par (2). Il vient :

$$(51) \qquad e_r(k) = z_o + \left[\tau_1+s,\ \tau_2+s\right]\left[Q\ U_k\ Q^{-1}\bar{m}_1 + Q\ V_k\ Q^{-1}\bar{m}_1 + Q\ W_k\ Q^{-1}\bar{\beta}\right]$$

où U_k, V_k et W_k sont des matrices diagonales d'éléments u_k^j, v_k^j, w_k^j, $j=1,2$, respectivement, donnés par :

$$u_k^j = \left[1 - \alpha_j(\gamma_r + \alpha_j(1-\gamma_r))^{k-1}\right]/(1-\alpha_j),$$

$$v_k^j = \left[1 - (\gamma_r + \alpha_j(1\gamma_r))^{k-1}\right]/(1-\alpha_j),$$

$$w_k^j = \lambda q \frac{\gamma_{r-1}}{1-\gamma_r}\left[(\gamma_r+ \alpha_j(1\gamma_r))^{k-1} + (k-1)(1-\alpha_j)(1-\gamma_r) - 1\right]/(1-\alpha_j)^2$$

$$+ \lambda Q\left[\alpha\ \gamma_r+\alpha_j(1-\gamma_r)\right]^{k-1}+\left[\gamma_r+k(1-\gamma_r)\right](1-\alpha)^{-1}\Big]/(1-\alpha_j)^2$$

Ces quantités sont obtenues simplement à partir de l'équation (27) du modèle M/M/1.

L'évaluation de $\bar{m}_1$ passe par le calcul de E_r^1 et E_r^2, temps d'attente moyen pour un programme de classe C_1 et de classe C_2, respectivement. En utilisant le théorème de LITTLE, nous avons alors :

$$(52) \qquad m_1^j = \beta_j\ \lambda\ E_r^j\ ,\quad j = 1,2,$$

avec :

$$(53) \qquad E_r^j = \sum_{k=1}^{\infty} e_r(k)\Pi^j(k),$$

où $\Pi^j(k)$, probabilité qu'un programme de classe j s'exécute en k quanta, est donnée par :

$$(54) \qquad \Pi^j(k) = \delta_j^k(1-\delta_j).$$

Nous obtenons alors :

$$(55) \qquad E_r^j = z_o + \left[\tau_1+s,\ \tau_2+s\right]\left[Q\, X_j\, Q^{-1}\, \bar{m}_1 + Q\, Y_j\, Q^{-1}\, \bar{m}_1' + Q\, Z_j\, Q^{-1}\, \bar{\beta} \right],$$

où X_j, Y_j, Z_j sont des matrices diagonales qui ont pour éléments x_j^ℓ, y_j^ℓ, z_j^ℓ, $\ell = 1,2$, donnés par :

$$x_j^\ell = \frac{1 - \delta_j \gamma_r}{1 - \delta_j\left[\gamma_r + \alpha_\ell(1-\gamma_r)\right]},$$

$$y_j^\ell = \frac{\delta_j(1-\gamma_r)}{1 - \delta_j\left[\gamma_r + \alpha_\ell(1-\gamma_r)\right]},$$

$$z_j^\ell = \frac{\lambda}{1 - \delta_j\left[\gamma_r + \alpha_\ell(1-\gamma_r)\right]} \quad \frac{1-\gamma_r}{1-\delta_j}\,\delta_j\left[q\,\delta_j\,\gamma_{r-1} + Q(1-\delta_j\gamma_r)\right].$$

Le temps d'attente moyen pour un programme quelconque E_r s'écrit simplement :

$$(56) \qquad E_r = \sum_{i=1}^{2} \beta_j\, E_r^j$$

Les équations (52) et (55) conduisent à un système de deux équations à deux inconnues qui a pour solution m_1^1 et m_1^2, ce qui achève la démonstration.

BIBLIOGRAPHIE

[1] I. ADIRI, B. AVI-ITZHAK — "A time sharing queue with a finite number of customers" J.ACM 16, 2, april 1969.

[2] H.A. ANDERSON, R. SARGENT — "The statistical evaluation of the scheduler of an experimental interactive system" Statistical Computer Performance Evaluation W. Freiberger (ed.), Academic Press, New York, 1972.

[3] J.W. BOYSE — "Execution characteristics of programs in a page-on-demand system" C.ACM 17, 4, april 1974.

[4] B. COCHI — "Several stochastic models of computer systems" Thèse de Ph.D., Stanford University, aug. 1973.

[5] E.G. COFFMAN — "Analysis of two time-sharing algorithms designed for limited swapping" J.ACM 15, 3, july 1968.

[6] E.G. COFFMAN (ed.) — "Computer and job-shop scheduling theory" John Wiley and sons, 1974.

[7] E.G. COFFMAN, P.J. DENNING — "Operating systems theory" Prentice-Hall, Englewoods Cliffs, N.J. (1973).

[8] E.G. COFFMAN, B. KRISHNAMOORTHI — "Preliminary analysis of time shared computer operation" SP-1719, System Dev. Cop., Santa Monica, Calif., aug. 1964.

[9] E.G. COFFMAN, R.C. WOOD — "Interarrival statistics for time-sharing systems" C.ACM 9, 7, july 1969.

[10] R.W. CONWAY, W.L. MAXWELL, L.W. MILLER — "Theory of scheduling" Addison Wesley, Reading, Mass. 1967.

[11] F.J. CORBATO et al. — "An experimental time-sharing system" Proc. AFIPS 1962 SJCC, 21 (1962).

[12] W. FELLER — "An introduction to probability theory" John Wiley and Sons, New York, 1968.

[13] H.C. HEACOX, P.W. PURDOM — "Analysis of two time-sharing queueing models" J.ACM 19, 1, jan. 1972.

[14] R.L. KASHYAP — "Optimization of stochastic finite state systems" IEEE-AC, Vol. AC-11, n° 4, oct. 1966.

[15] J.M. Mc KINNEY — "A survey of analytical time-sharing models" Computing Survey 1, 2, june 1969.

[16] B. KRISHNAMOORTHI, R.G. WOOD — "Time-shared operations with both interarrival and service times exponential" J.ACM 13, 3, july 1966.

[17] J. LABETOULLE — "Ordonnancement des processus temps réels sur une ressource préemptive" Thèse de 3ème cycle, Université de Paris VI, 1974.

[18] J. LEROUDIER — "Analyse d'un système à partage de ressources" RAIRO B-3, oct. 1973.

[19] J. LEROUDIER, M. PARENT — "Discrete event simulation of computer system performance" Rapport de Recherche n° 177, IRIA-LABORIA.

[20] D. POTIER, E. GELENBE, J. LENFANT — "Adaptive allocation of central processing unit quanta" J.ACM 23, 1, jan. 1976.

[21] P.J. RASCH — "A queueing theory study of round robin scheduling of time-shared computer systems" J.ACM 17, 1, jan. 1970.

[22] M. SAKATA, S. NOGUCHI, J. OIZUMI — "An analysis of the M/G/1 queue under round-robin scheduling" Oper. Res. 19, 2, march-april 1971.

[23] A.L. SCHERR — "An analysis of time-shared computer systems" M.I.T. Press, Cambridge, Mass. (1967).

[24] J.I. SCHWARTZ, E.G. COFFMAN, C. WEISSMAN — "A general purpose time-sharing system" Proc. AFIPS 1964 SJCC, 25 (1964).

[25] R.J. WYSZEWIANSKI, R.L. DISNEY — "Feedback queues in the modelling of computer systems : a survey" Technical Report 74-1, Dept. of Industrial and Operation Engineering, The University of Michigan.

Chapitre III

PARTAGE D'UN ENSEMBLE DE TAMPONS A UN NOEUD DE COMMUTATION

CHAPITRE III

Résumé

Dans le chapitre III, nous étudions différents algorithmes de partage de mémoires tampons au noeud d'un réseau à commutation de paquets. Le problème est d'allouer un nombre limité de tampons aux paquets se dirigeant vers les différentes lignes de sortie de façon à minimiser le taux de rejet des paquets par suite d'occupation de toutes les mémoires tampons. Le système étudié est analysé à l'aide d'un modèle à file d'attente simple et trois politiques d'allocation sont comparées. Les deux premières politiques sont des politiques sous-optimales où l'allocation d'un tampon a un paquet est décidée uniquement en fonction du nombre de paquets de même destination que le paquet concerné déjà en attente. La troisième politique est une politique optimale où l'allocation d'un tampon est décidée en fonction de l'ensemble des nombres de paquets en attente. Les performances des deux premières politiques sont calculées explicitement dans le cas de deux lignes de sortie, et un algorithme de programmation dynamique est utilisé pour calculer les politiques optimales. La comparaison des différentes politiques montre que les performances des politiques sous-optimales sont proches des performances des politiques optimales, et des expériences de simulation permettent de vérifier la robustesse des politiques obtenues par rapport aux hypothèses du modèle. A partir de ces résultats de performances, nous montrons comment la synthèse d'un mécanisme de contrôle peut être réalisée dans le cas de deux lignes de sortie.

1. - INTRODUCTION

Les réseaux de transmission de données à commutation de paquets comportent un certain nombre de ressources utilisées pour permettre à un abonné quelconque de demander une connection avec un site choisi et de disposer ainsi des services disponibles sur ce site. La mise en oeuvre de ces ressources est assurée de façon transparente à l'abonné qui ignore comment la connection a été physiquement établie. Le réseau ignore de même le comportement des abonnés et la charge qu'il doit supporter à un instant donné. Comme dans les systèmes multiprogrammés à mémoire virtuelle (cf. chap. IV), des procédures de contrôle de l'allocation des ressources du réseau aux abonnés sont indispensables afin d'assurer un service aussi régulier que possible et prévenir une saturation du réseau. Les résultats expérimentaux disponibles sur les performances des réseaux à commutation de paquets indiquent bien leur sensibilité au mode de gestion utilisé.

On peut très schématiquement partager les ressources d'un réseau en deux parties : d'une part les lignes de transmissions, quelque soit par ailleurs le mode de transmission utilisé, ligne classique ou canal radio ; d'autre part les ressources situées aux noeud de commutation destinées à assuré le stockage des paquets avant leur transmission. Cette classification rapide néglige en particulier les ressources de traitement présentes à chaque noeud, mais, pour les problèmes de contrôle et de partage qui nous intéressent, ces ressources ne sont, en général, pas critiques.

Le problème du partage des lignes de transmission entre les usagers du réseau a été étudié de façon très large : c'est le problème classique du routage, et de la détermination d'algorithmes de routage adaptatif visant à assurer une utilisation des lignes aussi équilibrée que possible en reportant la charge de lignes proches de la saturation sur des lignes moins utilisées [3, 7]. Dans le cas de transmission par canal radio intervient également, dans les liaisons entre un noeud du

réseau et les centres qui lui sont reliés par un même canal, le problème du contrôle de l'accès à ce canal, puisque la transmission d'un message émis par un centre n'est assurée que si aucun autre centre n'émet pendant le même temps. L'étude des conditions de fonctionnement stable d'un tel canal et des politiques de contrôle permettant d'optimiser son utilisation a été présentée dans [4, 5].

Le partage des ressources de mémorisation entre les paquets aux différents noeuds d'un réseau a également donné lieu a un nombre important d'études. Le problème consiste à déterminer l'allocation des blocs d'une mémoire tampon aux paquets arrivant à un noeud pour être transmis sur des lignes de sortie. Le problème le plus complètement traité a été celui du dimensionnement du tampon pour déterminer, suivant le taux d'arrivée des messages et la vitesse des lignes de sortie, la taille du tampon nécessaire pour ne pas dépasser un taux de rejet donné. Citons Chu [1] où l'analyse est conduite à l'aide d'un modèle à file d'attente en supposant un processus d'arrivée poissonnien et plusieurs lignes de sortie synchrones identiques à temps de transfert constant ; Philokypron [12] qui analyse un système de tampon avec des processus d'entrée et de sortie hétérogène, c'est-à-dire dont le taux dépend du temps ; ainsi que [2, 6, 14, 15].

Dans le contexte de réseau à commutation de paquets, les études sur le partage d'un ensemble de tampons par les différents flux de paquets sont moins nombreuses et, dans la pratique, comme c'est le cas pour la machine CIGALE [13], la règle implémentée est le plus souvent la règle premier-arrivé-premier-servi. Toutefois, cette politique a l'inconvénient de ne pas tenir compte des différentes destinations des paquets et peut entrainer un taux de rejet important des paquets ainsi qu'une mauvaise utilisation des lignes de sorties. C'est ce qui ressort des résultats de la simulation du réseau CIGALE menée par Irland [9] qui note que des paquets sont rejetés par un noeud saturé alors qu'ils étaient destinés à être transmis vers les lignes les moins

La représentation du noeud de commutation donnée sur la figure 1 peut être simplifiée comme sur la figure 2 en remarquant que le temps de service du commutateur est beaucoup plus petit que le temps de service des lignes de sortie, ou que le temps moyen entre deux arrivées consécutives de paquets aux noeuds et, qu'en conséquence la file A est pratiquement vide. Ce fait a été vérifié expérimentalement par simulation [9].

Soit n_i, i=1,...p, le nombre de paquets dans la file S_i à un instant donné. Les nombres n_i caractérisent l'état e du système de tampons représenté par le vecteur

$$e = (n_1, n_2, \ldots, n_p),$$

et nous notons

$$E = \{e \mid n_i \geq 0, \sum_{i=1}^{p} n_i \leq N\},$$

l'ensemble des états du système.

Une politique d'allocation consiste à décider, à l'arrivée de chaque nouveau paquet, l'allocation d'un tampon à ce paquet ou, au contraire, le rejet de ce paquet, même si les tampons ne sont pas tous occupés. Cette décision est prise en examinant l'état e de l'ensemble des tampons à l'instant d'arrivée. Une politique d'allocation est caractérisée par les fonctions de décision $\mathcal{O}_i$, i=1,...p

$$\mathcal{O}_i : E \to (0,1), \quad i=1,\ldots p,$$

où

- $\mathcal{O}_i(e) = 1$ signifie que tout paquet arrivant dans la file S_i est rejeté si l'état du système est e ;

- $\mathcal{O}_i(e) = 0$ signifie que tout paquet arrivant dans la file S_i est accepté si l'état du système est e.

En raison de la limitation du nombre de tampons disponibles nous avons les contraintes suivantes sur les fonctions de décision $\mathcal{O}_i$

$$\mathcal{O}_i(e) = 1 \ , \ i=1,\ldots p, \text{ pour tout e tel que } \sum_{i=1}^{p} n_i = N \ ,$$

Dans la suite de ce chapître, nous examinerons trois politiques d'allocation de tampon. Les deux premières politiques constituent des politiques sous-optimales en ce sens que la décision concernant un paquet i est prise suivant uniquement l'état n_i de la file S_i, et non à partir de la connaissance totale $(n_1,\ldots,n_i,\ldots,n_p)$ de l'état du système. L'intérêt de telles politiques est évidemment leur simplicité de mise en oeuvre puisque les $\mathcal{O}_i$ ne dépendent que d'une variable. La troisième politique est une politique optimale, et la comparaison de ses performances avec celles des politiques sous-optimales permettra d'apprécier la valeur de ces dernières.

Pour toutes ces politiques, le critère considéré est la minimisation de la probabilité de rejet de paquets par le noeud de commutation. Les différentes politiques sont définies de la façon suivante :

1 - Politique sous-optimale à partage égal limité

Cette politique appartient à la classe des politiques de gestion de tampons à partage égal limité analysées par Irland [10]. Ces politiques dépendent d'un paramètre unique m, et se définissent comme ci-dessous :

$$\left.\begin{array}{ll} \mathcal{O}_i(e) = 1 & \text{si } e = (n_1,\ldots,n_i,..,n_p) \text{ tel que } n_i \geq m \\ \mathcal{O}_i(e) = 0 & \text{si } e = (n_1,\ldots,n_i,\ldots,n_p) \text{ tel que } n_i < m \end{array}\right\} \ i=1,\ldots,p$$

où $mp \geq N$ de façon à assurer que tous les tampons sont utilisés.

Ces politiques consistent donc à imposer une limite sur le nombre de paquets présents dans chaque file S_i. Les performances de ces politiques dépendent de m, et on note m* le paramètre qui assurent la performance optimale : m* définit la politique sous-optimale à partage égal limité.

2 - Politique sous-optimale à partage inégal limité

Les politiques à partage inégal limité sont caractérisées par l'ensemble de paramètres $\bar{m} = (m_1,\ldots,m_i,\ldots,m_p)$, et constituent une généralisation des politiques précédentes au cas où les limites imposées sur la longueur de chaque file de sortie ne sont pas identiques. Nous avons donc, avec comme précédemment l'hypothèse $\sum_i m_i \geq N$,

$$\left.\begin{array}{ll} \mathcal{O}_i(e) = 1 & \text{si } e = (n_1,\ldots,n_i,\ldots,n_p) \text{ tel que } n_i \geq m_i \\ \mathcal{O}_i(e) = 0 & \text{si } e = (n_1,\ldots,n_i,\ldots,n_p) \text{ tel que } n_i < m_i \end{array}\right\} i=1,\ldots p$$

Le vecteur $\bar{m}^* = (m_1^*,\ldots,m_i^*,\ldots,m_p^*)$ qui assure la meilleure performance définit la politique sous-optimale à partage inégal limité.

3 - Politique optimale

La politique optimale $\mathcal{O}_i^*(e)$, i=1,...p est calculée, comme nous le verrons, à l'aide d'un algorithme de contrôle optimal et pour les hypothèses d'analyse retenues nous montrerons son existence et son unicité.

II. - ANALYSE DES POLITIQUES A PARTAGE EGAL LIMITE

Le modèle simplifié est présenté sur la figure 2.

Nous faisons les hypothèses suivantes en vue d'analyser son fonctionnement :

- les arrivées de paquets commutés sur la ligne de sortie i, i=1,...p se font suivant un processus de Poisson de taux λ_i. Remarquons que cette hypothèse est raisonnable, puisque les paquets commutés sur une ligne donnée proviennent des différentes lignes d'entrée qui constituent autant de sources indépendantes. Dans la suite, afin de simplifier l'exposé nous désignerons par paquet i un paquet commuté sur la ligne de sortie i, i=1,...p.

- les temps de transmission sur la ligne i, i=1,...p des paquets sont indépendants et identiquement distribués suivant une loi exponentielle de moyenne $1/\mu_i$. Cette hypothèse n'est pas complètement réaliste puisque, même si on admet que la longueur des paquets est distribuée suivant une loi exponentielle, le temps de transfert est non-exponentiel du fait des opérations d'initialisation et de fin de transfert qui introduisent des délais fixes. Les conséquences de cette hypothèse sur la valeur des résultats seront analysées par la suite en simulation.

A un instant donné l'état du noeud est caractérisé par les nombres de paquets n_i dans chaque file de sortie i, i=1,...,p, avec les conditions suivantes, liées au nombre fini de tampons et à la politique de partage limité :

$$(1) \qquad \sum_{i=1}^{p} n_i \leq N,$$

$$(2) \qquad 0 \leq n_i \leq m, \quad i=1,...p.$$

Une politique de partage illimité revient à faire m=N. Dans les paragraphes suivants nous analysons le modèle afin de calculer la valeur de m qui minimise le taux de rejet des paquets. Les valeurs de m considérées seront choisies évidemment telles que

$$(3) \qquad m \geq \frac{N}{p} ,$$

afin d'assurer une utilisation maximum des tampons.

Analyse du modèle et calcul de la probabilité de rejet

Soit $(n_1, n_2,\ldots,n_p)$ l'état du système où n_i est défini comme précédemment et $P(n_1, n_2,\ldots,n_p)$ la probabilité stationnaire que le système se trouve dans l'état $(n_1, n_2,\ldots,n_p)$. En raison des nombreuses conditions aux limites imposées par les équations (1) et (2) l'écriture des équations d'équilibre global est difficile, aussi la méthode des équations d'équilibre local [16] est-elle utilisée. En convenant que $P(n_1, n_2,\ldots,n_p) = 0$ lorsque $(n_1, n_2,\ldots,n_p)$ ne satisfait pas aux conditions (1) et (2) ces équations s'écrivent

$$\begin{cases} \mu_1 \, P(n_1+1,\ldots,n_i,\ldots,n_p) = \lambda_1 \, P(n_1,\ldots,n_i,\ldots,n_p) \\ \mu_i \, P(n_1,\ldots,n_i+1,\ldots,n_p) = \lambda_i \, P(n_1,\ldots,n_i,\ldots,n_p) \\ \mu_p \, P(n_1,\ldots,n_i,\ldots,n_p+1) = \lambda_p \, P(n_1,\ldots,n_i,\ldots,n_p) \end{cases}$$

Posons $\rho_i = \lambda_i/\mu_i$, i=1,...p, où ρ_i désigne la charge de la ligne i. La solution du système (4) s'écrit alors

$$P(n_1,\ldots,n_i,\ldots,n_p) = \begin{cases} 1/C \prod_{i=1}^{p} \rho_i^{n_i} & \text{si (1) et (2) sont satisfaites} \\ 0 & \text{sinon} \end{cases}$$

chargées, et qui suggère l'introduction de politiques d'allocation qui évitent la monopolisation de l'ensemble des tampons par les paquets se dirigeant vers une destination donnée.

Nous présentons dans ce chapitre l'étude de politiques d'allocation d'un ensemble de tampons conduite dans le cadre du réseau CIGALE. Trois politiques de partage dont analysées à l'aide d'un modèle simplifié du noeud de commutation. L'observation des performances de ces politiques permet de dégager les principes d'une politique simple qui réalise un contrôle adaptatif du partage des tampons et le principe d'un contrôleur adaptatif est proposé.

Description du système et principes des politiques d'allocation

Un noeud de commutation peut être simplement représenté comme sur la figure 1. Les paquets arrivant vers le noeud sont placés dans la file d'arrivée A avant d'être traités par le commutateur C qui leur attribue une file de sortie S_i suivant leur destination i. Chaque noeud possède un ensemble de tampons en nombre fixe N, et un tampon contient exactement un paquet, quelque soit sa longueur. L'ensemble des tampons est partagé entre les files d'attente A et S_i, i=1,...,p, où p désigne le nombre de lignes de sortie. On suppose que l'algorithme de routage est fixe comme c'est le cas dans le réseau CIGALE, c'est-à-dire que le flux d'arrivée se partage de façon toujours identique vers les différentes lignes de sortie.

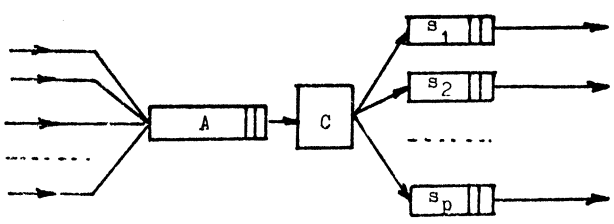


Figure 1

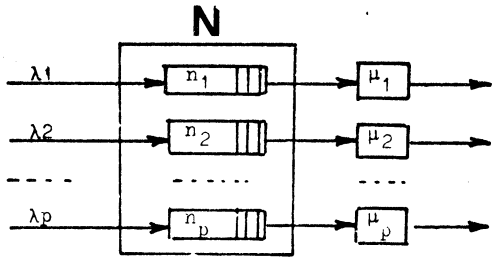


Figure 2

La constante C est une constante de normalisation calculée de façon à ce que la sommation de $P(n_1,\ldots,n_i\ldots,n_p)$ étendue à tous les états du système soit égale à 1. La probabilité de rejet est calculée en notant qu'un paquet est rejeté chaque fois que l'ensemble des tampons est utilisé, c'est à dire $\sum_{i=1} n_i=N$, ou chaque fois que le nombre de paquets présents dans la file de sortie du paquet arrivant atteint m. Le détail des calculs est donné dans l'Annexe A. Les résultats suivants sont obtenus pour p=2 en supposant $\rho_1 \neq 1$, $\rho_2 \neq 1$, $\rho_1 = \rho_2$:

$$R' = \frac{1}{C} \quad \frac{\rho_1^{m+1}\ \rho_2^{N-m} - \rho_1^{N-m} - \rho_2^{m+1}}{1 - \rho_2} + \Pi_1\rho_1^m \frac{1-\rho_2^{N-m}}{1-\rho_2} + \Pi_2\rho_2^m \frac{1-\rho_1^{N-m}}{1-\rho_1} ,$$

avec

$$C = \frac{1-\rho_1^{m+1} - \rho_2^{m+1}}{(1-\rho_1)\ (1-\rho_2)} + \frac{\rho_1^{N+m+1}\ \rho_2^{m+1}}{(1-\rho_1)(\rho_1-\rho_2)} - \frac{\rho_1^{m+1}\rho_2^{N-m+1}}{(1-\rho_2)(\rho_1-\rho_2)} ,$$

et

$$\Pi_i = \lambda_i \Big/ \sum_{j=1}^{2} \lambda_j , \quad i= 1,2.$$

Les expressions de R' et C dans le cas où $\rho_1=1$ ou $\rho_2=1$ ou dans le cas $\rho_1=\rho_2$ sont données en Annexe A.

III. - ANALYSE DES POLITIQUES A PARTAGE INEGAL LIMITE

L'analyse est conduite avec les mêmes hypothèses et suivant le même principe que précédemment. Nous posons

$$s = \sum_{i=1}^{p} m_i \tag{32}$$

avec l'hypothèse, identique à la condition (3)

$$s \geq N. \tag{33}$$

A un instant donné, l'état du noeud est défini par les n_i, i=1,...p, avec les conditions suivantes dues au nombre total limité de tampons N, et à la politique de partage :

$$\sum_{i=1}^{p} n_i \leq N, \tag{34}$$

$$0 \leq n_i \leq m_i, \quad i=1,...p. \tag{35}$$

Les équations du système sont les mêmes qu'au dessus, et la solution s'écrit

$$p(n_1,...,n_i,...,n_p) = \begin{cases} 1/C \prod_{i=1}^{p} \rho_i^{n_i} & \text{si (34) et (35) sont vérifiées} \\ \text{sinon} \end{cases} \tag{36}$$

Le calcul de la constante C et de la probabilité de rejet R" est fait suivant le même principe que dans le cas précédent. La démonstration est donnée en Annexe B et conduit aux résultats suivants pour p=2 avec $\rho_1 \neq 1$, $\rho_2 \neq 1$ et $\rho_1 \neq \rho_2$

$$R'' = \frac{1}{C} \quad \frac{\rho_1^{m_1+1}\rho_2^{N-m_1} - \rho_1^{N-m_1}\rho_2^{m_2+1}}{\rho_1 - \rho_2} + \Pi_1\rho_1^{m_1} \quad \frac{1-\rho_2^{N-m_1}}{1-\rho_2} + \Pi_2\rho_2^{m_2} \quad \frac{1-\rho_1^{N-m_1}}{1-\rho_1}$$

avec

$$C = \frac{1-\rho_1^{m_1+1}-\rho_2^{m_2+1}}{(1-\rho_1)(1-\rho_2)} + \frac{\rho_2^{m_2+1}\rho_1^{N+1-m_2}}{(1-\rho_1)(\rho_1-\rho_2)} - \frac{\rho_1^{m_1+1}\rho_2^{N+1-m_1}}{(1-\rho_2)(\rho_1-\rho_2)}$$

et

$$\Pi_i = \lambda_i / \sum_{i=1}^{2} \lambda_j \quad , \; i = 1, 2.$$

IV. - CALCUL DES POLITIQUES OPTIMALES

IV.1. - FORMULATION DU PROBLEME

Reprenons les hypothèses markoviennes prises dans les deux paragraphes précédents. En raison de la contrainte $\sum_{i=1}^{p} n_i \leq N$, pour toute politique de régulation $\mathcal{O}_i(e)$, il existe toujours un régime de fonctionnement stationnaire du système $P(n_1,\ldots,n_i,\ldots,n_p) = P(e)$ où $P(e)$ représente la probabilité que le système soit dant l'état e à l'équilibre.

Evaluons le critère de fonctionnement du système, c'est-à-dire la probabilité de rejet moyenne pour une politique $\mathcal{O}_i(e)$, i=1,...p donnée. Soit P(e), e $\in$ E les probabilités d'équilibre associées à cette politique. La probabilité qu'un paquet j soit rejeté s'écrit

$$r_j = \frac{\lambda_j}{\sum_{i=1}^{p} \lambda_i} \sum_{e \in E} \mathcal{O}_i(e)\ P(e) \tag{52}$$

et nous en déduisons la probabilité de rejet totale R

$$R = \sum_{j=1}^{p} r_j = \frac{1}{\sum_{i=1}^{p} \lambda_i} \sum_{j=1}^{p} \lambda_j \sum_{e \in E} \mathcal{O}_j(e)\ P(e) \tag{53}$$

Le problème est de trouver la politique $\mathcal{O}_j^*$ qui minimise R. D'après Howard [8] , et en raison des hypothèses markoviennes, on peut montrer simplement que cette politique $\mathcal{O}_j^*$ existe et est unique, et peut être calculée par l'algorithme suivant.

IV.2. - ALGORITHME DE CALCUL

Pour une politique $\mathcal{O}_j$, j=1,...p, correspond à chaque état e du système une décision $\{\mathcal{O}_1(e),\ldots,\mathcal{O}_p(e)\}$ prise parmi l'ensemble des

n_2 \ n_1	0	1	2	3	4	5	6	7	8	9	10
0	1	1	1	1	1	1	1	1	2	2	4
1	1	1	1	1	1	1	1	2	2	4	
2	1	1	1	1	1	1	1	2	4		
3	1	1	1	1	1	1	2	4			
4	1	1	1	1	1	1	4				
5	1	1	1	1	1	4					
6	1	1	1	3	4						
7	1	3	3	4							
8	3	3	4								
9	3	4									
10	4										

Figure 3a : $\lambda_1 = \lambda_2 = 1$.

n_2 \ n_1	0	1	2	3	4	5	6	7	8	9	10
0	1	1	1	1	1	1	2	2	2	2	4
1	1	1	1	1	1	1	2	2	2	4	
2	1	1	1	1	1	1	2	2	4		
3	1	1	1	1	1	1	2	4			
4	1	1	1	1	1	2	4				
5	1	1	1	1	3	4					
6	3	3	3	3	4						
7	3	3	3	4							
8	3	3	4								
9	3	4									
10	4										

Figure 3b : $\lambda_1 = \lambda_2 = 2$.

n_2 \ n_1	0	1	2	3	4	5	6	7	8	9	10
0	1	1	1	1	1	1	1	1	1	2	4
1	1	1	1	1	1	1	1	1	2	4	
2	1	1	1	1	1	1	1	1	4		
3	1	1	1	1	1	1	1	4			
4	1	1	1	1	1	1	3	4			
5	1	1	1	1	3	4					
6	1	1	3	3	4						
7	3	3	3	4							
8	3	3	4								
9	3	4									
10	4										

Figure 4a : $\lambda_1 = 2/3$, $\lambda_2 = 4/3$

n_2 \ n_1	0	1	2	3	4	5	6	7	8	9	10
0	1	1	1	1	1	1	1	2	2	2	4
1	1	1	1	1	1	1	1	2	2	4	
2	1	1	1	1	1	1	1	2	4		
3	1	1	1	1	1	1	1	4			
4	1	1	1	3	3	3	4				
5	3	3	3	3	3	4					
6	3	3	3	3	4						
3	3	3	3	4							
8	3	3	4								
9	3	4									
10	4										

Figure 4b : $\lambda_1 = 4/3$, $\lambda_2 = 8/3$.

décisions possibles pour cet état, c'est-à-dire, l'ensemble des suites binaires à p éléments, soit un ensemble de 2^p décisions possibles. Le calcul de la politique optimale consiste donc à choisir, pour chaque état e du système une des 2^p décisions possibles. Pour simplifier la suite de l'exposé nous numérotons ces différentes décisions de 1 à K en posant $K = 2^p$, et la décision k désignera la décision portant le numéro k.

Soit A la matrice de transition du système d'élément $a_{e,e'}$. La probabilité de transition $a_{e,e'}$ de l'état e à l'état e' dépend de la décision appliquée lorsque le système se trouve dans l'état e, et nous noterons $a^k_{e,e'}$ la probabilité de transition de e vers e' lorsque la décision k est appliquée.

A chaque transition du système est associé un coût dépendant de l'état de départ, de l'état d'arrivée et de la décision prise. Notons q^k_e le coût moyen de transition à partir de l'état e lorsque la décision k est appliquée. Dans le cas p=2, nous avons, pour le système étudié, à partir de l'équation (52)

$$\text{décision } 1 = (0,0) : q^1_e = 0 ,$$

$$\text{décision } 2 = (1,0) : q^2_e = \lambda_1/(\lambda_1+\lambda_2) ,$$

$$\text{décision } 3 = (0,1) : q^3_e = \lambda_2/(\lambda_1+\lambda_2) ,$$

$$\text{décision } 4 = (1,1) : q^4_e = 1.$$

L'algorithme de calcul est un algorithme itératif, chaque itération se décomposant en deux phases :

a) Première phase :

Pour une politique donnée $(\mathcal{O}_1,\ldots,\mathcal{O}_p)$ on calcule le coût de cette politique sous la forme d'un coût par unité de temps g et de coûts

relatifs V_e associés à chaque état e du système. Si le système était dans l'état e à l'instant 0, $V_e + gt$ représente le coût de fonctionnement de 0 à t lorsque la politique $(\mathcal{O}_1,\ldots,\mathcal{O}_p)$ est appliquée.

Le calcul des V_e et de g se fait en résolvant le système linéaire suivant

$$V_e + g = q_e + \sum_{e' \in E} a_{ee'} V_{e'} \quad , \quad e \in E$$

Les V_e étant des coûts relatifs définis à une constante additive près, la solution est obtenue en posant un des V_e égal à 0.

b) Deuxième phase :

C'est la phase d'optimisation qui consiste pour chaque état e du système à calculer la décision k qui minimise

$$q_e^k + \left\{ \sum_{e'} \left[a^k_{e,e'} V_{e'} \right] - V_e \right\} ,$$

où les coûts relatifs V_e sont les coûts calculés dans la première phase. A la fin de la seconde phase, on a calculé une nouvelle politique $(\mathcal{O}'_1,\ldots,\mathcal{O}'_p)$. Le calcul est terminé si $(\mathcal{O}'_1,\ldots,\mathcal{O}'_p) = (\mathcal{O}_1,\ldots,\mathcal{O}_p)$, sinon la phase 1 est reprise avec la politique $(\mathcal{O}'_1,\ldots,\mathcal{O}'_p)$.

On montre que l'algorithme converge en un nombre fini d'itérations, indépendamment du choix de la politique initiale pris pour la première itération. La probabilité de rejet minimale obtenue pour la politique optimale sera notée R^*.

V. - RESULTATS NUMERIQUES ET COMPARAISONS

Les comparaisons des différentes politiques de contrôle sont faites avec les paramètres suivants : p=2, N=10, $\mu_1 = \mu_2 = 1$, et différentes valeurs de ρ_1 et ρ_2. Les résultats sont présentés d'abord pour les politiques

optimales qui permettent de caractériser le plus complètement le comportement de l'ensemble de tampons, et d'introduire les politiques sous-optimales.

Les politiques optimales calculées pour différentes charges du noeud de commutation sont présentées sur les figures 3a, 3b et 4a, 4b. Les figures 3a, 3b correspondent à une charge égale sur les deux lignes, c'est-à-dire $\rho_1 = \rho_2$ et les figures 4a, 4b au cas $\rho_2 = 2\rho_1$. Nous pouvons observer que lorsque les charges sur les lignes ne sont pas égales, la politique optimale tend à minimiser le taux de rejet en refusant préventivement les paquets correspondant à la ligne la plus chargée afin de conserver des tampons disponibles pour les paquets qui sont transférés sur la ligne moins chargée. Nous pouvons également noter que lorsque la charge totale du noeud augmente comme dans les figures 3b et 4b la politique optimale prend une forme proche de celle des politiques à partage limité. Ainsi la politique présentée sur la figure 4b peut être simplement approchée par une politique à partage limité inégal avec $m_1^* = 6$, $m_2^* = 4$.

Les politiques sous-optimales à partage inégal limité sont données sur la figure 5 pour ρ_1 et ρ_2 variant entre 1 et 3. Pour chaque couple (ρ_1, ρ_2) la politique est définie par le couple (m_1^*, m_2^*) où m_i^* représente la limite optimum au nombre de paquets i dans le noeud. Les résultats montrent la sensibilité importante de la politique sous-optimale au couple (ρ_1, ρ_2). Si nous comparons les politiques optimales et sous-optimales, nous pouvons vérifier dans le cas où la charge totale $\rho_1 + \rho_2$ est élevée la similitude de ces deux politiques. La figure 6 présente les politiques sous-optimales à partage égal limité définies, pour chaque couple (ρ_1, ρ_2) par un seul paramètre m^*. La comparaison avec les politiques à partage inégal limité est illustrée en traçant sur la figure 5 les différentes zones de valeur de m^*, et on peut observer que les politiques à partage inégal limité s'écartent d'autant plus des politiques à partage égal limité que ρ_1 est différent de ρ_2.

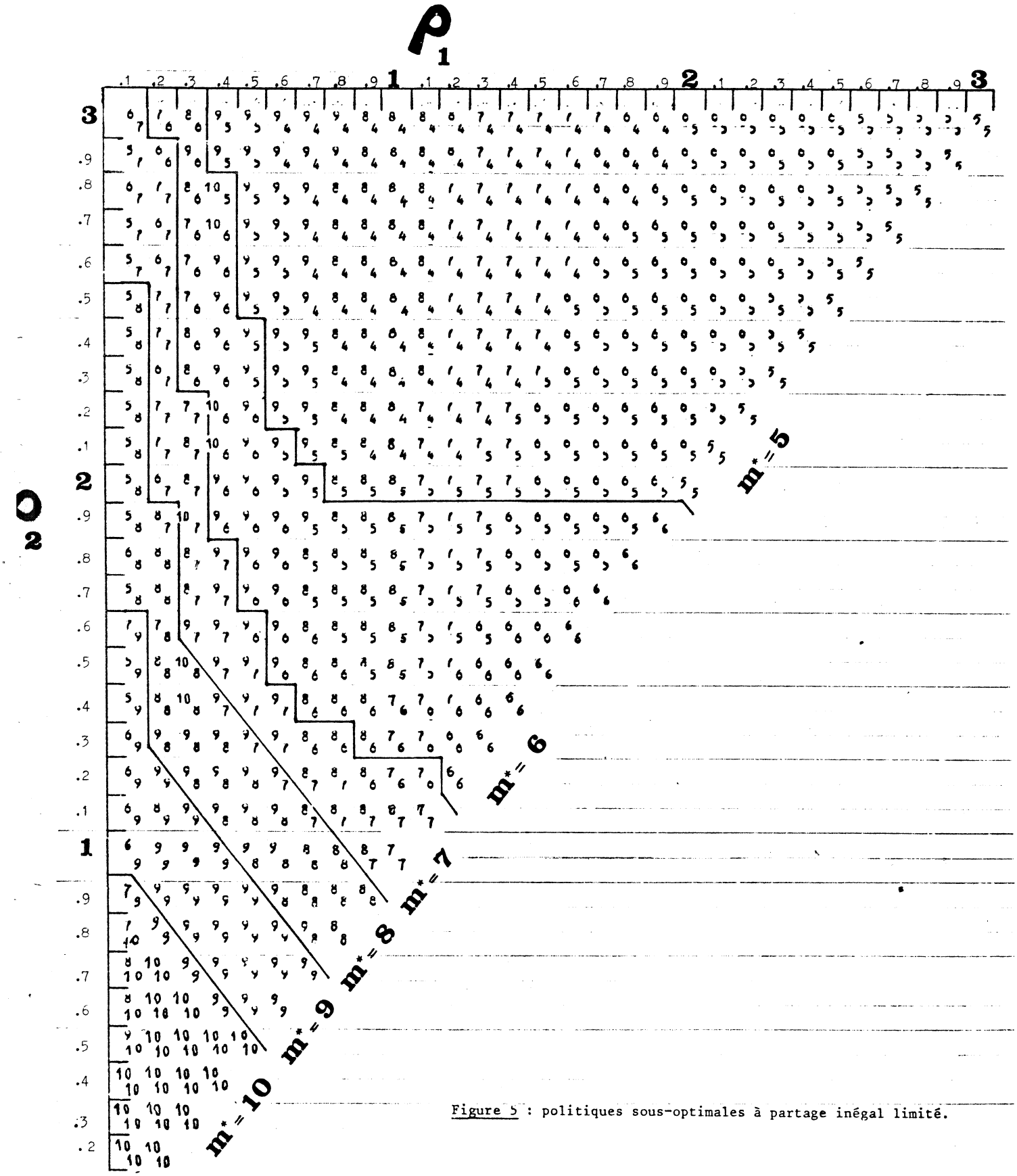


Figure 5 : politiques sous-optimales à partage inégal limité.

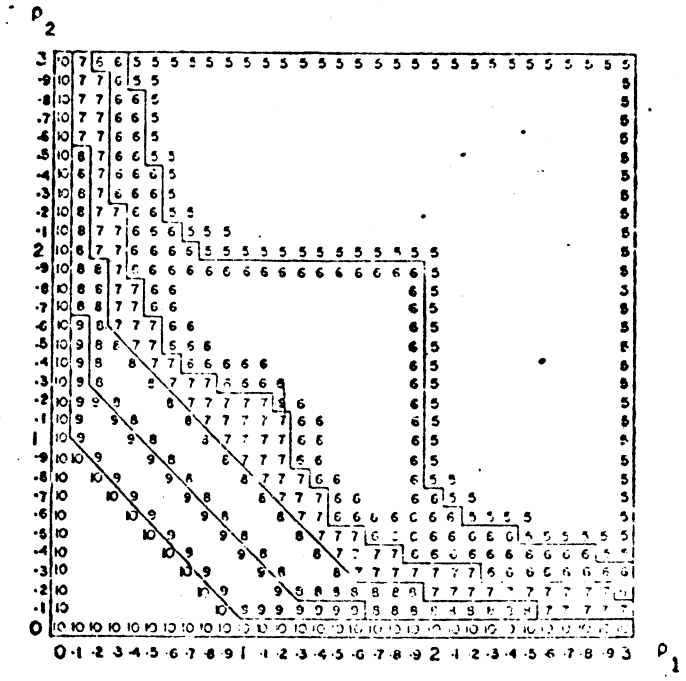


Figure 6 : valeurs de m pour les politiques sous-optimales à partage égal limité.

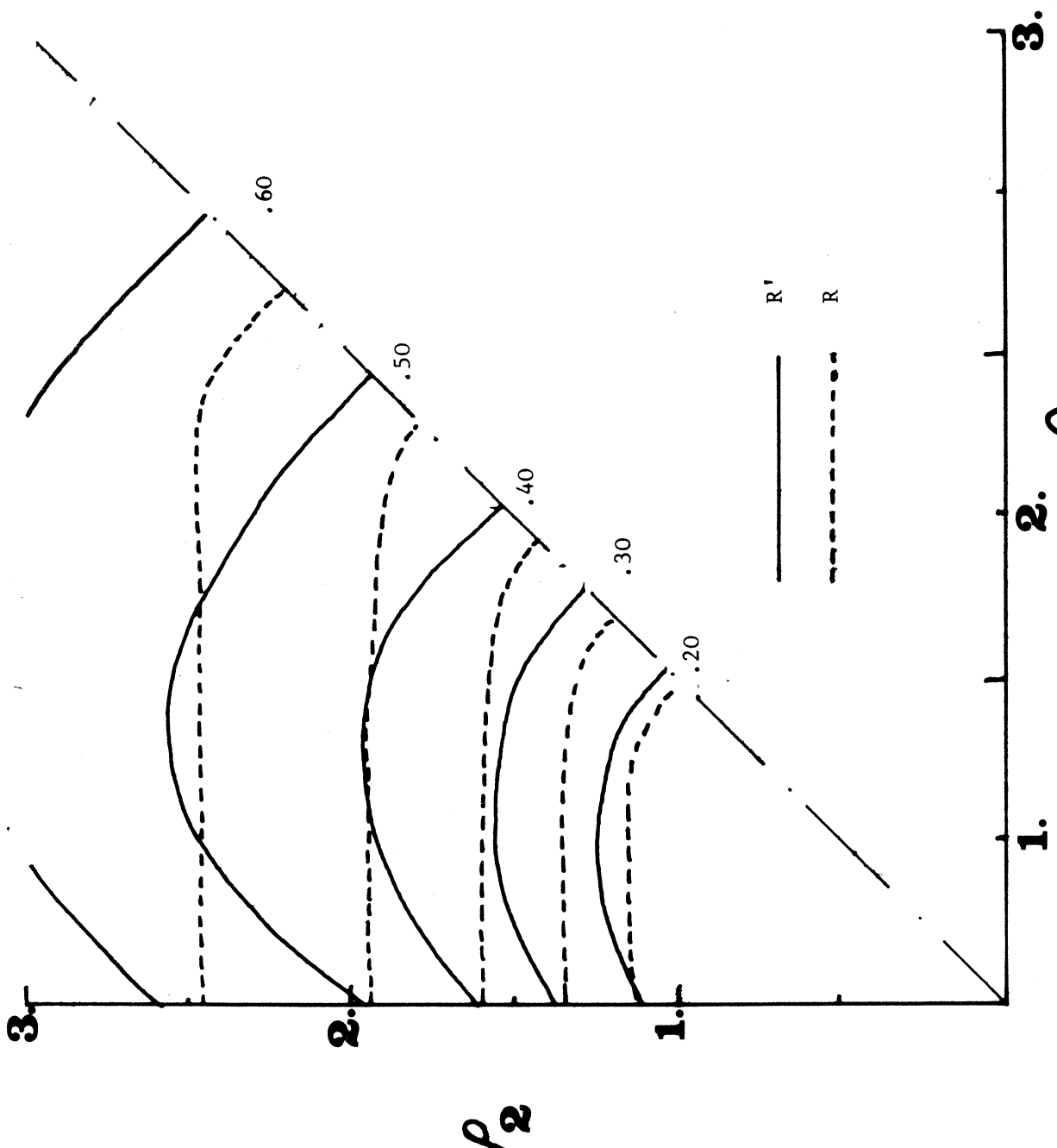


Figure 7 : courbes isovaleurs des probabilités de rejet R' et R avec et sans partage limité.

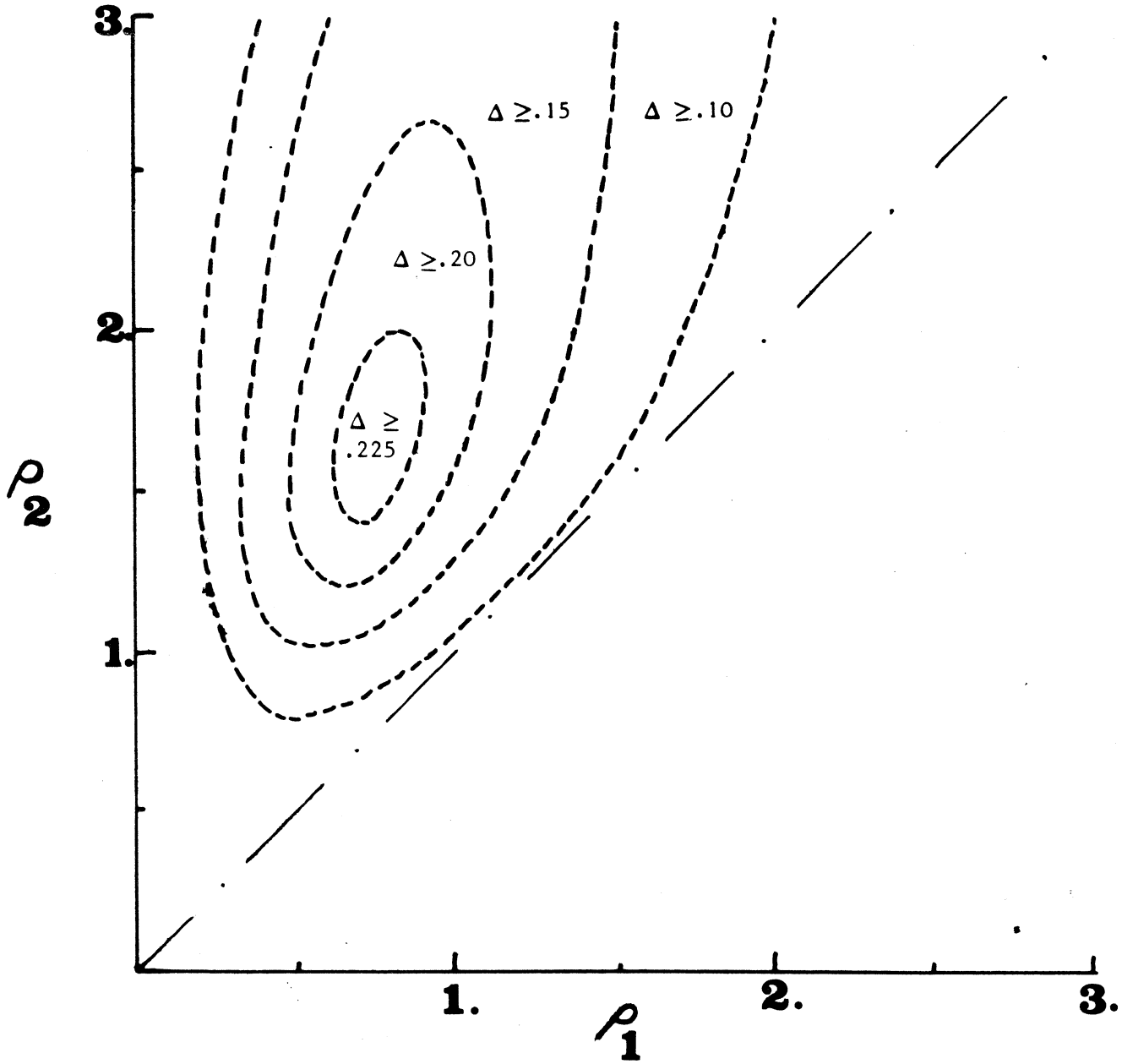


Figure 8 : courbes isovaleurs des améliorations relatives $\Delta = (R-R')/R$ pour les politiques à partage limité.

Les performances des politiques à partage égal limité sont rapportées sur la figure 7 qui donnent les courbes d'isovaleur des probabilités de rejet R' et R avec et sans politiques de contrôle. L'amélioration relative (R-R')/R est représentée de la même façon sur la figure 8 qui met en évidence les zones de fonctionnement où le bénéfice apporté par le contrôle est maximum. La comparaison entre les performances des politiques à partage égal limité et celles des politiques à partage inégal limité montre qu'en dépit de l'allure différente de ces politiques comme noté sur les figures 4 et 5 il n'y a pas de différences sensibles en ce qui concerne leurs performances : les écarts les plus importants sont de 0.14 et sont atteints pour les zones de fonctionnement où le contrôle est le plus éfficace. La même observation peut être faite en ce qui concerne les politiques optimales dont les performances sont comparables à celle des politiques à partage limité, avec un écart relatif qui ne dépasse pas 2.5 %. Le résultat, qui contredit l'intuition puisqu'il semblerait qu'une politiques tenant davantage compte des charges sur les différentes lignes et réalisant une limitation variable suivant les files d'attente doive apporter une amélioration sensiblement meilleure qu'une politique à partage égal limité, montre que l'on peut se contenter d'implémenter les politiques les plus simples. Cette conclusion a été vérifiée sur d'autres exemples avec un nombre de tampons variant de 5 à 50 et, dans tous les cas examinés, l'écart maximum entre les performances de deux politiques reste négligeable.

La validation de ces résultats, qui dépendent bien évidemment des hypothèses de base du modèle utilisé, a été réalisée en supposant que les temps de service des lignes de sortie sont constants afin de s'affranchir de l'hypothèse exponentielle, peu réaliste en ce qui concerne les lignes de transmission. Le modèle a été simulé, et, pour chaque expérience, 100 000 paquets ont été générés.

Les premières expériences portent sur les performances comparées des différentes politiques dans l'hypothèse d'un temps de transmission

ρ_1	1.	1.5	2.	.666	1.	1.333
ρ_2	1.	1.5	2.	1.333	2.	2.666
R'	.088	.341	.500	.175	.361	.503
R^*	.093	.341	.499	.179	.372	.506
$\left\|\frac{R^*-R'}{R^*}\right\|$	.06	-	-	.02	.03	-

Figure 9 : comparaisons des performances des politiques optimales et à partage limité à temps de transmission constant.

ρ_1	1.5	1.5	1.5	2.	2.	2.
ρ_2	.5	2.	3.	.5	1.5	3.
R'	.242	.422	.548	.401	.425	.592
R''	.250	.427	.557	.393	.435	.598
$\left\|\frac{R'-R''}{R''}\right\|$	.03	-	.02	.02	.02	-

Figure 10 : comparaison des performances des politiques à partage limité à temps de transmission constant.

constant. La figure 9 donne les résultats de comparaisons entre les probabilités de rejet R^* et R' des politiques optimales et des politiques à partage égal limité ; la figure 10 les mêmes résultats pour les probabilités R' et R^* des deux politiques à partage limité. Comme dans le cas de l'hypothèse exponentielle, les performances des trois politiques sont très proches, et nous pouvons donc toujours considérer que la politique la plus simple est quasi-optimale.

VI. - CONTROLE ADAPTATIF DU PARTAGE DES TAMPONS

L'implémentation d'une politique à partage égal limité reste toutefois difficile puisqu'elle exige de modifier la valeur de m suivant la charge (ρ_1, ρ_2) du système, et il est donc important d'examiner les performances de politiques simplifiées où m conserve une valeur fixe pour tout couple (ρ_1, ρ_2). Les résultats sont rapportés sur la figure 11 qui donne pour deux valeurs différentes de m, m=5 et m=7, les courbes isovaleurs des améliorations relatives de la probabilité de rejet R. Si l'on compare ces courbes avec les courbes semblables obtenues pour les politiques optimales à paratage limité on remarque les phénomènes suivants :

i) les zones $\Delta \leq .10$, $\Delta \leq .15$, $\Delta \leq .20$ et $\Delta \leq .225$ sont déplacées pour m=5 et m=7, mais restent à l'intérieur des zones semblables obtenues pour m^* ;

ii) la réunion des zones obtenues pour m=5 et m=7 reconstitue sensiblement les zones obtenues pour m^* ;

iii) les points d'intersection des courbes isovaleurs semblables pour m=5 et m=7 définissent une ligne de partage du plan (ρ_1, ρ_2) en deux régions I et II comme figuré sur la figure 11 : dans la région I correspondant aux faibles valeurs de (ρ_1, ρ_2) la politique optimale est approchée par m=7 ; dans la région II par m=5.

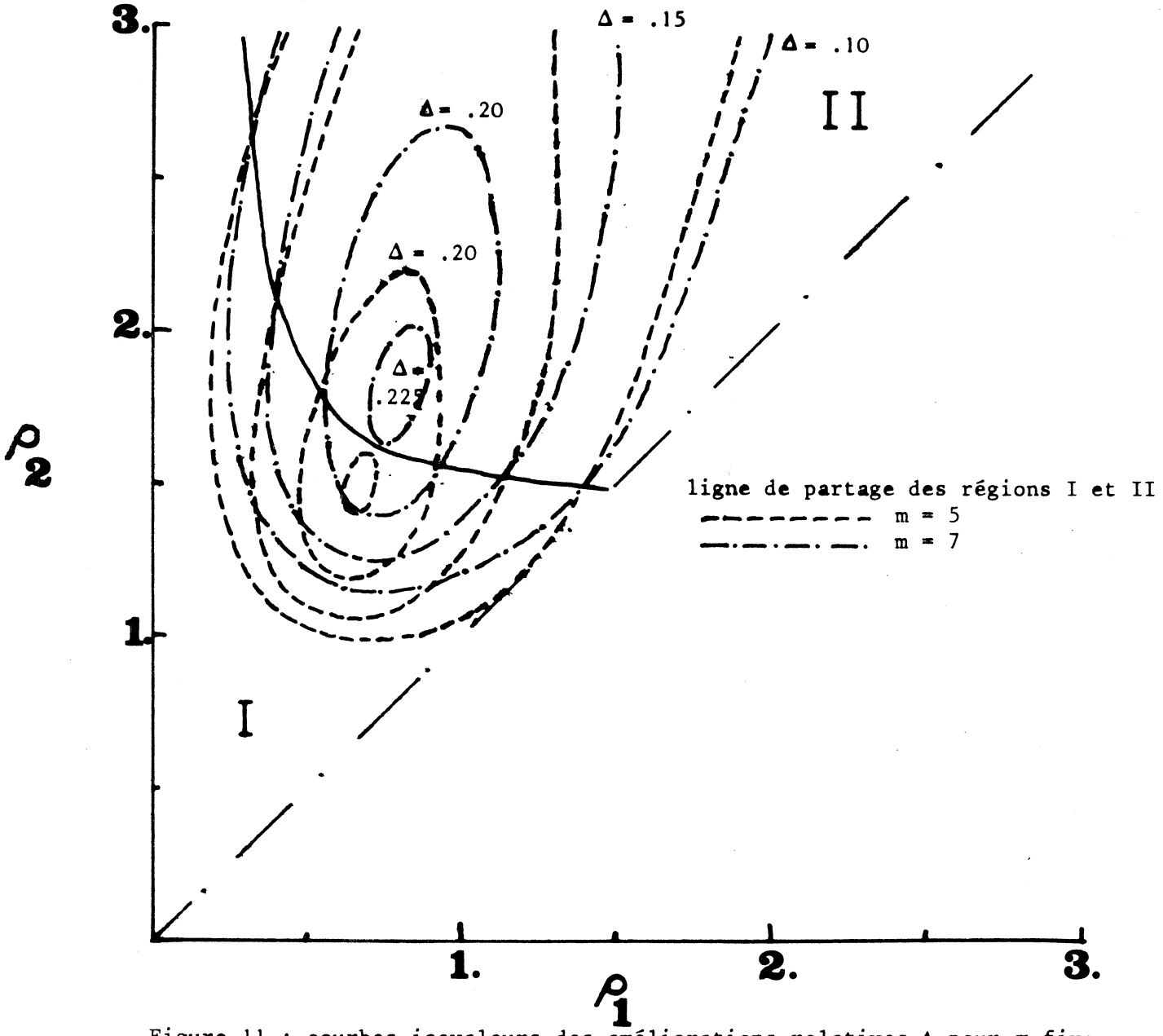


Figure 11 : courbes isovaleurs des améliorations relatives Δ pour m fixe.

Ces remarques permettent de définir un controleur adaptatif du partage des tampons puisque l'implémentation d'une politique optimale de contrôle se réduit donc, dans le cas étudié, à un mécanisme simple où le paramètre de contrôle peut prendre deux valeurs suivant la charge imposée au noeud de commutation. La loi de contrôle est en effet entièrement déterminée par les quantités suivantes :

i) l'équation $f(\rho_1, \rho_2) = 0$ de la ligne de partage entre les deux régions du plan (ρ_1, ρ_2) ;

ii) les paramètres m_I et m_{II} pour chacune des régions.

Nous en déduisons la structure d'un controleur adaptatif du partage des tampons à un noeud de commuation. Le controleur est représenté sur la figure 12 et se compose des éléments suivants :

i) un dispositif EST de mesure et d'estimation des paramètres ρ_1 et ρ_2 ;

ii) un optimisateur OPT calculant m suivant les estimations de ρ_1 et ρ_2 ;

iii) un dispositif limitant la longueur des files d'attente de sortie à m en rejetant à la sortie du commutateur les paquets dont la file de sortie a atteint la longueur limite m.

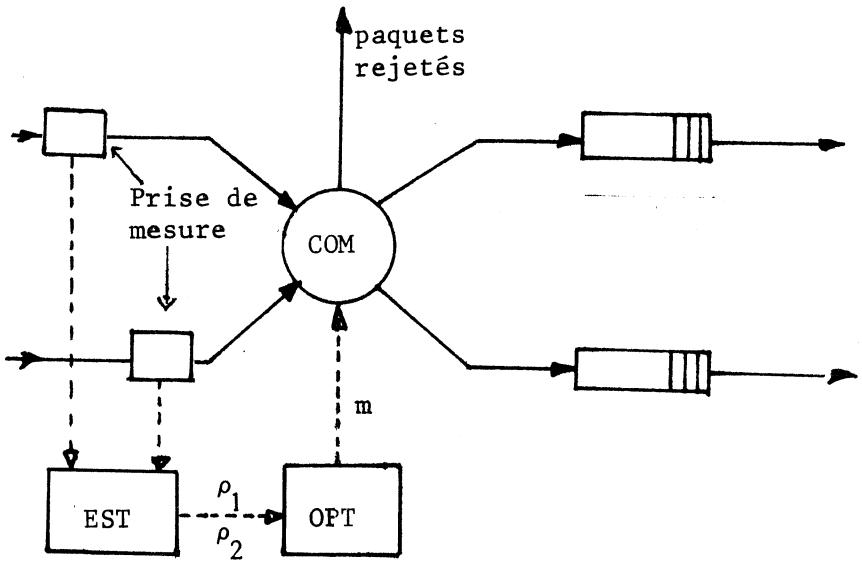


Figure 12 : structure du controleur adaptatif de partage de tampons.

VII. - CONCLUSIONS

Les conclusions les plus importantes de l'étude présentée dans ce chapitre nous paraissent devoir porter sur la démarche adoptée qui a permis, à partir d'une formulation simple du problème, d'aboutir, par différentes phases d'analyse, à la synthèse d'un mécanisme de contrôle adaptatif du partage des tampons au noeud d'un réseau de commutation de paquets.

La première phase a consisté à calculer les politiques optimales d'allocation des tampons. Le résultat fournit ainsi une borne supérieure de l'amélioration que peut apporter une politique d'allocation, et donne une caractérisation des politiques optimales et une indication sur des politiques sous-optimales. Les politiques sous-optimales sont analysées dans une seconde étape et leurs performances sont comparées à celles des politiques optimales. Enfin, la caractérisation des politiques sous-optimales les plus simples suivant la charge du système fournit les principes d'un contrôle adaptatif.

Une telle démarche n'est pas en général possible, en raison notamment de la complexité du système étudié comme nous le verrons dans le chapitre IV. Dans l'exemple de l'allocation de tampons le problème peut se formuler assez simplement pour que les politiques optimales puissent être calculées et les politiques sous-optimales complètement analysées. Pratiquement les résultats obtenues montrent que l'allocation des tampons aux files de sortie peut se faire de façon simple avec un nombre limité de lois de contrôle, puisque, dans l'exemple analysé avec deux lignes de sortie, la synthèse du contrôle optimal est réalisée par deux lois de contrôle.

Nous avons restreint dans l'introduction de ce chapitre les problèmes de contrôle aux partage des lignes et des mémoires tampons entre les paquets. Avec le développement des systèmes répartis apparaît également

le problème de l'allocation et du partage des fichiers entre les différents sites d'un réseau. Les études sur ce sujet sont encore peu nombreuses et celles que nous connaissons aboutissent le plus souvent à un programme mathématique dont la solution est l'implémentation optimale des fichiers pour la structure de coût retenue. L'inconvénient d'une telle approche est qu'elle conduit à une solution statique, valable uniquement sur une période donnée. Une autre approche, semblable à celle que nous avons suivie dans ce chapitre, devrait permettre de caractériser des règles générales d'implantation optimale plutôt qu'une implantation optimale.

Annexe A

Calcul de la constante C

La contrainte de normalisation entraine que C est défini par

$$C = \sum \prod_{i=1}^{p} \rho_i^{n_i} , \tag{6}$$

où la sommation est étendue à tous les états $(n_1,\ldots,n_i,\ldots,n_p)$ satisfaisant les conditions (1) et (2).

Nous pouvons réécrire C de la façon suivante en partitionnant l'ensemble des états en N+1 sous ensembles tels que, dans chacun des sous-ensembles la somme $\sum_{i=1}^{p} n_i$ soit constante. Il vient

$$C = \sum_{k=0}^{N} \sum_{n_1 +\ldots+ n_p = k} \prod_{i=1}^{p} \rho_i^{n_i} . \tag{7}$$

Définissons $\bar{\rho}_i^{(m)}$ comme la suite infinie

$$1,\ \rho_i,\ \rho_i^2,\ldots,\rho_i^n,\ 0,\ 0,\ldots$$

et soit $\otimes$ l'opérateur désigant la convolution de deux suites infinies quelconques. Ainsi, si $\bar{a} = a(0), a(1),\ldots$ et $\bar{b} = b(0), b(1),\ldots$ sont des suites infinies, alors $\bar{a} \otimes \bar{b}$ est aussi une suite infinie d'élément $(\bar{a} \otimes \bar{b})\ (k)$, k=0, 1, 2,... donné par

$$(\bar{a} \otimes \bar{b})\ (k) = \sum_{j=0}^{k} a(j)\ b(k-j).$$

A partir de cette notation nous pouvons exprimer C comme une convolution multiple

$$(8) \qquad C = \sum_{k=0}^{N} (\bar{\rho}_1^{(m)} \otimes \ldots \otimes \bar{\rho}^{(m)}) \ (k).$$

L'utilisation des transformées en z nous fournira la méthode de calcul des convolutions. Notons $Z(\bar{a})$ la transformée en z d'une suite $\bar{a}$ définie par

$$Z(\bar{a}) = \sum_{i=0}^{\infty} a(i)/z^i.$$

Pour le calcul de $Z\left[\bar{\rho}_i^{(m)}\right]$, remarquons que nous pouvons écrire

$$\bar{\rho}_i^{(m)} = \bar{\rho}_i^{(\infty)} \otimes \bar{\ell}_i^{(m)} ,$$

avec

$$\bar{\rho}_i^{(\infty)} = 1, \ \rho_i, \ \rho_i^2, \ldots, \rho_i^m, \ \rho_i^{m+1}, \ldots$$

$$\bar{\ell}_i^{\infty} = 1, \ 0, \ 0, \ldots, 0, \ -\rho_i^{m+1}, \ 0, \ldots$$

d'où

$$Z\left[\bar{\rho}_i^{(m)}\right] = Z\left[\bar{\rho}_i^{(\infty)}\right] . \ Z\left[\bar{\ell}_i^{(m)}\right]$$

$$= \frac{z}{z-\rho_i} \ . \ (1 - \frac{\rho_i^{m+1}}{z^{m+1}})$$

$$(9) \qquad Z\left[\bar{\rho}_i^{(m)}\right] = \frac{1}{z^m} \ \frac{z^{m+1} - \rho_i^{m+1}}{z - \rho_i}$$

Considérons C pour une valeur donnée N comme un élément d'une suite infinie $\bar{C} = C(1), C(2), \ldots, C(N), \ldots$ Alors, à partir de (8), nous avons

$$Z(\bar{C}) = \frac{z}{z-1} Z\left[\bar{\rho}_1^{(m)} \bullet \ldots \bullet \bar{\rho}_p^{(m)}\right] ,$$

$$Z(\bar{C}) = \frac{z}{z-1} Z\left[\bar{\rho}_1^{(m)}\right] \ldots\ldots Z\left[\bar{\rho}_p^{(m)}\right] ,$$

d'où en remplacement $Z\left[\bar{\rho}_i^{(m)}\right]$ par (9), il vient

$$(10) \qquad Z(\bar{C}) = \frac{1}{z^{pm}} \frac{z}{z-1} \prod_{i=1}^{p} \frac{z^{m+1} - \rho_i^{m+1}}{z - \rho_i} .$$

Dans le cas général où p est quelconque, l'inversion de ( ) est impraticable directement. Toutefois, dans le cas p=2, nous pouvons conduire le calcul. La méthode consiste à faire une décomposition partielle en fractions rationnelles de (10) en écrivant $Z(\bar{C})$, lorsque $\rho_i \neq 1$, $\rho_2 \neq 1$, $\rho_1 \neq \rho_2$, sous la forme

$$(11) \qquad Z(\bar{C}) = \frac{1}{z^{2m}} \left[\frac{z}{z-1} p_0(z) + \frac{z}{z-\rho_1} p_1(z) + \frac{z}{z-\rho_2} p_2(z)\right] ,$$

où $p_0(z)$, $p_1(z)$ et $p_2(z)$ sont des polynomes de degré 2m tels que

$$(12) \qquad \frac{p_0(z)}{z-1} + \frac{p_1(z)}{z-\rho_1} + \frac{p_2(z)}{z-\rho_2} = \frac{(z^{m+1} - \rho_1^{m+1})(z^{m+1} - \rho_2^{m+1})}{(z-1)(z-\rho_1)(z-\rho_2)} ,$$

On vérifie que les polynomes $p_0(z)$, $p_1(z)$ et $p_2(z)$ peuvent être choisis de la forme suivante

$$(13) \qquad \begin{cases} p_0(z) = a_{00} z^{2m} + a_{01} z^m + a_{02} , \\ p_1(z) = a_{10} z^{2m} + a_{11} z^m + a_{12} , \\ p_2(z) = a_{20} z^{2m} + a_{21} z^m + a_{22} , \end{cases}$$

Il reste à substituer les expressions de $p_0(z)$, $p_1(z)$ et $p_2(z)$ et à identifier les coefficients des différentes puissances de z, ce qui

conduit aux trois systèmes linéaires ci-dessous

$$(14)\qquad \begin{cases} a_{00} + a_{10} + a_{20} = 1 , \\ (\rho_1+\rho_2)a_{00} + (1+\rho_2)a_{10} + (1+\rho_1)a_{20} = 0 , \\ \rho_1\rho_2 a_{00} + \rho_2 a_{10} + \rho_1 a_{20} = 0 , \end{cases}$$

$$(15)\qquad \begin{cases} a_{01} + a_{11} + a_{21} = 0 , \\ (\rho_1+\rho_2)a_{01} + (1+\rho_2)a_{11} + (1+\rho_1)a_{21} = \rho_1^{m+1} + \rho_2^{m+1} , \\ \rho_1\rho_2 a_{01} + \rho_2 a_{11} + \rho_1 a_{21} = 0 , \end{cases}$$

$$(16)\qquad \begin{cases} a_{02} + a_{12} + a_{22} = 0 , \\ (\rho_1+\rho_2)a_{02} + (1+\rho_2)a_{12} + (1+\rho_1)a_{22} = 0 , \\ \rho_1\rho_2 a_{02} + \rho_2 a_{12} + \rho_1 a_{22} = \rho_1^{m+1}\ \rho_2^{m+1} . \end{cases}$$

Chacun de ces systèmes a une solution unique lorsque $\rho_1 \neq 1$, $\rho_2 \neq 1$, $\rho_1 \neq \rho_2$, donnée par

$$(17)\qquad \begin{cases} a_{00} = \dfrac{1}{(1-\rho_1)(1-\rho_2)} , \ a_{01} = \dfrac{\rho_1^{m+1}+\rho_2^{m+1}}{(1-\rho_1)(1-\rho_2)} , \ a_{02} = \dfrac{\rho_1^{m+1}\ \rho_2^{m+1}}{(1-\rho_1)(1-\rho_2)} , \\ a_{10} = \dfrac{-\rho_1^2}{(1-\rho_1)(\rho_1-\rho_2)}, \ a_{11} = \rho_1 \dfrac{\rho_1^{m+1}+\rho_2^{m+1}}{(1-\rho_1)(\rho_1-\rho_2)}, \ a_{12} = \dfrac{\rho_1^{m+1}\ \rho_2^{m+1}}{(1-\rho_1)(\rho_1-\rho_2)}, \\ a_{20} = \dfrac{\rho_2^2}{(1-\rho_2)(\rho_2-\rho_1)}, \ a_{21} = \rho_2 \dfrac{\rho_1^{m+1}+\rho_2^{m+1}}{(1-\rho_2)(\rho_1-\rho_2)}, \ a_{22} = \dfrac{\rho_1^{m+1}\ \rho_2^{m+1}}{(1-\rho_2)(\rho_1-\rho_2)} , \end{cases}$$

Nous en déduisons l'expression de $Z(\bar{C})$

$$(18)\qquad Z(\bar{C}) = \frac{1}{(1-\rho_1)(1-\rho_2)}\,\frac{z}{z-1} - \frac{\rho_1^2}{(1-\rho_1)(\rho_1-\rho_2)}\cdot\frac{z}{z-\rho_1} + \frac{\rho_2^2}{(1-\rho_2)(\rho_1-\rho_2)}\,\frac{z}{z-\rho_2}$$

$$- \frac{\rho_1^{m+1}+\rho_2^{m+1}}{z^m}\left[\frac{\rho_1}{(1-\rho_1)(1-\rho_2)}\,\frac{z}{z-1} - \frac{\rho_1}{(1-\rho_1)(\rho_1-\rho_2)}\,\frac{z}{z-\rho_1} + \frac{\rho_2}{(1-\rho_2)(\rho_1-\rho_2)}\,\frac{z}{z-\rho_2}\right]$$

$$+ \frac{\rho_1^{m+1}\rho_2^{m+1}}{z^{2m}}\left[\frac{1}{(1-\rho_1)(1-\rho_2)}\,\frac{z}{z-1} - \frac{\rho_1^{N-2m}}{(1-\rho_1)(\rho_1-\rho_2)}\,\frac{z}{z-\rho_1} + \frac{\rho_2^{N-2m}}{(1-\rho_2)(\rho_1-\rho_2)}\,\frac{z}{z-\rho_2}\right].$$

On en déduit directement C(N) suivant

$$(19)\qquad C(N) = \frac{1}{(1-\rho_1)(1-\rho_2)} - \frac{\rho_1^{N+2}}{(1-\rho_1)(\rho_1-\rho_2)} + \frac{\rho_2^{N+2}}{(1-\rho_2)(\rho_1-\rho_2)}$$

$$- (\rho_1^{m+1}+\rho_2^{m+1})\left[\frac{1}{(1-\rho_1)(1-\rho_2)} - \frac{\rho_1^{N-m+1}}{(1-\rho_1)(\rho_1-\rho_2)} + \frac{\rho_2^{N-m+1}}{(1-\rho_2)(\rho_1-\rho_2)}\right]$$

$$+ \rho_1^{m+1}\rho_2^{m+1}\left[\frac{1}{(1-\rho_1)(1-\rho_2)} - \frac{\rho_1^{N-2m}}{(1-\rho_1)(\rho_1-\rho_2)} + \frac{\rho_2^{N-2m}}{(1-\rho_2)(\rho_1-\rho_2)}\right]$$

La dernière ligne de (19) s'annule pour $N < 2m$, ce qui correspond à la condition posée en (3), et l'expression de $C = C(N)$ devient, après simplifications

$$(20)\qquad C = \frac{1 - \rho_1^{m+1} - \rho_2^{m+1}}{(1-\rho_1)\,(1-\rho_2)} + \frac{\rho_1^{N-m+1}\,\rho_2^{m+1}}{(1-\rho_1)(\rho_1-\rho_2)} - \frac{\rho_1^{m+1}\,\rho_2^{N-m+1}}{(1-\rho_2)(\rho_1-\rho_2)}$$

Dans le cas où $\rho_1 = 1$, $\rho_2 = 1$ et $\rho_1 = \rho_2$, l'expression de C devient, respectivement

$$(21)\qquad C = \frac{m+1 - (N-m+1)\,\rho_2^{m+1}}{1-\rho_2} - \frac{\rho_2^{N-m+1}}{(1-\rho_2)^2}\ ,\ \text{pour } \rho_1 = 1\ ,$$

$$(22)\qquad C = \frac{m+1 - (N-m+1)\,\rho_1^{m+1}}{1-\rho_1} - \frac{\rho_1^{N-m+1}}{(1-\rho_1)^2}\ ,\ \text{pour } \rho_2 = 1\ ,$$

(23) $$C = \frac{1}{(1-\rho)^2}\left[1-2\rho^{m+1} - \rho^{N+1}\left[2m-N - \rho(2m-N+1)\right]\right] \text{ , pour } \rho_1 = \rho_2 = \rho.$$

Calcul de la probabilité de rejet

Un paquet est rejeté à son arrivée chaque fois que l'ensemble des tampons est utilisé, c'est-à-dire $\sum_{i=1}^{p} n_i = N$, où chaque fois que le nombre de paquets présents dans la file de sortie du paquet arrivant dépasse m. Calculons la probabilité de rejet, noté R', en posant $\Pi_i = \lambda_i / \sum_{j=1}^{p} \lambda_j$, la probabilité qu'un paquet arrivant soit un paquet i.

Désignons par E et D_i, i=1,...p les sous-ensembles d'états suivant

$$E = \left\{(n_1,\ldots,n_i,\ldots,n_p) \mid \sum_{i=1}^{p} n_i = N\right\} ,$$

$$D_i = \left\{(n_1,\ldots,n_i,\ldots,n_p) \mid n_i=m, \sum_{j\neq i} n_j < N-m\right\} .$$

Nous pouvons alors écrire l'expression de la probabilité de rejet à partir de la définition donnée ci-dessous. Il vient

$$R' = \sum_{E} p(n_1,\ldots,n_i,\ldots,n_p) + \sum_{i=1}^{p} \Pi_i \sum_{D_i} p(n_1,\ldots,n_i,\ldots,n_p) ,$$

soit, en utilisant l'expression de $P(n_1,\ldots,n_i,\ldots,n_p)$, nous avons

(24) $$R' = \frac{1}{C}\left\{\sum_{E}\prod_{i=1}^{p}\rho_i^{n_i} + \sum_{i=1}^{p}\Pi_i \sum_{D_i}\prod_{j=1}^{p}\rho_j^{n_j}\right\} ,$$

où C est donné par (6). En utilisant la même notation que précédemment pour le calcul de C, nous pouvons écrire

(25) $$R' = \frac{1}{C}\left\{\left[\rho_1^{(m)} \otimes \ldots \otimes \rho_p^{(m)}\right] (N) + \sum_{i=1}^{p} \Pi_i \rho_i^m \sum_{k=0}^{N-m-1} \left[\rho_1^{(m)} \otimes \ldots \otimes \rho_{i-1}^{(m)} \otimes \rho_{i+1}^{(m)} \otimes \ldots \otimes \rho_p^{(m)}\right] (k)\right\} .$$

La transformée en z de $\bar{R}' = R'(1), R'(2),\ldots,R'(N),\ldots$ s'obtient simplement comme au-dessus, d'où

$$(26)\qquad Z(\bar{R}') = \frac{1}{z^{pm}}\frac{1}{C}\prod_{i=1}^{p}\frac{z^{m+1}-\rho_i^{m+1}}{z-\rho_i} + \frac{1}{z^{pm}}\frac{1}{z-1}\frac{1}{C}\sum_{i=1}^{p}\Pi_i\,\rho_i^m\prod_{j\neq i}\frac{z^{m+1}-\rho_j^{m+1}}{z-\rho_j}$$

Dans le cas p=2, nous obtenons

$$(27)\qquad Z(\bar{R}') = \frac{1}{C}\left\{\frac{1}{z^{2m}}\frac{z^{m+1}-\rho_1^{m+1}}{z-\rho_1}\,\frac{z^{m+1}-\rho_2^{m+1}}{z-\rho_2} + \frac{1}{z^{2m}}\,\frac{1}{z-1}\left[\Pi_1\,\rho_1\,\frac{z^{m+1}-\rho_2^{m+1}}{z-\rho_2} + \Pi_2\,\rho_2\,\frac{z^{m+1}-\rho_1^{m+1}}{z-\rho_1}\right]\right\}$$

et, après décomposition en fractions rationnelles et inversion de $Z(\bar{p})$, nous avons

$$(28)\quad R' = R'(N) = \frac{1}{C}\left\{\frac{\rho_1^{m+1}\,\rho_2^{N-m}-\rho_1^{N-m}\,\rho_2^{m+1}}{1-\rho_2} + \Pi_1\,\rho_1^m\,\frac{1-\rho_2^{N-m}}{1-\rho_2} + \Pi_2\,\rho_2^m\,\frac{1-\rho_1^{N-m}}{1-\rho_1}\right\}$$

Lorsque $\rho_1 = 1$, $\rho_2 = 1$, ou $\rho_1 = \rho_2 = \rho$, l'expression de R' devient

$$(29)\qquad R' = \frac{1}{C}\left\{\frac{(1-\rho_2^{m+1}) - 2(1-\rho_2^{N-m})}{1-\rho_2} + \Pi_2(N-m)\rho_2^m\right\}\quad\text{, pour } \rho_1 = 1\ ,$$

$$(30)\qquad R' = \frac{1}{C}\left\{\frac{1-\rho_1^{m+1} - \Pi_1(1-\rho_1^{N-m})}{1-\rho_1} + \Pi_1(N-m)\rho_1^m\right\}\quad\text{, pour } \rho_2 = 1\ ,$$

$$(31)\qquad R' = \frac{1}{C}\left\{(2m-N+1)\,\rho^N + \rho^m\,\frac{1-\rho^{N-m}}{1-\rho}\right\}\quad\text{, pour } \rho_1 = \rho_2 = \rho\ .$$

Annexe B

Pour le calcul de C, nous écrivons C suivant

$$C = \sum_{k=0}^{N} \sum_{\substack{n_1+\ldots+n_p=k \\ n_i \leq m_i}} \prod_{i=1}^{p} \rho_i^{n_i} , \tag{37}$$

et nous définissons $\underline{\rho}_i^{(m_i)}$ comme la suite

$$1, \rho_i, \rho_i^2, \ldots \rho_i^{m_i}, 0, 0, \ldots$$

Alors, nous avons

$$C = \sum_{k=0}^{N} \left[\underline{\rho}_1^{(m_1)} \circledast \ldots \underline{\rho}_p^{(m_p)}\right](k). \tag{38}$$

En passant aux transformées en z, et en utilisant les notations précédentes il vient

$$Z(\bar{C}) = \frac{1}{z-1} Z\left[\underline{\rho}_1^{(m_1)}\right] \ldots Z\left[\underline{\rho}_p^{(m_p)}\right] , \tag{39}$$

et en remplaçant $Z\left[\underline{\rho}_i^{(m_i)}\right]$ par son expression ( ) nous obtenons

$$Z(\bar{C}) = \frac{1}{z^s} \frac{z}{z-1} \prod_{i=1}^{p} \frac{z^{m_i+1} - \rho_i^{m_i+1}}{z - \rho_i} \tag{40}$$

Dans le cas p=2, la décomposition partielle en fractions rationnelles de $Z(\bar{C})$ s'écrit

$$Z(\bar{C}) = \frac{1}{z^{m_1+m_2}} \left\{ \frac{z}{z-1} p_0(z) + \frac{z}{z-\rho_1} p_1(z) + \frac{z}{z-\rho_2} p_2(z) \right\} , \tag{41}$$

où $p_0(z)$, $p_1(z)$, $p_2(z)$ vérifient

$$\frac{p_0(z)}{z-1} + \frac{p_1(z)}{z-\rho_1} + \frac{p_2(z)}{z-\rho_2} = \frac{(z^{m_1+1} - \rho_1^{m_1+1})(z^{m_2+1} - \rho_2^{m_2+1})}{(z-1)(z-\rho_1)(z-\rho_2)} \tag{42}$$

On montre simplement que les polynômes $p_0(z)$, $p_1(z)$ et $p_2(z)$ sont de la forme

$$(43)\qquad \begin{cases} p_0(z) = a_{00}\, z^{m_1+m_2} + a'_{01}\, z^{m_1} + a''_{01}\, z^{m_2} + a_{02}\,, \\ p_1(z) = a_{10}\, z^{m_1+m_2} + a'_{11}\, z^{m_1} + a''_{11}\, z^{m_2} + a_{12}\,, \\ p_2(z) = a_{20}\, z^{m_1+m_2} + a'_{21}\, z^{m_2} + a''_{21}\, z^{m_1} + a_{22}\,, \end{cases}$$

On en déduit, à partir de (42), les relations entre les coefficients de $p_0(z)$, $p_1(z)$, $p_2(z)$.

$$(44)\qquad \begin{cases} a_{00} + a_{10} + a_{20} = 1 \\ (\rho_i + \rho_2)\, a_{00} + (1 + \rho_2)\, a_{10} + (1 + \rho_1)\, a_{20} = 0\,, \\ \rho_1 \rho_2\, a_{00} + \rho_2\, a_{00} + \rho_1\, a_{20} = 0 \end{cases}$$

$$(45)\qquad \begin{cases} \rho_1 \rho_2\, a_{02} + \rho_2\, a_{22} + \rho_1\, a_{22} = \rho_1^{m_1+1} + \rho_2^{m_2+1}\,, \\ (\rho_1 + \rho_2)\, a_{02} + (1 + \rho_2)\, a_{12} + (1 + \rho_1)\, a_{22} = 0\,, \\ a_{02} + a_{12} + a_{22} = 0\,, \end{cases}$$

$$(46)\qquad \begin{cases} a'_{01} + a''_{11} + a''_{21} = 0\,, \\ (\rho_1 + \rho_2)\, a'_{01} + (1 + \rho_2)\, a'_{11} + (1 + \rho_1)\, a''_{21} = \rho_2^{m_2+1} \\ \rho_1 \rho_2\, a'_{01} + \rho_2\, a'_{11} + \rho_1\, a''_{21} = 0\,, \end{cases}$$

$$(47)\qquad \begin{cases} a''_{01} + a''_{11} + a'_{21} = 0\,, \\ (\rho_1 + \rho_2)\, a''_{01} + (1 + \rho_2)\, a''_{11} + (1 + \rho_1)\, a'_{21} = \rho_1^{m_1+1}\,, \\ \rho_1 \rho_2\, a''_{01} + \rho_2\, a''_{11} + \rho_1\, a'_{21} = 0 \end{cases}$$

En supposant $\rho_1 \neq 1$, $\rho_2 \neq 1$ et $\rho_1 \neq \rho_2$, les solutions de ces systèmes linéaires sont uniques, et nous avons

$$(48)\quad \begin{cases} a_{00} + \dfrac{1}{(1-\rho_1)(1-\rho_2)} \;, & a_{02} = \dfrac{\rho_1^{m_1+1}\ \rho_2^{m_2+1}}{(1-\rho_1)(1-\rho_2)} \;, \\ a_{10} = \dfrac{-\rho_1^2}{(1-\rho_1)(\rho_1-\rho_2)} \;, & a_{12} = \dfrac{\rho_1^{m_1+1}\ \rho_2^{m_2+1}}{(1-\rho_1)(\rho_1-\rho_2)} \;, \\ a_{20} = \dfrac{\rho_2^2}{(1-\rho_2)(\rho_2-\rho_1)} \;, & a_{22} = \dfrac{\rho_1^{m_1+1}\ \rho_2^{m_2+1}}{(1-\rho_2)(\rho_1-\rho_2)} \;, \end{cases}$$

et

$$(49)\quad \begin{cases} a'_{01} = \dfrac{\rho_2^{m_2+1}}{(1-\rho_2)(1-\rho_1)} \;, & a''_{01} = \dfrac{\rho_1^{m_1+1}}{(1-\rho_2)(1-\rho_1)} \;, \\ a'_{11} = \dfrac{\rho_2^{m_2+1}}{(1-\rho_1)(\rho_1-\rho_2)} \;, & a''_{11} = \dfrac{\rho_1^{m_1+2}}{(1-\rho_1)(\rho_1-\rho_2)} \;, \\ a''_{21} = \dfrac{\rho_2^{m_2+1}}{(1-\rho_2)(\rho_1-\rho_2)} \;, & a'_{21} = \dfrac{\rho_1^{m_1+1}}{(1-\rho_2)(\rho_1-\rho_2)} \;, \end{cases}$$

Il reste à écrire $Z(\bar{C})$ en remplaçant les coefficients de $p_0(z)$; $p_1(z)$ et $p_2(z)$ par leurs expressions données ci-dessus, et, ensuite, à inverser $Z(\bar{C})$ en remarquant que d'après (33), $m_1+m_2 \geq N$. Il vient finalement

$$(50)\quad C = \frac{1 - \rho_1^{m_1+1} - \rho_2^{m_2+1}}{(1-\rho_1)(1-\rho_2)} + \frac{\rho_2^{m_2+1}\ \rho_1^{N+1-m_2}}{(1-\rho_1)(\rho_1-\rho_2)} - \frac{\rho_1^{m_1+1}\ \rho_2^{N+1-m_1}}{(1-\rho_2)(\rho_1-\rho_2)} \cdot$$

Nous obtenons de la même façon l'expression de la probabilité de rejet R"

(51)

$$R'' = \frac{1}{C}\left\{\frac{\rho_1^{m_1+1}\ \rho_2^{N-m_1} - \rho_1^{N-m_1}\ \rho_2^{m_2+1}}{\rho_1 - \rho_2} + \Pi_1\rho_1^{m_1}\ \frac{1-\rho_2^{N-m_1}}{1-\rho_2} + \Pi_2\rho_2^{m_2}\ \frac{1-\rho_1^{N-m_1}}{1-\rho_1}\right\}$$

BIBLIOGRAPHIE

[1] W.W CHU — "Buffer behaviour for poisson arrivals" IEEE Trans. on Comp. Vol. C-19, n° 6, june 1970.

[2] W.W. CHU — "Dynamic buffer management for computer communications"

[3] D.W. DAVIES — "The control of congestion in packet switching networks" Proc. 2nd ACM/IEEE Symposium on problems in the optimization of Data communications systems, 1971.

[4] G. FAYOLLE, E. GELENBE, J. LABETOULLE, D. BASTIN — "The stability problem of broadcast packet switching computer networks" To appear in Acta Informatica.

[5] G. FAYOLLE, E. GELENBE, J. LABETOULLE — "Stability and control of packet switching broadcast channels" Rapport de Recherche n° 116, IRIA-LABORIA, avril 1975.

[6] D.P. GAVER — "Probability models for buffer storage allocation problems" J.ACM Vol. 18, n° 2, april 1967.

[7] M. GERLA, W.W. CHU, H. FRANK — "Computational considerations and routing problems for large computer communication networks" Proc. of the National Telecommunication Conference, Atlanta, nov. 1973.

[8] R.A. HOWARD — "Dynamic probabilistic systems, Vol. II" John Wiley, New York, 1971.

[9] M. IRLAND — "Simulation of CIGALE 1974" Proc. 4th Data Communication Symposium, Quebec, jan. 1975

[10] M. IRLAND — "Queueing analysis of a buffer allocation scheme for a packet switch" IEEE-NTC'75 Conference Proceedings, New Orleans, dec. 1975.

[11] M. IRLAND — "Computational algorithms for a system of exponential servers serving a common finite storage" Research Report, Computer Communications, Networks Group, University of Waterloo, Canada - 1976.

[12] G. PHILOKYPROU, D.G. MARITSAS — "A class of buffer systems with heterogenous input and output processes" Int. J. Systems Sci., Vol. 5, n° 3, 1974.

[13] L. POUZIN — "Presentation and major design aspects of the Cyclades computer network" Proc. 3rd Data Communications Symposium, St Petersburg (Florida), nov. 1973.

[14] M.A. RICH, M. SCHWARTZ — "Buffer sharing in computer communications network nodes"

[15] P.J. SCHWEITZER, S.S. LAM — "Buffer overflow in a store-and-forward network node" IBM Research Report, RC 5759, nov. 1975.

[16] F. BASKETT et al — "Open, closed, and Mixed Networks of queues with Different classes of Customers" J. ACM 22, 2 (April 75)

Chapitre IV

GESTION DE LA MULTIPROGRAMMATION DANS LES SYSTEMES A MEMOIRE VIRTUELLE

CHAPITRE IV

Résumé

Nous présentons dans le quatrième chapitre les principes de contrôle du degré de multiprogrammation dans les systèmes multiprogrammés à mémoire virtuelle. Ces principes sont abordés par l'étude de la dynamique des programmes et de la dynamique du système étudié. La dynamique des programmes est analysée à partir de données expérimentales et de modèles mathématiques et nous caractérisons à l'aide d'un modèle à file d'attente le fonctionnement d'une architecture à mémoire virtuelle du point de vue de ses performances. Nous montrons ensuite que les règles de contrôle du degré de multiprogrammation peuvent se classer en deux catégories suivant que contrôle est réalisé à partir d'informations prises sur le comportement des programmes ou sur les performances du système. Dans le première catégorie entrent la règle de l'ensemble de travail, la règle du "genou", et le règle L=S, toutes trois dues à Denning ; dans la seconde catégorie l'algorithme de contrôle adaptatif fondé sur le critère de la dilatation maximum et le règle des 50 %. Ces règles sont présentées et analysées et les problèmes posés par leur mise en oeuvre sont discutés. Nous montrons que l'implémentation d'une règle de contrôle pose des problèmes d'instrumentation, d'estimation et de détermination d'algorithmes de contrôle. Ces questions sont illustrées à partir des différents principes de contrôle présentés.

1. - INTRODUCTION

Les années soixante sont marquées par l'apparition des premiers ordinateurs à mémoire virtuelle [22] . Le mécanisme de traduction dynamique des adresses permet alors de résoudre les problèmes de gestion et de partage de la mémoire dans les ordinateurs importants de l'époque conçus pour travailler en multiprogrammation. Les programmes peuvent ainsi être déchargés de la mémoire principale et y être rechargés à des emplacements physiques différents, sans que soit nécessairement préservée la contiguité logique de l'information dans la mémoire physique.

En plus d'un avantage propre à la multiprogrammation, ce mécanisme permet d'exécuter des programmes dont l'espace d'adressage à une taille supérieure à celle de la mémoire physique de la machine, sans que pour celà l'utilisateur ait à construire d'arbres de recouvrement (overlay). Cette possibilité est mise en oeuvre en ne chargeant que la partie utile d'un programme à un instant donné (les instructions en cours et les données accédées) et en modifiant cette partie utile à la demande au cours de l'exécution (chargement "à la demande"). C'est alors que le terme de mémoire virtuelle prend son plein sens et que le mécanisme de traduction dynamique des adresses est exploité au mieux de ses possibilités.

Une telle solution se généralisa rapidement à des machines telles que GE 645 et IBM 360/67 sur lesquelles furent implantés des systèmes d'exploitation qui utilisaient les mécanismes de traduction d'adresses : MULTICS[5] MTS [40] , CP67 [36] . Actuellement, ces mécanismes sont disponibles sur les ordinateurs de haut de gamme récents tels que IBM 370 exploité sous les systèmes d'exploitation VM [20] ou VS chez IBM et IRIS 80 exploité sous le système MAS [8] développé à l'Université de Grenoble.

Toutefois, les avantages apportés aux utilisateurs par la technique de la mémoire virtuelle se traduisent par une charge supplémentaire sur certaines ressources du système. Les algorithmes d'allocation dynamique des blocs d'information, ou page, ainsi que les tables nécessaires pour réaliser la traduction entre adresses virtuelles et adresses réelles augmentent le volume du système d'exploitation et réduisent d'autant l'espace mémoire disponible aux utilisateurs. D'autre part, par le jeu du mécanisme d'allocation dynamique, chaque fois qu'une page référencée ne se trouve pas en mémoire principale, le

programme en cours d'exécution est interrompu, son contexte sauvegardé, les algorithmes de pagination sont exécutés et les transferts de la page requise et éventuellement de la page à décharger sont initialisées. Ces opérations entraînent un dépassement ("overhead") du temps d'unité centrale au détriment de l'exécution des programmes utilisateurs. Enfin, la pagination a pour effet d'imposer un échange important d'information entre la mémoire principale et la mémoire secondaire qui supporte la mémoire virtuelle. On conçoit alors que le mécanisme de pagination va faire apparaître, à forte charge, un goulot d'étranglement sur la mémoire secondaire en imposant des temps d'attente à chaque transfert de pages de plus en plus importants. Lorsque le nombre de programmes qui se partagent les ressources du système augmente, le chargement à la demande des pages et la libre compétition des programmes dans la mémoire physique provoquent un phénomène en avalanche connu sous le nom d'écroulement ou thrashing [13].

Nous voyons donc qu'à côté de problèmes d'adressage liés à la gestion des espaces virtuels, le fonctionnement des machines à mémoire virtuelle pose des problèmes de performances qu'il est difficile d'appréhender directement en raison de la complexité même de l'organisation de ces machines. Durant ces dix dernières années, de nombreux travaux ont été publiés dans cette direction pour étudier les performances de différents algorithmes de pagination [18,17,25] et pour évaluer les performances de systèmes à mémoire virtuelle [14,16,17,34]. Dans le même temps s'est développée l'étude des lois de comportement des programmes vis à vis des ressources qu'ils utilisent : mémoire centrale, unité entrée-sortie, unité centrale, et plusieurs modèles de ces comportements ont été proposés [3,7,25,28]. Associé à un modèle d'architecture de système, ce type de modèle permet de caractériser et d'analyser globalement les performances d'une architecture en faisant apparaître les relations entre les phénomènes de comportement de programme et l'organisation de l'architecture, et en permettant une analyse quantitative de ces relations [10,31,34,39].

L'ensemble de ces analyses des performances des systèmes à mémoire virtuelle met en évidence la sensibilité de ces performances et du phénomène d'écroulement au degré de multiprogrammation du système (ou nombre de programmes qui se partagent les ressources) et la nécessité d'opérer un contrôle sur le partage des ressources. Plus précisément, les analyses montrent qu'il

existe pour un type de programme donné un degré de multiprogrammation optimum qui maximise le débit du système et l'utilisation de ses ressources. Toutefois, en raison du caractère hautement variable des caractéristiques des programmes soumis au système, le degré de multiprogrammation optimum ne peut être fixé à priori, et il s'agit de mettre en oeuvre des techniques de contrôle adaptatif qui assurent que le dégré de multiprogrammation reste toujours, quelque soient les conditions de fonctionnement, proche du degré de multiprogrammation optimal.

Comme nous l'avons noté plus haut, les performances globales d'un système multiprogrammé à mémoire virtuelle dépendent d'une part des propriétés de comportement des programmes, d'autre part de la dynamique du système lui-même face à ces propriétés de comportement de programme. C'est l'étude de ces deux phénomènes qui permettra de dégager des lois générales sur les conditions de fonctionnement optimum des systèmes à mémoires virtuelles, et de tirer de ces lois des régles de contrôle du degré de multiprogrammation. Comme nous le verrons par la suite, ces régles de contrôle peuvent être classées deux familles suivant qu'elles sont construites sur des propriétés de comportement des programmes, ou sur des propriétés de comportement du système.

Nous présentons dans ce chapitre l'analyse de plusieurs règles de contrôle du degré de multiprogrammation. Les deux premiers paragraphes sont consacrées aux propriétés de comportement des programmes et à l'étude de la dynamique du système . Les différentes règles sont ensuite définies et analysées et leurs performances sont comparées.

2. - DYNAMIQUE DES PROGRAMMES

Nous présentons dans ce paragraphe des résultats expérimentaux et des modèles caractéristiques du comportement dynamique des programmes dans leur espace d'adressage dans le contexte d'un système à mémoire virtuelle gérée suivant la technique de la page à la demande. Les résultats sont obtenus pour deux modèles : le modèle de l'ensemble de travail et le modèle de durée de vie et les données présentées sont choisies afin de mettre en évidence les propriétés de ces modèles les plus importantes pour la gestion dynamique du dégré de multiprogrammation.

Les données expérimentales ont été obtenues à partir de traces de références de programme créées par interprétation de l'exécution de programmes [25] Cinq programmes différents ont été mesurés et leurs caractéristiques principales sont les suivantes :

Programme A : compilateur FORTRAN (30 k mots) compilant deux modules
Programme B : exécution d'un programme FORTRAN (16k mots)
Programme C : compilateur LP 70 (15,5 k mots) compilant un programme de 500cartes
Programme D : interpréteur LISP (14 k mots)
Programme E : exécution d'un programme COBOL (7,5 k mots)

L'ensemble des résultats dont sont extraites les données expérimentales présentées ici ont été publiés dans [28].

2.1. - Le modèle de l'ensemble de travail

La notion de localité est un concept essentiel pour décrire le comportement des programmes dans leur espace d'adressage. La propriété de localité signifie que durant un intervalle d'exécution donné la plus grande partie des références générées constitue un sous ensemble de l'espace d'adressage, sous ensemble stable qui se modifie lentement au cours de l'exécution. Cette propriété que l'on peut observer expérimentalement à partir de l'étude de traces de références de programmes a été vérifiée dans des conditions tres larges et peut s'expliquer, au moins partiellement par la façon dont sont conçus et écrits les programmes :

- un programme comporte des suites d'instructions exécutées en séquence séparées par des instructions de branchement ;
- les suites d'instructions peuvent être répétées plusieurs fois consécutivement ;
- les données sont organisées en tableaux et leur traitement est séquentiel ;
- au cours des différentes phases du programme une partie seulement des données est référencée.

Pour ces différentes raisons, il se créé des concentrations de références dans certaines régions, ou localités de l'espace d'adressage. Cette notion de localité a été formalisée par Denning [12b] à l'aide du concept de l'ensemble de travail, puis étendue par Batson [3] par le concept d'ensemble actif (activity set) analysé également par Lenfant[26]. Nous nous intéressons dans ce paragraphe au concept d'ensemble de travail à partir duquel un principe de contrôle du degré de multiprogrammation peut être établi et nous présentons des résultats expérimentaux caractérisant la taille de l'ensemble

Nous définissons l'ensemble de travail d'un programme dans le contexte d'un système paginé. Soit P = {1,2,...n }l'ensemble des pages du programme considéré. La suite des pages référencées au cours d'une exécution est représentée par un processus stochastique (R_t), $t \in \mathbb{N}$.

Définition

Soit t et T deux entiers positifs tels que $T \leqslant t$. Pour une réalisation

$r_o, r_1, \ldots, r_t, \ldots$

du processus (R_t), l'ensemble de travail de fenêtre T est, à l'instant t

$$W(t,T) = \{ i \in P \mid \exists\ n \in \mathbb{N}\ ,\ t-T+1 \leqslant n \leqslant t, r_n = i\}$$

et la taille de W (t,T), notée w(t,T) est le nombre de pages contenues dans l'ensemble de travail W(t,T)

Remarquons que pour une taille de page et un découpage en pages données de l'espace d'adressage, la notion d'ensemble de travailest une notion purement intrinsèque au programme, et ne dépend pas des algorithmes d'allocation de mémoire utilisée.

L'ensemble de travail W (t,T) représente, à la t-ième référence, le sous ensemble de l'espace d'adressage contenu dans les T références précédentes. La mesure de l'ensemble de travail d'un programme, et plus précisément de la taille de cet ensemble doit donc permettre de vérifier les propriétés de localité des programmes. Cette vérification peut être faite en considérant les trois mesures suivantes :

- la variation de la taille moyenne de l'ensemble de travail, noté $\bar{\omega}$ (T), avec la taille de la fenêtre ;
- la dispersion de la taille de l'ensemble de travail, c'est à dire le rapport $c=\sqrt{\text{var } \omega(t,T)} \ / \ \bar{\omega}$ (T)
- la fonction d'autocorrélation $\rho(k)$, c'est à dire la corrélation entre ω (i,T) et ω (i+ k,T)

La première mesure donne une indication sur la taille moyenne des localités, la seconde sur la stabilité de cette taille, la troisième sur la possibilité de prédire la taille de l'ensemble de travail d'un programme à un instant donné, connaissant la taille de l'ensemble de travail aux instants précédents.

Les résultats présentés sur les figures1 à 4 ont été obtenus à partir de la trace de références du programme C. La figure 1 met en évidence l'allure classique des courbes de $\bar{\omega}$ (T) en fonction de T. Dans ces expériences la taille de la fenêtre a été limitée à 1000 références, mais les résultats pour des tailles de fenêtres plus grandes montreraient encore plus clairement que la taille moyenne de l'ensemble de travail augmente peu lorsque T croit. Notons également que la taille de l'ensemble de travail mesurée en rapport avec la taille de l'espace d'adressage est d'autant plus faible, pour une taille de fenêtre donnée, que la taille de page est petite. Ceci est lié au phénomène de fragmentation qui entraîne que la quantité d'information inutile comptée dans l'ensemble de travail est d'autant plus grande que la taille de la page est importante.

Le coefficient de variation de la taille de l'ensemble de travail est donné sur la figure 2 en fonction de la taille de la fenêtre T et pour différentes tailles de page. Les résultats montrent que le coefficient de variation est toujours très sensiblement inférieur à 1 avec des valeurs proches de 0.3 pour les tailles de fenêtres utilisées pratiquement [37].La forme des courbes est conforme au modèle théorique [15] avec un maximum, et une branche asymptotique pour les T croissants et une limité vers 0 quant T tend vers 0. Des observations semblables ont été faites par Rodriguez [37] .

Les figures 3 et 4 présentent la fonction d'auto corrélation $\rho(k)$ en fonction de k. La figure 3 montre que $\rho(k)$ est pratiquement indépendant de la taille de page considérée. La figure 4 où $\rho(k)$ est calculée pour une taille de fenêtre réaliste illustre les propriétés de prédiction de l'ensemble de travail puisque le coefficient de corrélation vaut encore 0.6 pour deux mesures de l'ensemble de travail séparés par 30 000 références.

L'ensemble des résultats qui viennent d'être décrits seront commentés plus loin, en comparaison avec les résultats obtenus pour d'autres modèles de comportement, et en fonction de l'utilisation de ces modèles pour opérer une régulation du degré de multiprogrammation.

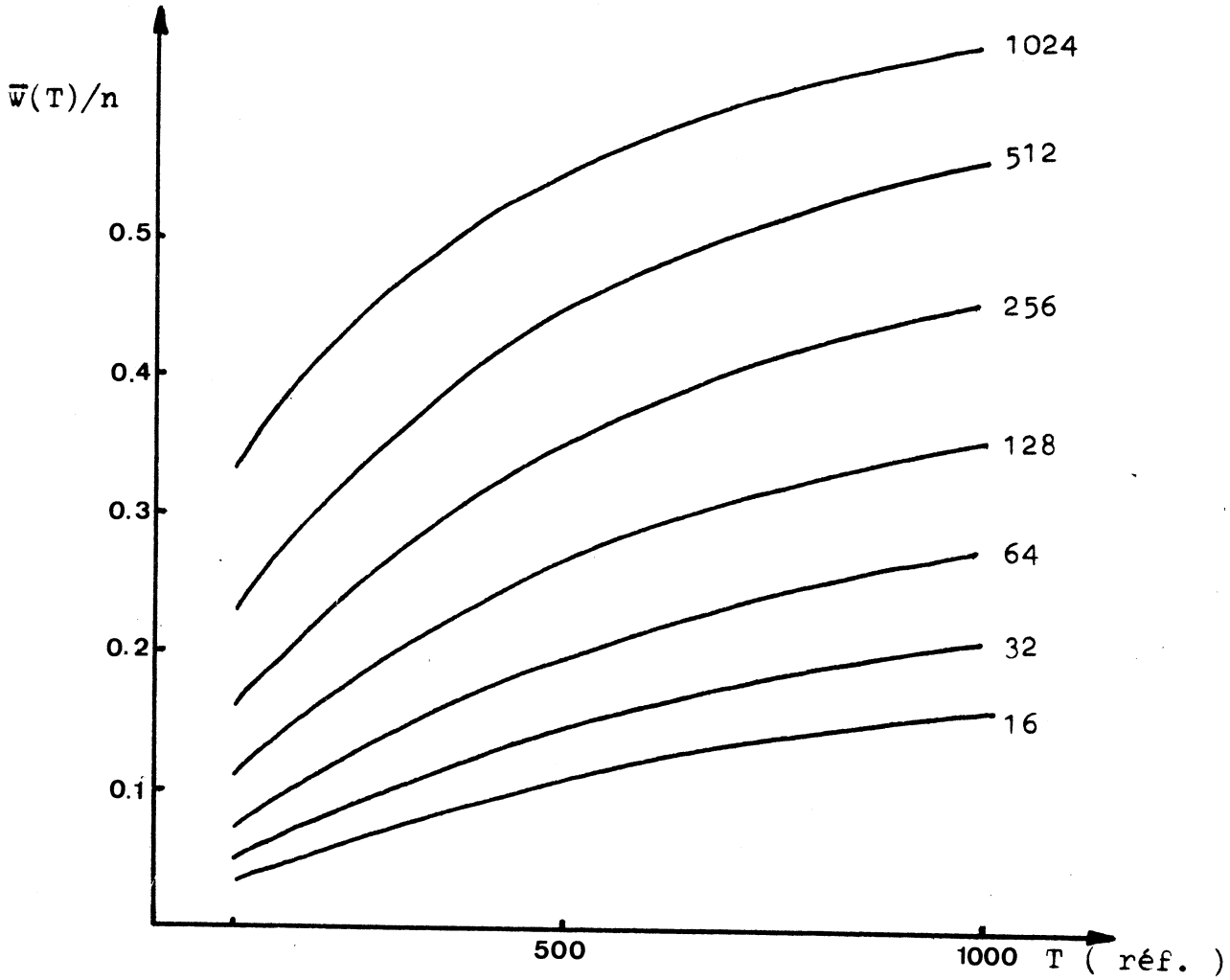


FIGURE 1

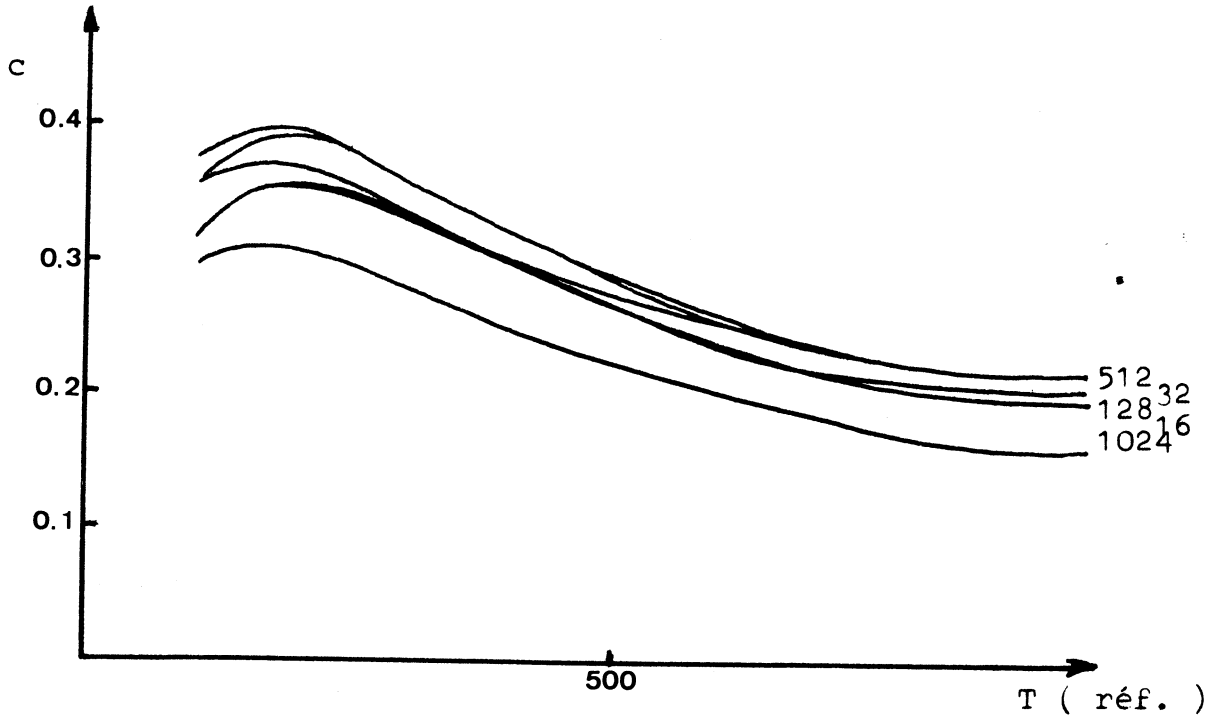


FIGURE 2

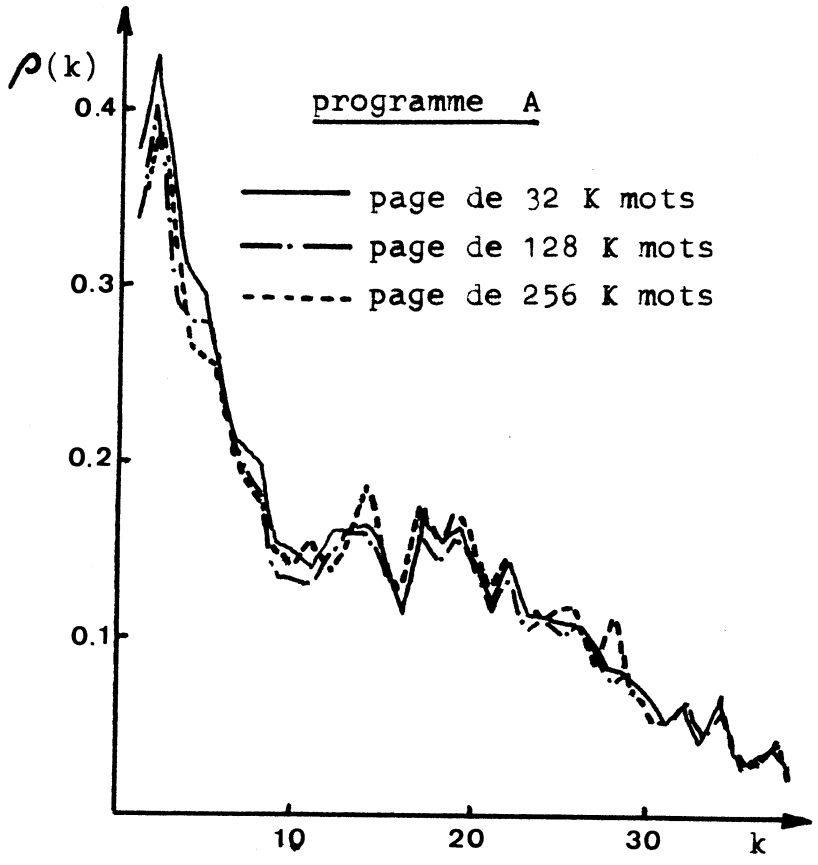


FIGURE 3

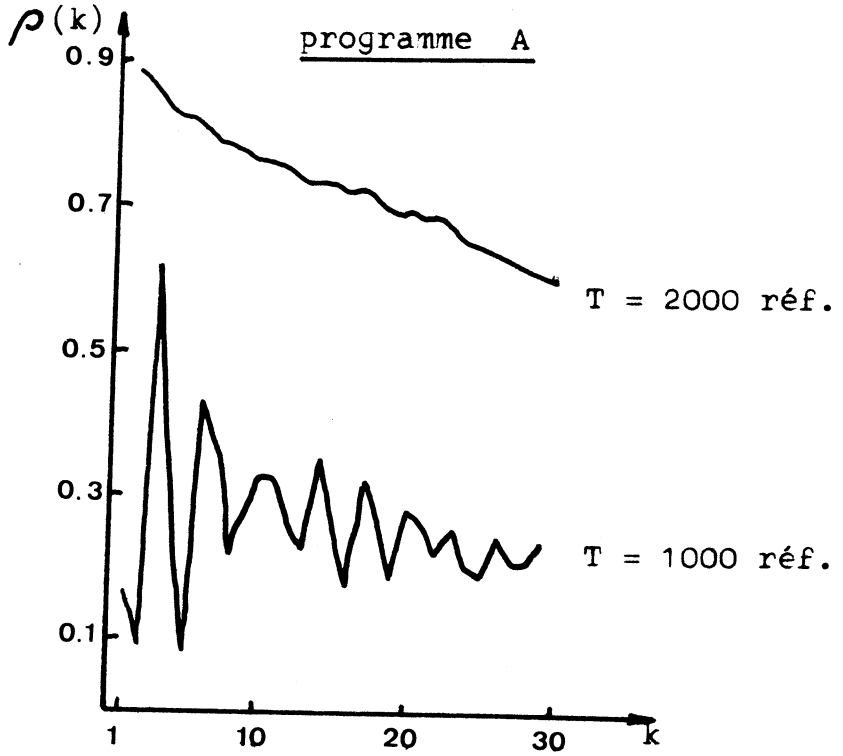


FIGURE 4

2.2. - Le modèle de durée de vie

L'étude des performances des systèmes à mémoire virtuelle exige de connaître le comportement des programmes dans leur espace d'adressage Plus précisément, lorsque cette étude est faite à l'aide de modèles à files d'attente, le principe de ces modèles impose que les besoins en mémoire des programmes soient représentés comme des requêtes sur un ou plusieurs serveurs du réseau. C'est-à-dire que des besoins exprimés initialement en terme d'espace doivent être transformés en besoins exprimés en terme de temps. Dans le cadre des systèmes gérés suivant la régle de la page à la demande cette transformation est faite par fonction de durée de vie (FDV) d'un programme qui relie le temps moyen $\bar{e}$ entre deux défauts de page durant l'exécution de ce programme au nombre de pages s qui lui ont été allouées en mémoire centrale sous la forme d'une fonction $\bar{e} = f(s)$/

Cette définition proposée par Belady et Kuehner [4] a l'inconvénient de définir la notion de durée de vie dans un contexte d'exécution du programme. En particulier, pour une même allocation de pages, la durée de vie dépend de la politique de gestion de la mémoire du nombre de pages initialement chargées et de la durée totale d'exécution du programme [33] . Nous désignerons par la suite fonction de durée de vie réelle (FDVR) la fonction de durée de vie ainsi définie.

Si l'on souhaite caractériser les propriétés de comportement d'un programme indépendamment du contexte d'exécution, il convient de disposer d'une définition différente. Nous définissons la fonction de durée de vie intrinsèque (FDVI) d'un programme comme la relation entre le temps moyen entre deux défauts de page et le nombre de pages de ce programme chargées en mémoire. Remarquons que la FDVI n'est pas totalement indépendante du contexte d'exécution car elle dépend, comme la FDVR, de l'algorithme de remplacement de pages utilisées. Nous supposons dans cette étude que l'algorithme de remplacement est fixé et nous renvoyons à [18,25] pour l'analyse des effets de l'algorithme de remplacement sur la durée de vie.

Nous présentons dans les paragraphes suivants des données expérimentales sur la FDVI de différents programmes obtenus à partir de traces d'exécution simulées des programmes en utilisant la technique du "stack processing" pour calculer la fonction de durée de vie intrinsèque . Nous analysons ensuite un modèle de comportement de programme dans un contexte d'exécution permettant de passer de la FDVI à la FDVR d'un programme suivant la politique de gestion

de la mémoire utilisée et le comportement du programme pour les opérations d'entrée-sortie.

2.2.1. - Caractérisation expérimentale de la FDVI

Les données ont été obtenues à partir des mêmes traces que celles présentées dans le paragraphe consacré à la caractérisation expérimentale des ensembles de travail et l'algorithme de remplacement retenu est l'algorithme LRU. Les résultats présentés portent sur la validation d'un modèle simple de la FDVI, sur l'estimation des paramètres de ce modèle, et enfin sur la distribution des intervalles de temps entre défauts de page. Ils sont analysés en fonction de l'utilisation de la durée de vie d'une part comme modèle de comportement de programme, d'autre part comme estimateur des besoins de mémoire d'un programme pouvant permettre de dégager des principes de contrôle du degré de multiprogrammation.

a) Modèles de la fonction de durée de vie intrinséque

Le modèle le plus généralement utilisé est celui proposé par Belady et Kuehner, bien que, comme nous l'avons vu, il était destiné à l'origine à caractériser la durée de vie réelle d'un programme. Cependant, comme nous pourrons l'observer dans le paragraphe suivant consacré à la FDVR, il s'approche beaucoup plus de la FDVI que de la FDVR. Ce modèle que nous noterons BK, s'écrit :

$$(1) \qquad \bar{e} = \alpha\, s^{k},$$

où $\bar{e}$ et s sont définis comme précédemment. Dans toute la suite les unités choisies pour $\bar{e}$ et s sont la référence et le K mots de 32 bits. La forme du modèle BK indique que, tracé sur une échelle logarithmique, $\bar{e} = \bar{e}(s)$ doit apparaître comme une droite. Le résultat observé pour le programme A avec des tailles de pages variant de 32 à 1024 mots est reporté sur la figure 5 et montre que le modèle BK tend à sous-estimer $\bar{e}$. Afin de prendre en compte ce phénomène, trois autres modèles ont été proposés par BURGEVIN et LEROUDIER [28]. Ces modèles notés BL_1, BL_2 et BL_3 sont définis de la façon suivante

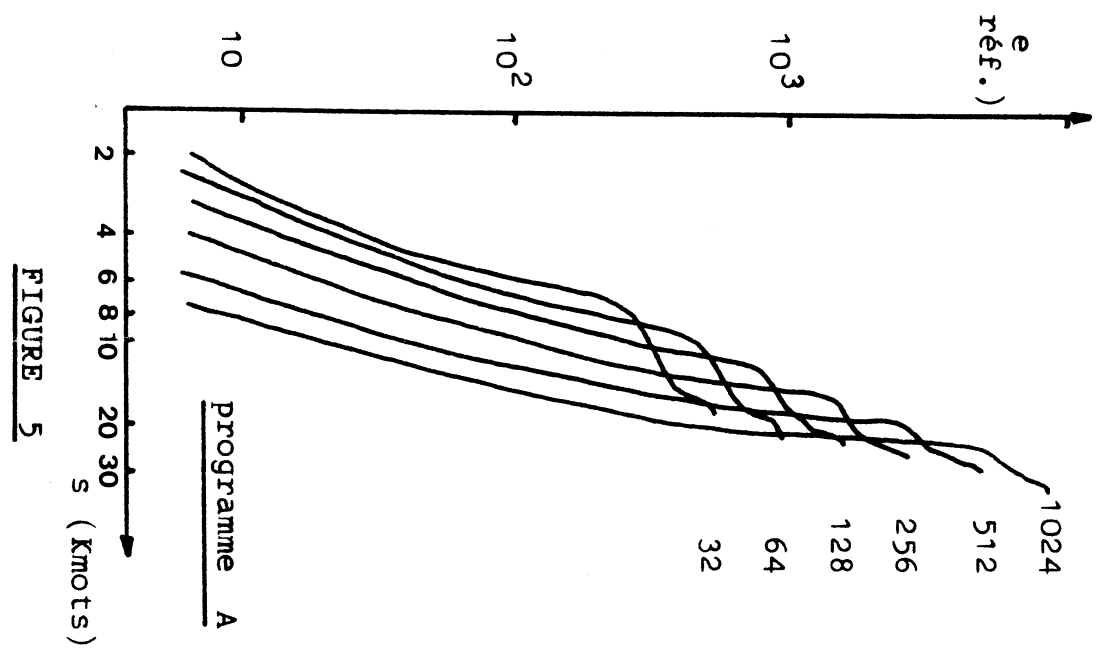


FIGURE 5

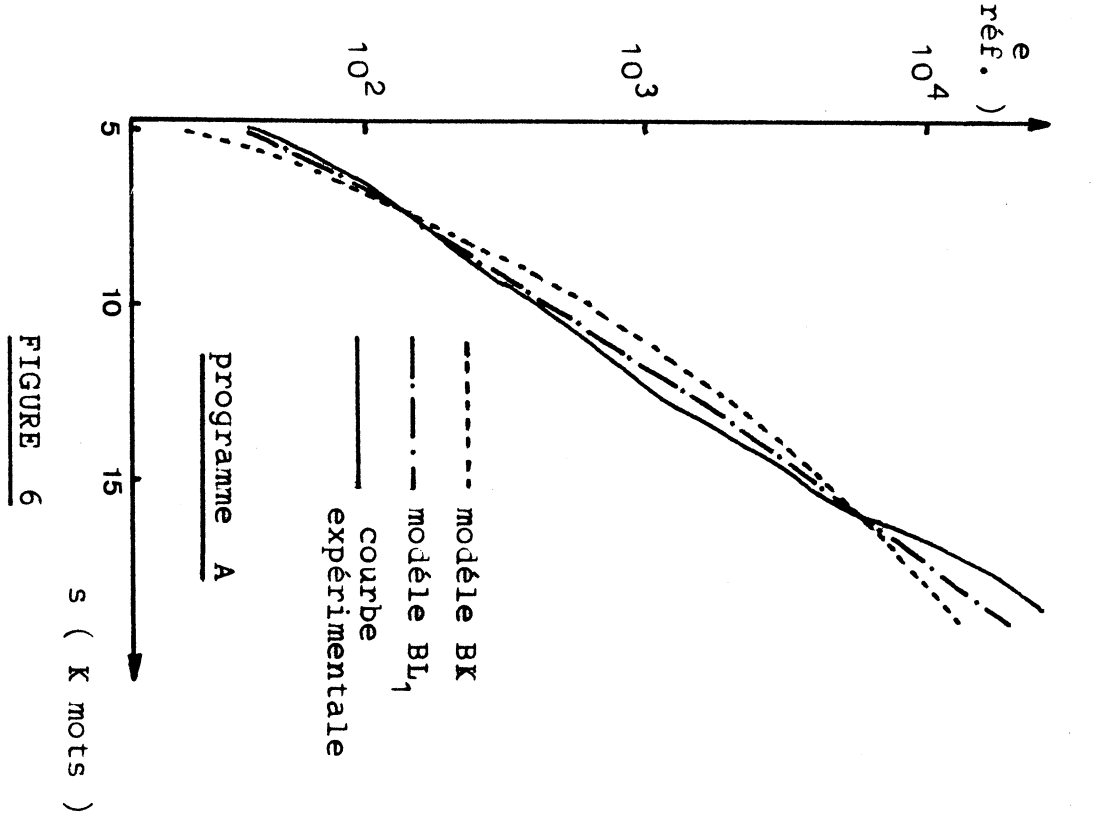


FIGURE 6

(2) $\bar{e} = \exp(\alpha s^k)$ (modèle BL_1)

(3) $\bar{e} = \exp[\alpha (Logs)^k]$ (modèle BL_2)

(4) $\bar{e} = \exp[\exp[\alpha s^k]]$ (modèle BL_3)

Pour chacun de ces modèles, les paramètres ont été estimés par une approximation des moindres carrés par rapport à la courbe expérimentale. Le meilleur résultat a été obtenu pour le modèle BL_1, cette conclusion est illustrée par la figure 6 où sont tracés la courbe expérimentale e(s), le modèle BK et le modèle BL_1.

Remarquons que dans les deux modèles BK et BL_1, la fonction $\bar{e}(s)$ est caractérisée par deux paramètres α et k. La valeur de ces paramètres et leurs dépendances vis à vis de la taille de page sont maintenant précisées.

b) Estimation des paramètres α et k

Pour les deux modèles BK et BL_1 les paramètres α et k ont été estimés pour les cinq programmes considérés avec différentes tailles de pages. Les résultats sont présentés sur les figures 7a,7b pour le paramètre α et les figures 8a,8b pour le paramètre k. Il est important de noter les valeurs relativement élevées de k qui pour des tailles de page courantes sont de l'ordre de 3 à 5, et atteignent 7 et 8 pour les programmes D et E. Les résultats sont à rapprocher des valeurs proposées par BELADY pour la FDVR avec k situé entre 1 et 2. Cette différence illustre bien le comportement dissemblable de la FDVI et de la FDVR que nous mettrons en évidence dans le paragraphe suivant. Notons également que les deux paramètres α et k ne varient pas indépendamment. Plus précisément, Leroudier [29] a montré qu'il existait pour tout programme une relation simple entre α et k de la forme :

(5) $k \simeq - \mathrm{Log}_{10} \alpha$

c) Distribution des intervalles entre défauts de page

La fonction de durée de vie caractérise la moyenne des intervalles entre deux défauts de page consécutifs d'un programme. Toutefois, aussi bien pour la modélisation des performances globales d'un système que

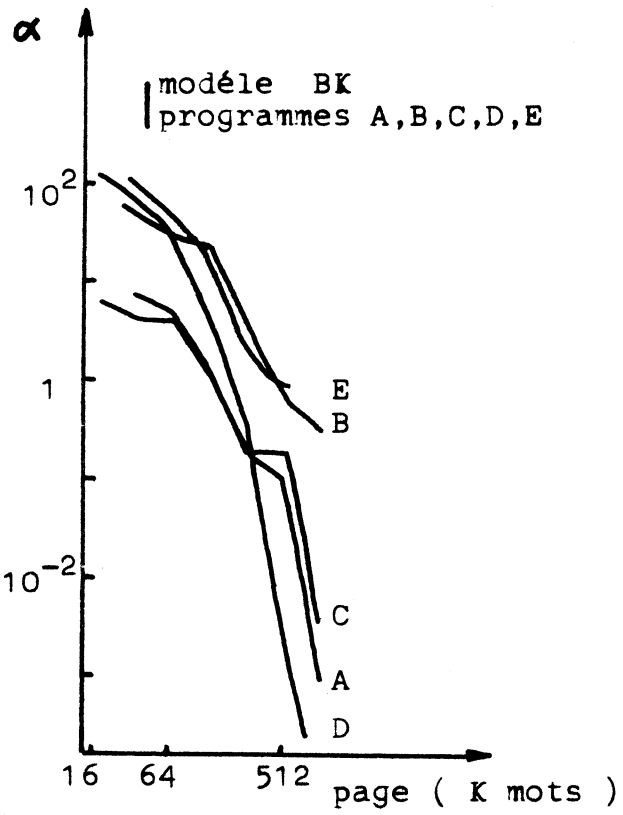


FIGURE 7a
estimation du paramètre α

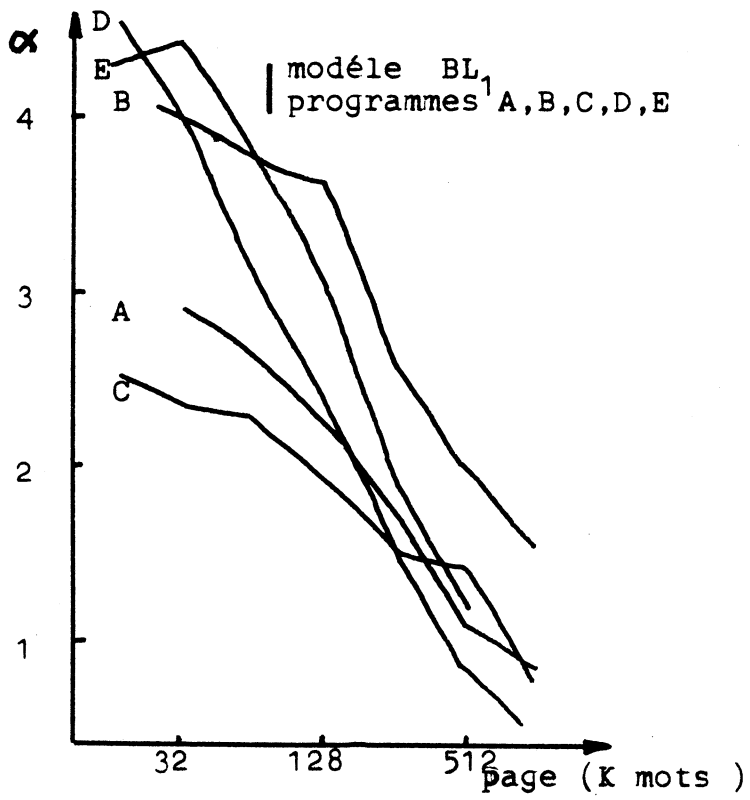


FIGURE 7b
estimation du paramètre α

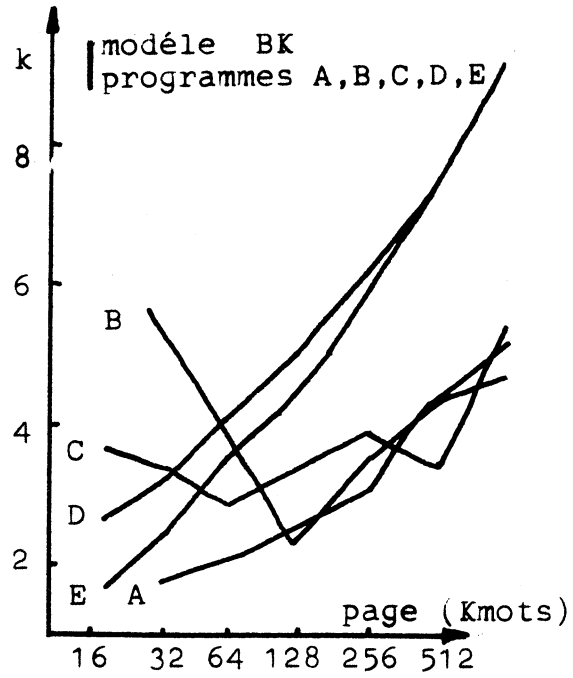


FIGURE 8a
estimation du paramètre k

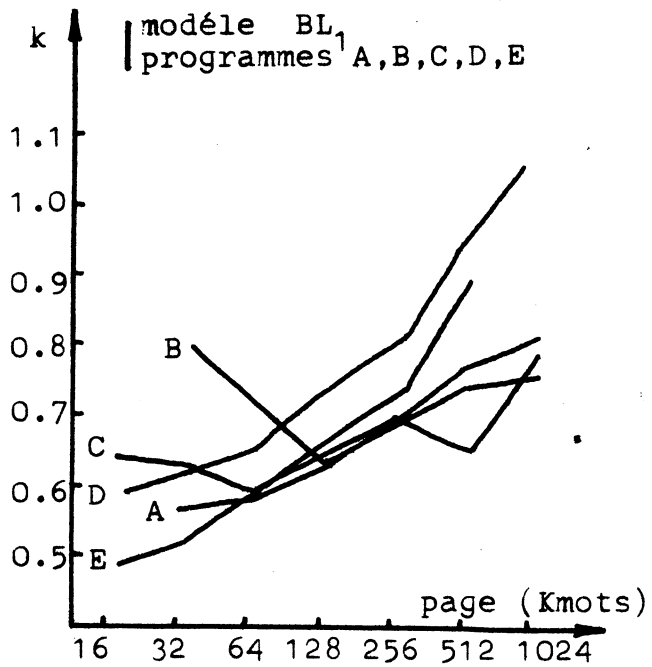


FIGURE 8b
estimation du paramètre k

pour l'estimation des besoins en mémoire des programmes à l'aide de la fonction de durée de vie, il est nécessaire de connaitre plus complétement la distribution des intervalles de temps entre défauts de page.

Une première indication supplémentaire sur ces distributions est donnée par le coefficient de variation c de la distribution défini comme le rapport de l'écart type à la moyenne. Le coefficient mesure donc la dispersion des quantités mesurées autour de leur moyenne. Rappelons que ce coefficient est égal à 1 pour la distribution exponentielle. Le coefficient de variation de la distribution des intervalles de temps entre défauts de page a été calculé pour le programme C pour différentes tailles de page en fonction du rapport "taille de l'espace alloué/taille de l'espace d'adressage"et les résultats sont présentés sur la figure 9 Ils mettent en évidence la dispersion importante des intervalles de temps entre défauts de page, et on peut observer que la dispersion est d'autant plus grande que la part du programme chargée en mémoire est petite.

La distribution des intervalles de temps entre deux défauts de page peut être précisément caractérisée en essayant de l'approcher par une distribution connue. Les figures 11a et 10b illustrent l'approximation obtenue par une distribution hyperexponentielle à deux phases de la forme [38]

$$(6) \qquad f(x) = \rho_1 (1-e^{-\lambda_1 x}) + \rho_2 (1-e^{-\lambda_2 x}), \rho_1 + \rho_2 = 1$$

d) La fonction d'autocorrélation des intervalles de durée de vie

La fonction d'autocorrélation des intervalles de durée de vie a été calculé pour le programme A avec une taille de page de 256 mots et une taille mémoire de 48 pages alors que le programme adresse 104 pages. Le résultat rapporté sur la figure 11 indique très nettement l'absence d'autocorrélation significative. Cette observation appelle deux remarques importantes : la première est que l'hypothsèe d'indépendance des intervalles de durée de vie faite lorsque le modèle de durée de vie est utilisé dans un modèle global à file d'attente est justifiée ; la seconde est qu'il est difficile de construire un bon prédicteur de la fonction de durée de vie en raison de la faible corrélation entre les mesures.

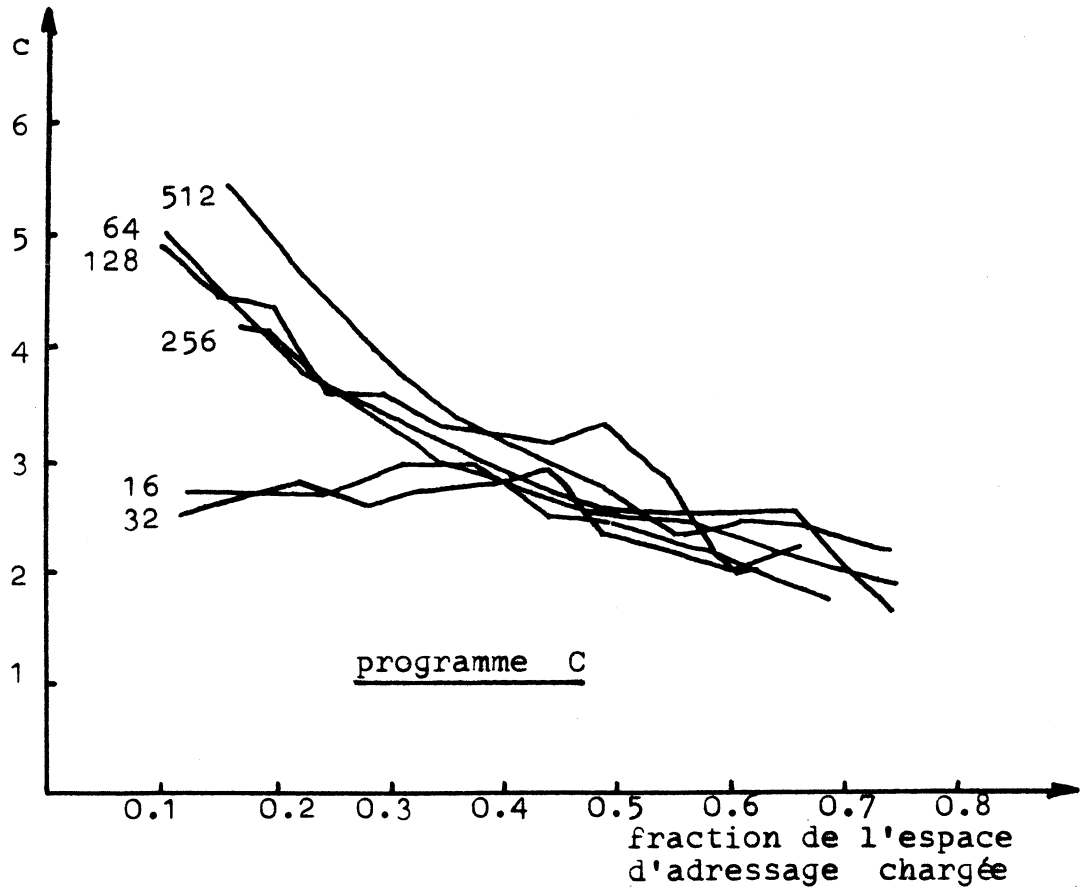


FIGURE 9

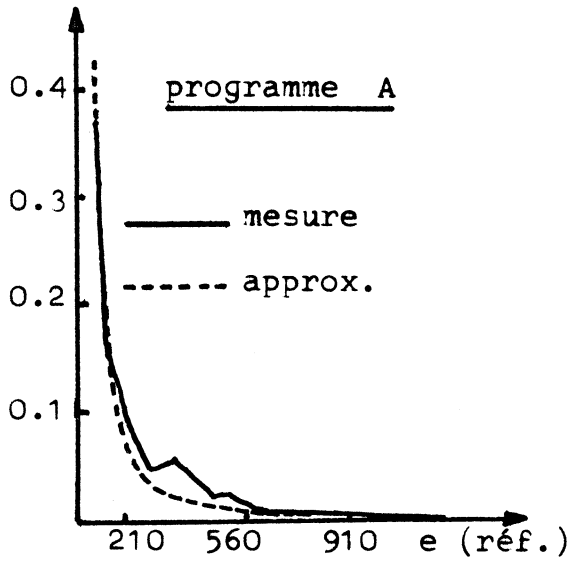


FIGURE 10a

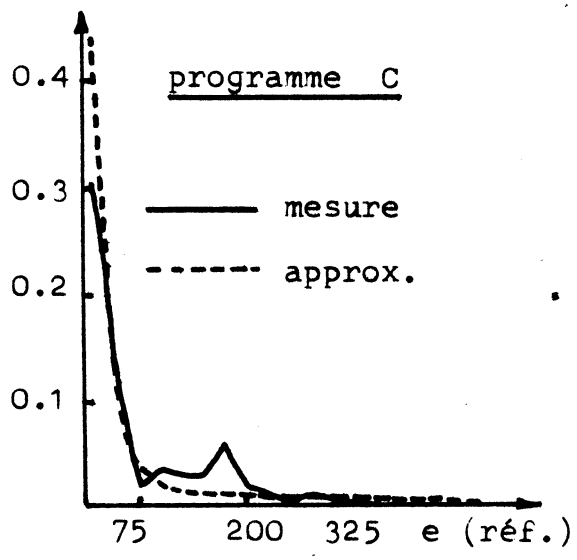


FIGURE 10b

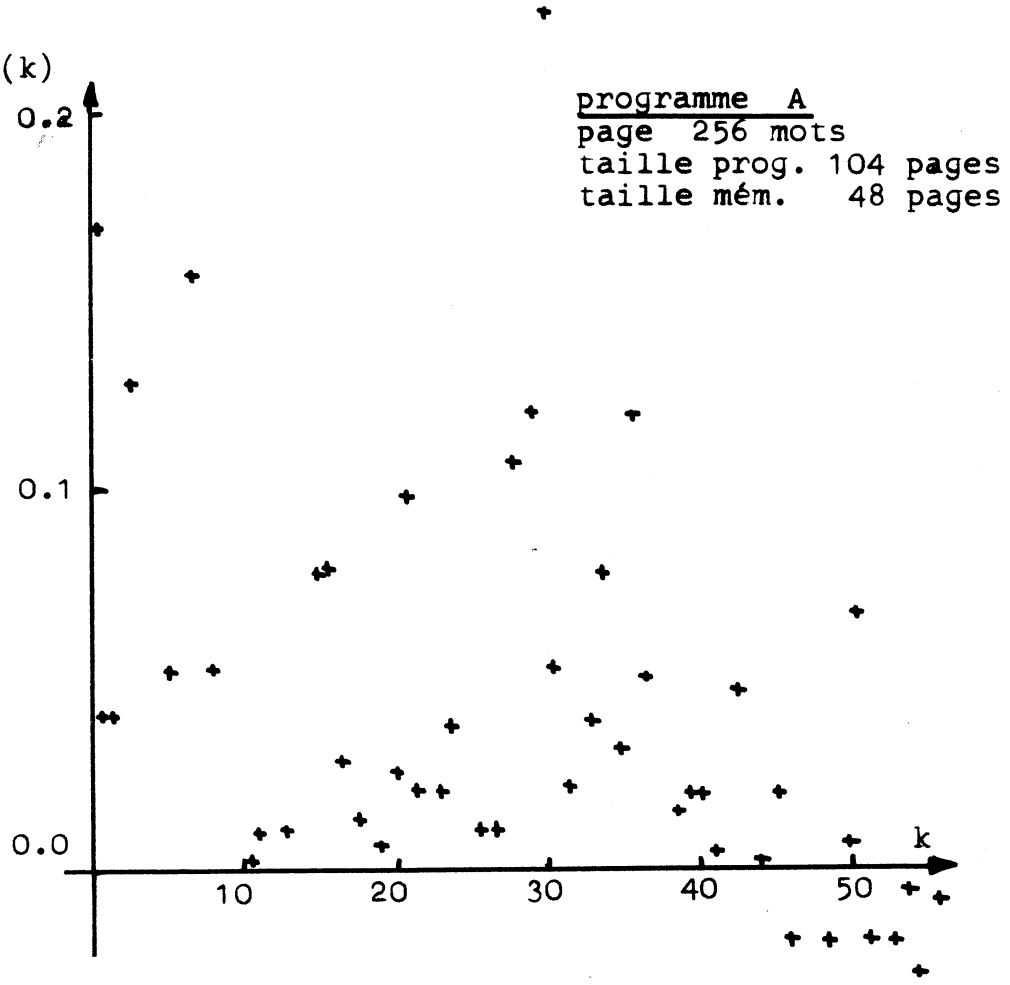

(k)
0.2
0.1
0.0
10
20
30
40
50
k
programme A
page 256 mots
taille prog. 104 pages
taille mém. 48 pages

FIGURE 11

2.2.2. - Caractérisation de la FDVR par modèle

a) Le modèle analysé

Dans le contexte des systèmes multiprogrammés, l'exécution d'un programme consiste en une suite d'intervalles de traitements sur l'U.C. et d'interruptions dues aux requêtes d'entrée-sortie, aux défauts de page, aux fins de quantum et à la fin de l'exécution. Suivant la cause de ces interruptions et la politique d'allocation de mémoire, un programme débutant un nouvel intervalle U.C. possède encore, ou ne possède plus l'ensemble des pages chargées en mémoire centrale immédiatement avant la dernière interruption.

Dans cette étude, nous nous intéressons uniquement à deux types d'interruption : les interruptions pour défaut de page qui ont pour effet d'augmenter de une unité le nombre de pages chargées en mémoire centrale et les interruptions de programmes telles que,durant l'intervalle de temps écoulé entre l'interruption et la réactivation du programme, certaines de ses pages ont pu être déchargées. Nous ferons les hypothèses suivantes :

H1 : les intervalles de temps, en temps virtuel, entre deux interruptions de programme, comme définies plus haut, sont des variables aléatoires, indépendantes et identiquement distribuées suivant une loi exponentielle de moyenne T ,

H2 : le nombre maximum M de pages allouées à un programme est fixe, et les pages sont chargées en mémoire suivant le principe de la page à la demande ;

H3 : les intervalles de temps entre deux défauts de page consécutifs d'un programme sont des variables aléatoires indépendantes, identiquement distribuées suivant une loi exponentielle de moyenne q_i lorsque i pages du programme sont chargées en mémoire. Les q_i définissent la FDVI du programme.

H4 : Les transitions entre le nombre i de pages du programme chargé en mémoire lorsque ce programme est interrompu - interruption de programme - et le nombre j de pages chargées en mémoire lorsqu'il débute un nouvel intervalle U.C. sont décrites par une chaîne de Markov du premier ordre de matrice de transition (α_{ij})

considérer que les intervalles de temps entre défauts de page sont des variables aléatoires exponentielles dont la moyenne ne dépend que du nombre de pages présentés en mémoire. Comme nous l'avons vu précédemment, les résultats expérimentaux montrent que cette hypothèse n'est pas déraisonnable, mais que la distribution des intervalles est hyperexponentielle. Par ailleurs, l'hypothèse H3 a fait la preuve que son intérêt dans plusieurs modèles de prédiction de programme de système à mémoire virtuelle [34,39] .

Etant donné les hypothèses H1-H4, l'exécution du programme en temps virtuel se caractérise par une suite d'intervalles U.C. séparés par des interruptions classées en deux catégories : interruptions pour défaut de page et interruptions de programme.

b) Calcul du comportement stationnaire.

Définissons l'état X(t) à un instant t du temps virtuel d'un programme comme le nombre de ses pages chargées en mémoire centrale. L'ensemble des états est :

$$E = \{1, 2, \ldots, M\}.$$

Suivant les hypothèses H1-H4, le processus X(t) est un processus semi-markovien que nous pouvons analyser de la façon suivante : soit $P = (p_{ij})$ sa matrice de transition. Les transitions dépendent de deux évènements différents : un défaut de page ou une interruption de programme. En vertu des hypothèses exponentielles H1-H3, la probabilité Θ_i que le prochain évènement soit un défaut de page alors que le programme est dans l'état i s'écrit :

$$\Theta_i = \frac{1/q_i}{1/q_i + 1/T}, \tag{7}$$

et le temps moyen de résidence dans l'état i, e_i, est donné par :

$$e_i = \frac{1}{1/q_i + 1/T} \tag{8}$$

Avec la probabilité $1-\Theta_i$ le programme est interrompu dans l'état i et sera réactivé dans l'état j avec la probabilité α_{ij} suivant l'hypo-

$$(9)\quad Pij = \begin{cases} \Theta_i & i = 1,\ldots,M-1 \ ; \ j = i+1 \\ (1-\Theta_i)\alpha_{ij} & i = 1,\ldots,M-1 \ ; \ j = 1,\ldots i \\ (1-\Theta_M)\alpha_{Mj} & i = M \ ; \ j = 1,\ldots M-1 \\ (1-\Theta_M)\alpha_{MM} + \Theta_M & i = M \ ; \ j = M \\ o & \text{sinon} \end{cases}$$

soit $\Pi = (\Pi_1,\ldots,\Pi_M)$ le vecteur de probabilités défini par

$$(10)\quad \begin{cases} \Pi = \Pi P \\ \sum_{i=1}^{M} \Pi_i = 1 \end{cases}$$

Π_i représente la probabilité stationnaire que le programme soit dans l'état i aux instants de transition. Nous avons alors le vecteur de probabilités $\Gamma=(\gamma_1,\ldots,\gamma_M)$, où γ_i représente la probabilité d'équilibre que X(t) soit égal à i , à partir des Π_i et e_i

$$(11)\quad \gamma_i = \Pi_i e_i / \left(\sum_{1}^{M} \Pi_j e_j\right),\ i=1,\ldots M$$

Nous en déduisons le nombre moyen m de pages du programme chargées en mémoire, ce nombre moyen étant calculé dans le temps virtuel du programme, par

$$(12)\quad m = \sum_{i=1}^{M} i\,\gamma_i$$

et le temps réel r_M entre défauts de page lorsqu'un nombre maximum M de pages est alloué au programme donné par

$$(13)\quad r_M = \sum_{i=1}^{M} q_i\,\gamma_i$$

avec $m \leq M$

$r_M \leq q_M$

A partir de l'équation (12), nous pouvons définir le taux d'utilisation m/M de la mémoire qui représente le pourcentage des pages allouées au programme réellement utilisées. L'équation (13) définit la FDVR du programme r_M à partir de la FDVI q_i.

Il reste à caractériser les matrices (σ_{ij}) qui définissent les politiques de gestion de mémoire. Quatre modèles différents sont pris en compte, notés modèles A,B,C et D. La description de ces modèles et les expressions de probabilités Π_i obtenues pour chacun d'eux sont présentées dans l'Annexe A.

c) Résultats numériques

Les résultats numériques ont été obtenus en caractérisant la FDVI q_i, i = 1,2, ... des programmes par le modèle de BELADY étudié et validé expérimentalement plus haut :

$$q_i = \alpha i^{k}$$

Les figures 14a,b et 15 présentent la FDVR, ou fonction de durée de vie réelle obtenue pour le modèle de fonction de durée de vie intrinsèque (FDVI) choisi, ainsi que le taux d'utilisation de la mémoire centrale, pour différentes valeurs du temps moyen entre deux interruptions de programme T et de l'exposant k du modèle de FDVI.

Pour tous les modèles étudiés, nous observons un tassement de la FDVR par rapport à la FDVI, et suivant les paramètres T et k, des écarts très sensibles entre la FDVR et la FDVI peuvent être notés. Notons que des écarts augmentent évidemment lorsque T diminue, mais aussi quand k croît, et ne peuvent être négligés pour des valeurs typiques de k, entre 2 et 4. Le taux d'utilisation de la mémoire décroît également rapidement avec le nombre de pages allouées aux programmes et ce phénomène indique bien la sous-utilisation de la mémoire que la technique de la page à la demande peut provoquer dans certaines conditions.

Ces résultats montrent l'importance d'une distinction entre la FDVI d'un programme et la FDVR de ce même programme dans un environnement d'exécution. Ils seront plus précisément illustrés lorsque nous aurons décrit et analysé le modèle de système multiprogrammé à mémoire virtuelle que nous utilisons, et pourrons mettre en évidence l'écart obtenu sur les performances du système lorsque la FDVI est utilisée en place de la FDVR.

Remarquons que la comparaison des courbes de la FDVI et de la FDVR montrent bien pourquoi les estimations du paramètre k dans le modèle BK de la fonction de durée de vie diffèrent suivant que l'on considère, comme l'ont fait BELADY et KUEHNER [4] , la FDVR, avec alors des valeurs de k comprises entre 1 et 2, ou que l'on considère la FDVI avec, à partir des estimations de BURGEVIN et LEROUDIER [28] , des valeurs de k beaucoup plus importantes. Nous pouvons en conclusion noter, comme l'a montré cette analyse, que la notion de durée de vie est fortement liée au contexte d'exécution d'un programme, et que donc la mesure de la durée de vie d'un programme ne fournit pas une information uniquement liée au comportement du programme dans son espace d'adressage.

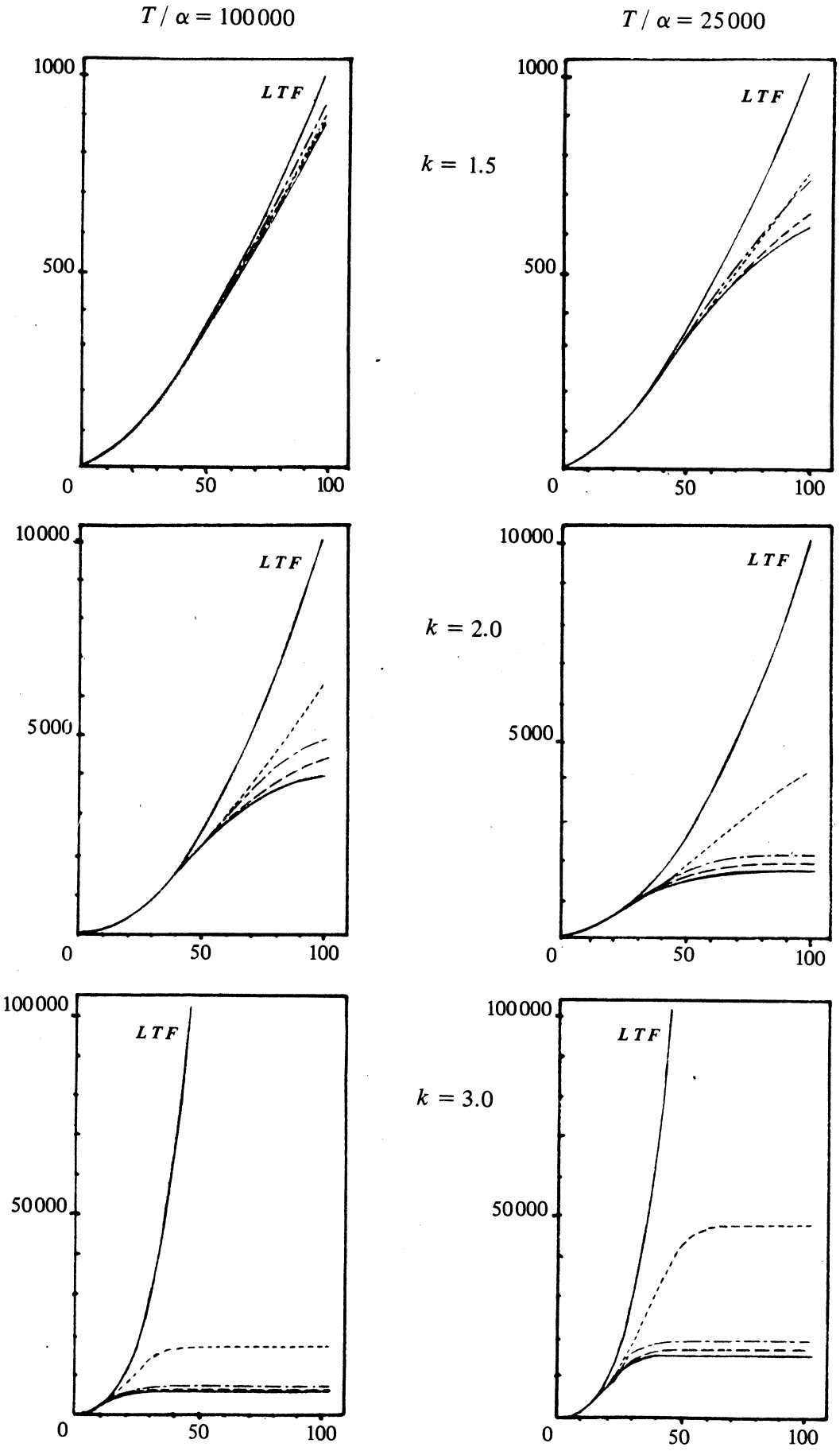


Figure 14a : FDVR (e/α en fonction de M)

(-mod.A ;-.-mod.B ;-.-mod.C(β=0.2) ;-.-mod.C(β=0.8))

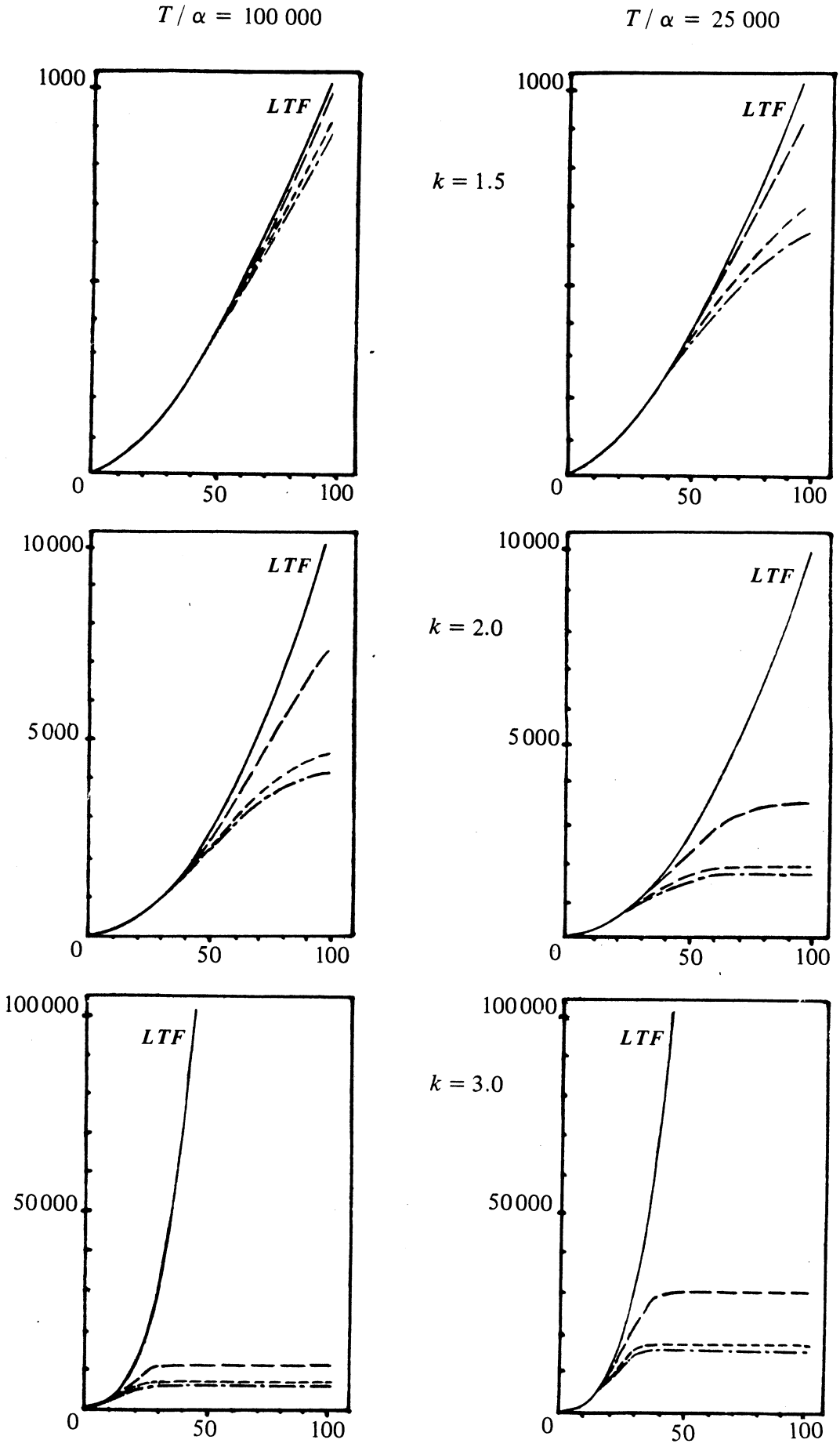


Figure 14b : FDVR (e/α en fonction de M)

(mod.D ;...γ=10. ;...γ=1. ;...γ <0.1)

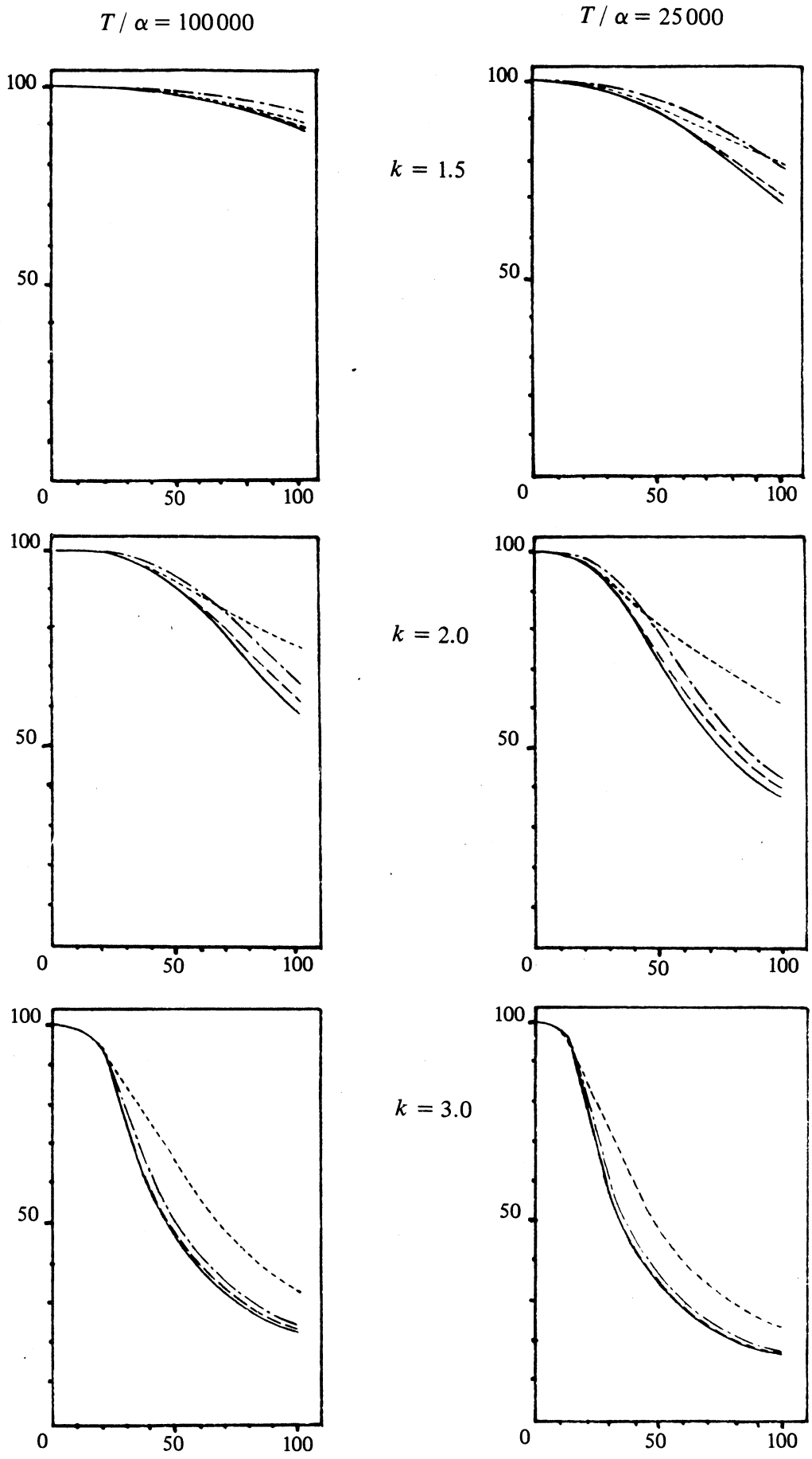


Figure 15 : utilisation de la mémoire

(-mod.A ;...mod.B ;...mod.C(β=0.2) ;...(β=0.8))

3. - DYNAMIQUE DU SYSTEME

3.1. - Le modèle d'architecture à mémoire virtuelle

Le modèle est construit pour représenter les fonctions fondamentales du système étudié :

- allocation de l'unité centrale,
- gestion de la mémoire,
- gestion des entrées-sorties,

et le comportement des programmes vis-à-vis de ces ressources. Chacune des fonctions citées est caractérisée de façon globale, sans que soient détaillés les mécanismes associés à chaque traitement : ainsi une opération d'entrée-sortie sur disque sera uniquement représentée par un temps de service,somme du temps d'accès(déplacement du bras et délai de rotation) et du temps de transfert.

Les évènements auxquels donne lieu l'exécution des programmes sont décrits comme des processus stochastiques. Ces évènements sont supposés indépendants entre eux et nous analysons le modèle dans l'hypothèse où tous les programmes sont représentés par le même modèle de comportement. L'architecture du système est modélisé par un réseau de files d'attente où chaque ressource du système est représenté par un serveur et la file d'attente associés. L'analyse du réseau est conduite pour mettre en évidence les phénomènes de saturation et d'attente liés au partage des ressources.

En fonction de ces hypothèses de base, le modèle représenté sur la figure 16 est défini comme suit :

a) Les consoles

Le comportement des utilisateurs est modélisé par leur temps de réflexion τ, c'est-à-dire le temps qui s'écoule entre l'instant ou l'utilisateur reçoit le contrôle de sa console et l'instant où il envoie une nouvelle commande au système. Soit $F_d(t)$ la fonction de distribution de τ, et d sa valeur moyenne. Remarquons que d=0 signifie que le nombre de programmes dans le système est maintenu constamment égal au nombre de consoles noté N.

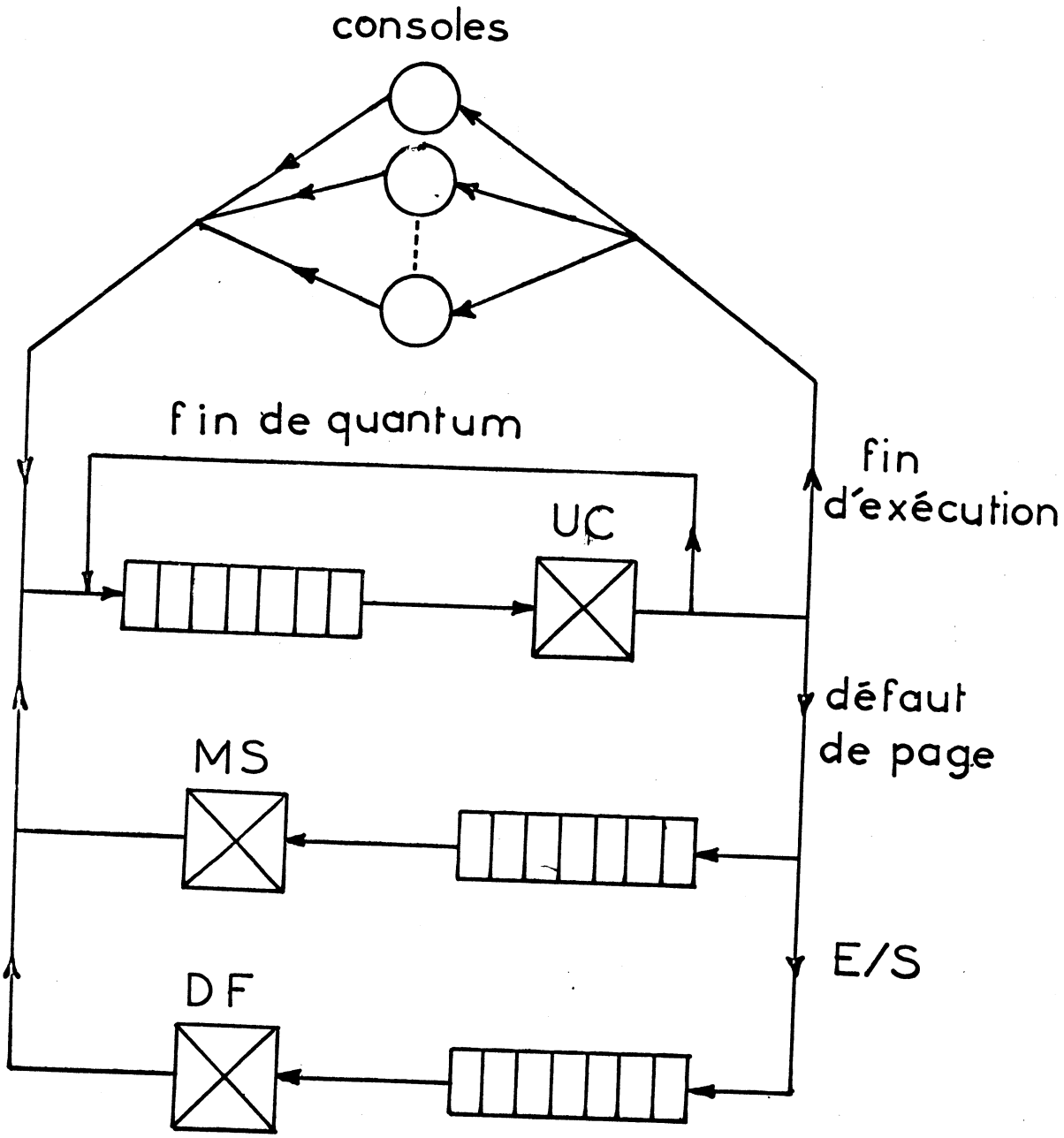

consoles
fin de quantum
fin
d'exécution
UC
défaut
de page
MS
E/S
DF

FIGURE 16

b) Le système

i - Gestion de l'unité centrale (UC)

Les programmes arrivant des consoles sont mis en file d'attente derrière l'UC dans l'ordre de leur arrivée dans cette file. L'exécution d'un programme sur l'UC peut être interrompue pour quatre raisons différentes :

1 - fin de quantum : auquel cas le programme est mis en attente d'UC.

2 - défaut de page : dans ce cas le programme est mis en attente derrière la mémoire secondaire.

3 - entrée-sortie : dans ce cas le programme est mis en attente derrière le disque de fichier.

4 - fin d'exécution : dans ce cas le programme quitte le système et le contrôle est rendu à la console qui l'avait généré.

ii - Gestion de la mémoire secondaire (MS)

Les demandes de transfert de page arrivant à la MS sont traitées dans l'ordre "premier-arrivé-premier-servi". Le fonctionnement détaillé de la MS n'est pas pris en compte et il est uniquement représenté par le temps total nécessaire pour traiter un défaut de page. Ce temps est considéré comme une variable aléatoire de distribution $F_{MS}(t)$ et de moyenne T_{MS}.

Remarquons qu'il correspond en général à deux opérations, une écriture et une lecture qui peuvent se réduirent à une lecture dans le cas où la page remplacée n'a pas été modifiée. La valeur moyenne T_{MS} représente donc le temps moyen de traitement d'un défaut de page par la mémoire secondaire, y compris éventuellement la réécriture de la page déchargée.

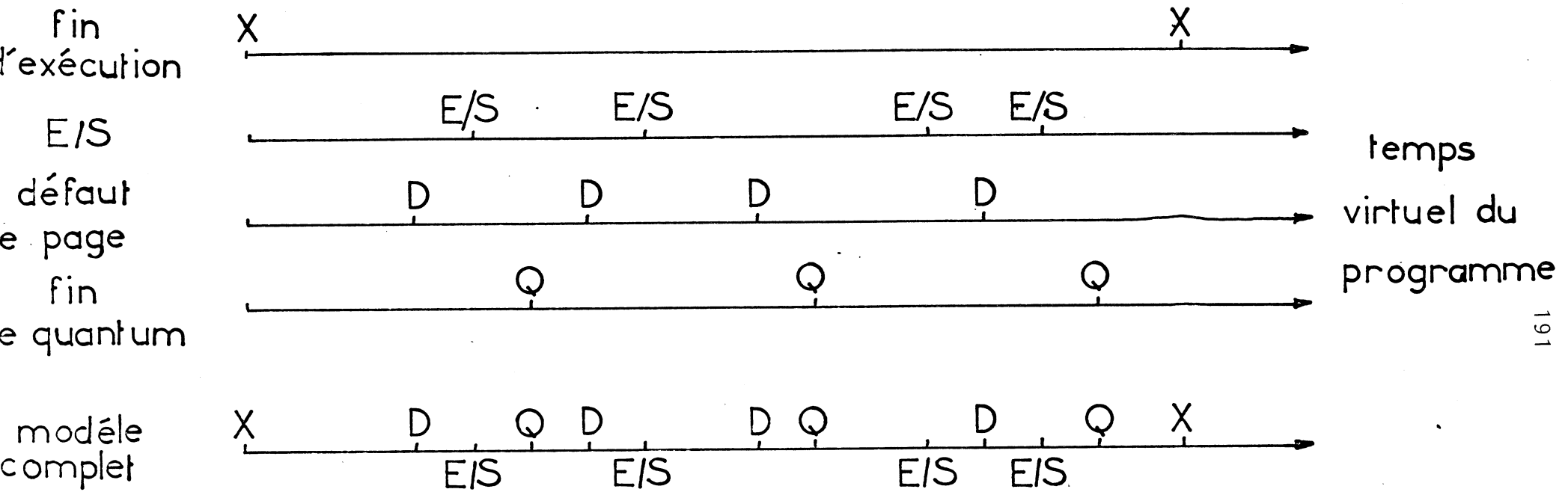

fin d'exécution
X
X
E/S
E/S
E/S
E/S
E/S
temps
défaut de page
D
D
D
D
virtuel du programme
fin de quantum
Q
Q
Q
modéle complet
X
D
Q
D
D
Q
D
Q
X
E/S
E/S
E/S
E/S

FIGURE 17

iii- Gestion du disque de fichiers (DF)

Le disque de fichiers est géré comme la MS et son fonctionnement est uniquement caractérisé par le temps total nécessaire pour exécuter une demande d'E/S. Comme précédemment, ce temps est considéré comme une variable aléatoire de distribution $F_{DF}(t)$ et de moyenne T_{DF}.

c) Le comportement des programmes

L'exécution d'un programme sur l'UC donne lieu à différents évènements provoqués par la demande ou la libération des ressources UC, MS et DF. En raison de l'hypothèse de base d'indépendance de ces différents évènements, le modèle de comportement de programme utilisé est obtenu par la superposition des modèles de comportement vis-à-vis de chacune des ressources. Nous avons donc successivement :

- le modèle de temps UC total requis pour l'exécution d'un programme considéré comme une variable aléatoire de distribution $F_c(t)$ et de moyenne c ;
- le modèle d'entrée-sortie : le temps virtuel entre deux E/S d'un programme est considéré comme une variable aléatoire de moyenne r distribuée suivant $F_r(t)$;
- le modèle de défaut de page : le temps virtuel entre deux défauts de page consécutifs d'un programme est considéré comme une variable aléatoire de moyenne e distribuée suivant $F_e(t)$;
- le modèle d'allocation de quantum UC : la durée d'un quantum est supposée constante et égale à q.

La figure 17 représente ces quatre modèles et leur superposition qui fournit le modèle de comportement.

Le temps moyen UC total et le taux d'entrée-sortie sont intrinsèques au programme et leurs valeurs ne dépendent donc pas de l'état du système. En revanche, le taux de défaut de page n'est pas intrinsèque au programme et dépend en particulier de l'allocation de la mémoire physique. Pour représenter cette dépendance, nous utilisons les modèles de durée de vie présentés précédemment qui relient le temps moyen e entre deux défauts de page consécutifs d'un programme au nombre de pages m qui lui sont allouées. Cette relation s'écrit :

(28) $$e = e(m)$$

Les programmes étant représentés par le même modèle de comportement, on suppose qu'en moyenne ils se partagent également l'espace de mémoire disponible M, soit une allocation mémoire pour chaque programme de :

(29) $$m = \frac{M}{n},$$

et :

(30) $$e = e(M/n),$$

où n est le degré de multiprogrammation, c'est-à-dire le nombre de programmes se partageant la mémoire :

n = N - le nombre de consoles actives.

d) Les temps de gestion propre

Ces temps représentent les temps d'unité centrale propres à l'exécution du moniteur. Ils apparaissent à chaque interruption de service d'un programme sur l'unité centrale, et correspondent aux opérations de changement de contexte, de pagination, et de lancement d'une entrée-sortie. Ces quantités sont considérées dans le modèle comme constantes avec pour valeurs :

θ_c dans le cas de la fin d'exécution d'un programme,

θ_e dans le cas d'une faute de page,

θ_r dans le cas d'une entrée-sortie,

θ_q dans le cas d'une fin de quantum.

e) Les critères de performance

Nous nous intéressons à deux catégories de mesures de performances. D'une part, l'utilisation des ressources du système : UC, MS, DF, d'autre part le bénéfice obtenu par la multiprogrammation mesuré par le rapport.

$$D(n) = \frac{\text{Débit du système en multiprogrammation de degré n}}{\text{Débit du système en monoprogrammation}} .$$

D(n), appelé "dilatation" [2] , mesure la dilation du temps réel obtenu par la multiprogrammation de n programmes. L'utilisation de l'UC, de la MS et du DF pour un degré de multiprogrammation n donné sont notés respectivement, $A_{UC}(n)$ $A_{MS}(n)$ et $A_{DF}(n)$.

Dans la suite nous présentons les techniques de résolution du modèle qui permettent d'obtenir les mesures de performances ci-dessus définies en fonction d'une part des paramètres de l'architecture M, T_{MS}, T_{DF}, N et des paramètres de comportement de programmes c, r, e(m), d. Ces études sont menées à degré de multiprogrammation constant en supposant d=0, c'est-à-dire qu'un programme dont l'exécution est achevée est immédiatement remplacé par un autre programme et l'on a donc :

$$n = N.$$

Par la suite la notation n sera seule utilisée.

En résumé le modèle est défini par deux familles de paramètres : les paramètres de l'architecture qui décrivent la configuration, les caractéristiques des unités périphériques (MS, DF), et certains éléments du système d'exploitation :

n degré de multiprogrammation,

M nombre de pages de la mémoire physique,

T_{MS} temps moyen d'accès et de transfert d'une page par la MS,

T_{DF} temps moyen d'accès et de transfert d'un enregistrement sur la DF,

q valeur du quantum UC,

θ_c, θ_e, θ_r, θ_q temps de gestion propres,

les paramètres de comportement des programmes et des utilisateurs :

d temps moyen de réflexion aux consoles,

c temps moyen UC total requis pour l'exécution d'un programme,

r temps moyen virtuel entre deux E/S consécutives,

e(m) représentant la fonction de durée de vie.

Par ailleurs, les variables dont les moyennes sont définies ci-dessus sont plus complètement caractérisées par leurs lois de distribution $F_{MS}(t)$, $F_{DF}(t)$, $F_d(t)$, $F_c(t)$, $F_r(t)$, $F_e(t)$. Dans les deux paragraphes qui suivent nous établissons des relations simples entre les différents paramètres et les mesures de performances sans qu'il soit nécessaire de préciser les lois de distribution, et nous montrerons ensuite en calculant les mesures de performances que les résultats obtenus sont peu sensibles à la forme de ces lois.

3.2. - ANALYSE DU MODELE

3.2.1. - Propriétés générales du modèle

L'analyse détaillée du modèle présenté est donnée en [34] . Elle est fondée sur l'application des théorèmes de CHANG et LAVENBERG [12] et conduit à des relations simples reliant l'utilisation des ressources UC, MS, DF et les paramètres de l'architecture et de comportement des programmes.

Notons A'_{UC} l'utilisation de l'UC consacrée aux programmes. Nous obtenons :

$$(31) \qquad A'_{UC}(n) = A_{UC}(n) \frac{1}{1+\sigma_c/c+\sigma_e/e+\sigma_r/r+\sigma_q/q},$$

$$(32) \qquad A_{MS}(n) = \frac{T_{MS}}{e} A'_{UC}(n),$$

$$(33) \qquad A_{DF}(n) = \frac{T_{DF}}{r} A'_{UC}(n),$$

ou

$$(34) \qquad D(n) = A'_{UC}(n) + A_{DP}(n) + A'_{UC}(n) \{\frac{\sigma_r}{r} + \frac{\sigma_e}{e} + \frac{T_{MS}}{e_1}\},$$

où $e_1 = e(M)$ représente la durée de vie moyenne d'un programme s'executant seul sur le système. Soit s le temps moyen de service ininterrompu à l'UC , alors

$$(35) \qquad s = (1+\sigma_c/c+\sigma_e/e+\sigma_r/r+\sigma_q/q)(1/c + 1/r + 1/e + 1/q).$$

La "dilatation" s'exprime donc comme une somme des taux d'utilisation de l'UC et de DF à laquelle s'ajoute un terme correctif exprimant l'influence des overheads et de la virtualisation de la mémoire (possibilité d'exécuter des programmes plus grands que la mémoire physique). Cette somme de taux d'utilisation constitue une mesure du parallélisme de fonctionnement entre les unités du système

Remarquons que les relations que nous avons écrites (équations (31), (32), (33), (34)) ont été obtenues en l'absence de toute hypothèse sur les distributions des intervalles de temps entre défauts de page et entre entrées-sorties, et sur les distributions de temps de service à la mémoire secondaire et au disque de fichier. Dans toutes ces relations, seuls les rapports simples T_{MS}/e, T_{DF}/r et σ_c/c, σ_e/e, σ_r/r et σ_q/q interviennent.

Chacune de ces quantités exprime le rapport d'un paramètre de comportement de programme, par exemple le temps moyen r entre deux entrées-sorties consécutives, et d'un paramètre de l'architecture associé à ce comportement, par exemple le temps moyen de service T_{DF} du disque de fichier, ou le temps propre σ_r provoqué par le lancement d'une entrée-sortie.

En raison de leur généralité, ces relations ont un intérêt important pour l'évaluation des performances du système étudié. Ainsi, le taux de gestion propre σ, défini comme le rapport entre l'activité de l'UC en gestion propre et l'activité de l'UC consacrée aux programmes, s'ecrit, à partir de (31).

$$\sigma = \frac{\sigma_c}{c} + \frac{\sigma_e}{e} + \frac{\sigma_r}{r} + \frac{\sigma_q}{q} \tag{36}$$

D'autre part, également à partir de l'équation (31), on peut déduire que, puisque on a toujours,

$$A_{UC}(n) \leqslant 1,$$

l'activité de l'UC consacrée aux programmes est limitée par les overheads comme ci-dessous :

$$A'_{UC}(n) \leqslant \frac{1}{1 + \frac{\sigma_c}{c} + \frac{\sigma_e}{e} + \frac{\sigma_r}{r} + \frac{\sigma_q}{q}} \tag{37}$$

L'effet des temps propres a également une conséquence directe sur la dilatation D(n). A partir de (34) et (37), il vient :

$$D(n) \leqslant \frac{1 + \frac{T_{DF}}{r} + \frac{\sigma_r}{r} + \frac{T_{MS}}{e_1} + \frac{\sigma_e}{e_1}}{1 + \frac{\sigma_c}{c} + \frac{\sigma_e}{e} + \frac{\sigma_r}{r} + \frac{\sigma_q}{q}} . \tag{38}$$

L'équation (38) permet d'écrire la condition à partir de laquelle la dilatation devient supérieur à 1, pour n donné, D(n) > 1 si :

$$(39) \qquad \frac{T_{DF}}{r} > \frac{\sigma_c}{c} + \frac{\sigma_e}{e} + \frac{\sigma_q}{q} - \frac{T_{MS}}{e_1} - \frac{\sigma_e}{e_1} \text{ , avec } e = e(\frac{M}{r})$$

Il convient de noter que la condition (39) est une condition nécessaire, mais pas suffisante pour assurer une dilatation supérieure à 1.

Les relations (31), (32), (33) et (34) ont également une application directe dans l'évaluation de systèmes par simulation ou mesures. En effet, elles permettent de réduire le nombre de grandeurs à mesurer du fait des liaisons simples qui existent entre elles, et de faciliter l'interprétation des résultats en fournissant un modèle de référence.

Toutefois, afin de poursuivre l'exploitation du modèle, il reste à calculer explicitement les quantités inconnues et principalement, en la circonstance, $A_{UC}(n)$ en fonction des différents paramètres du modèle.

3.2.2. - Résolution du modèle

Les techniques de calcul de la solution stationnaire d'un réseau de files d'attente [19,21] où les files d'attente sont gérées suivant la politique PAPS (Premier Arrivé, Premier Servi) ne sont applicables que si les distributions de temps de service aux différents serveurs sont des lois exponentielles négatives.

Si pour ces raisons nous supposons que $F_c(t)$, $F_r(t)$, $F_e(t)$, $F_{MS}(t)$ et $F_{DF}(t)$ sont des distributions exponentielles négatives et que $\sigma_c = \sigma_e = \sigma_r = \sigma_q = 0$ et $q = \infty$ (pas de quantum), on peut alors montrer que le temps de service ininterrompu à l'UC est distribué également suivant une loi exponentielle de moyenne s donnée par l'équation (35), et que le réseau de files d'attente est un réseau de JACKSON [21]. Lorsque les temps propres sont pris en compte, le temps de service ininterrompu à l'UC n'est plus distribué suivant une loi exponentielle, et les résultats de JACKSON ne

s'appliquent pas. Dans ce cas nous calculerons une solution approchée en supposant que ces temps suivent une loi exponentielle de moyenne s donnée par l'équation (35), et les résultats obtenus seront validés par la simulation.

Dans ces deux hypothèses, la résolution se ramène au calcul de $A_{UC}(n)$ puisque toutes les autres quantités cherchées s'en déduisent. La méthode de calcul dûe à BUZEN [11]consiste à exprimer la quantité $A_{UC}(n)$ sous la forme :

$$A_{UC}(n) = \frac{G(n-1)}{G(n)} , \qquad (40)$$

Les G(n) étant calculés par récurrence. Notons que dans le calcul de G(n) n'interviennent également que les rapports simples entre paramètres de l'architecture et paramètres de comportement correspondants définis dans le paragraphe précédent.

Dans le cas où les hypothèses de JACKSON ne sont pas satisfaites, la valeur des résultats obtenus par la résolution analytique a été évaluée par comparaison avec les résultats obtenus par la simulation du même modèle.

3.2.3. - Validation de la résolution analytique

Les expériences de validation de la méthode de résolution analytique portent sur deux points. Tout d'abord étudier les conséquences sur les résultats des hypothèses de distribution exponentielle faites sur les temps de service de la mémoire secondaire et du disque de fichier. Ensuite, vérifier que l'approximation faite pour le calcul de la solution dans le cas où les temps propres ne sont pas nuls est satisfaisante.

Sauf précisions contraires, les expériences ont été conduites avec le choix de paramètres suivant : M=128 pages, r=20, q=∞, c=800, T_{MS}=5, T_{DF}=30, $e(m)=\alpha m^k$, avec α=0.01, k=1.5, l'unité étant la milliseconde. Les simulations écrites en langage SIMULA ont été conduites pendant un temps simulé de 45 secondes correspondant à un temps d'exploitation sur CII-IRIS 80 de 5 à 7 minutes. La durée de 45 secondes a été fixée afin d'atteindre une précision de l'ordre de 5% sur la valeur des quantités mesurées $A_{UC}(n)$ et $A'_{UC}(n)$ et permettre ainsi une comparaison valable avec les résultats fournis par la résolution analytique du modèle. Il convient de noter que cette méthode demande moins de 0.001 minute sur CII-IRIS 80 pour le calcul de $A_{UC}(n)$ et $A'_{UC}(n)$.

Les résultats de validation sont rassemblés sur les figures 18, 19 et 20. Les premières expériences portent sur l'hypothèse de distribution de service à la mémoire secondaire et au disque de fichier et les figures 18a, 18b montrent que même lorsque les distributions ne sont pas exponentielles, mais constantes comme il est plus réaliste de le supposer, en particulier pour la mémoire secondaire, l'approximation réalisée par l'hypothèse exponentielle est satisfaisante, et conduit à une erreur relative le plus souvent inférieure à 5% et qui ne dépasse pas 12%. La seconde série de figures 19a, 19b indique l'erreur obtenue lorsque les intervalles de temps entre défauts de page sont supposés suivre une distribution exponentielle, alors que nous l'avons vu dans le paragraphe 2.2.1, la distribution réelle est mieux approchée par une distribution hyperexponentielle. Il est intéressant de noter que l'erreur obtenue est sensiblement plus petite dans le cas où les temps de service à la MS et au DF sont constants que dans le cas où seuls les temps de service MS sont constants. La raison est que comme le montrent les résultats du chapitre les erreurs dues à une distribution constante et à une distribution hyperexponentielle sont de signes contraires, et tendent donc à s'annuler lorsqu'elles interviennent ensemble.

Lorsque les overheads sont pris en compte, la résolution analytique n'est qu'approchée puisque les hypothèses exponentielles sur les intervalles de temps entre évènements ne sont plus satisfaites. La comparaison des résultats calculés avec ceux fournis par la simulation est présentée sur les figures 20a, 20b. Les conclusions sont semblables à celles obtenues plus haut, et l'approximation donnée par la résolution analytique peut être considérée comme très satisfaisante.

n	1	2	3	4	5	6	7	8	9	10
Résultat analytique	.35	.43	.41	.34	.25	.20	.16	.13	.11	.09
Simulation	.33	.43	.46	.34	.25	.18	.16	.13	.11	.09
Erreur Relative	.06	-	.12	-	-	.10	-	-	-	-

$A_{UC}(n)$

Résultats analytiques : temps de service exponentiels à la MS et au DF

Résultats de simulation : temps de service constants à la MS exponentiels au DF

$\sigma_q = \sigma_c = \sigma_e = \sigma_r = 0$

(-signifie que l'erreur est inférieure à 5 %)

Figure 18a

n	1	2	3	4	5	6	7	8	9	10
Résultat analytique	.35	.43	.41	.34	.25	.20	.16	.13	.11	.09
Simulation	.35	.45	.44	.35	.25	.20	.15	.13	.11	.09
Erreur relative	-	-	.08	-	-	-	.06	-	-	-

$A_{UC}(n)$

Résultats analytiques : temps de service exponetiels à la MS et au DF

Résultats de simulation : temps de service constants à la MS et au DF

$\sigma_q = \sigma_c = \sigma_e = \sigma_r = 0$

(-signifie que l'erreur est inférieure à 5 %)

n	1	2	3	4	5	6	7	8	9	10
Résultat analytique	.35	.43	.41	.34	.25	.20	.16	.13	.11	.09
Simulation	.34	.37	.37	.31	.24	.18	.15	.12	.10	.09
Erreur relative	-	.16	.11	.10	-	.11	.07	.08	.10	-

$A_{UC}(n)$ { Résultats analytiques { temps de service MS et DF exponentiels ; durée de vie exponentielle
Résultats simulation { temps de service MS constant ; temps de service DF exponentiel ; durée de vie hyperexponentielle

Figure 19a

n	1	2	3	4	5	6	7	8	9	10
Résultat analytique	.35	.43	.41	.34	.25	.20	.16	.13	.11	.09
Simulation	.33	.41	.39	.31	.23	.19	.15	.12	.11	.09
Erreur relative	.06	.05	.05	.10	.09	.05	.07	.08	-	-

$A_{UC}(n)$ { Résultats analytiques { temps de service MS et DF exponentiels ; durée de vie exponentielle
Résultats simulation { temps de service MS et DF constants ; durée de vie hyperexponentielle

Figure 19b

n	1	2	3	4	5	6	7	8	9	10
Résultat analytique	.40	.56	.64	.65	.63	.60	.57	.54	.52	.50
Simulation	.41	.56	.67	.69	.65	.61	.56	.54	.52	.50
Erreur relative	-	-	-	.06	-	-	-	-	-	-

$A_{UC}(n)$: comparaison des résultats analytiques et de simulation

$\sigma_q = \sigma_c = 1$ ms, $\sigma_e = \sigma_r = 2$ ms.

(-signifie que l'erreur est inférieure à 5 %)

Figure 20a

n	1	2	3	4	5	6	7	8	9	10
Résultat analytique	.32	.37	.35	.29	.24	.19	.15	.13	.11	.09
Simulation	.33	.38	.37	.31	.24	.19	.15	.13	.11	.09
Erreur relative	-	-	-	.06	-	-	-	-	-	-

$A'_{UC}(n)$: comparaison des résultats analytiques et de simulation

$\sigma_q = \sigma_c = 1$ ms, $\sigma_e = \sigma_r = 2$ ms

(-signifie que l'erreur est inférieure à 5 %)

Figure 20b

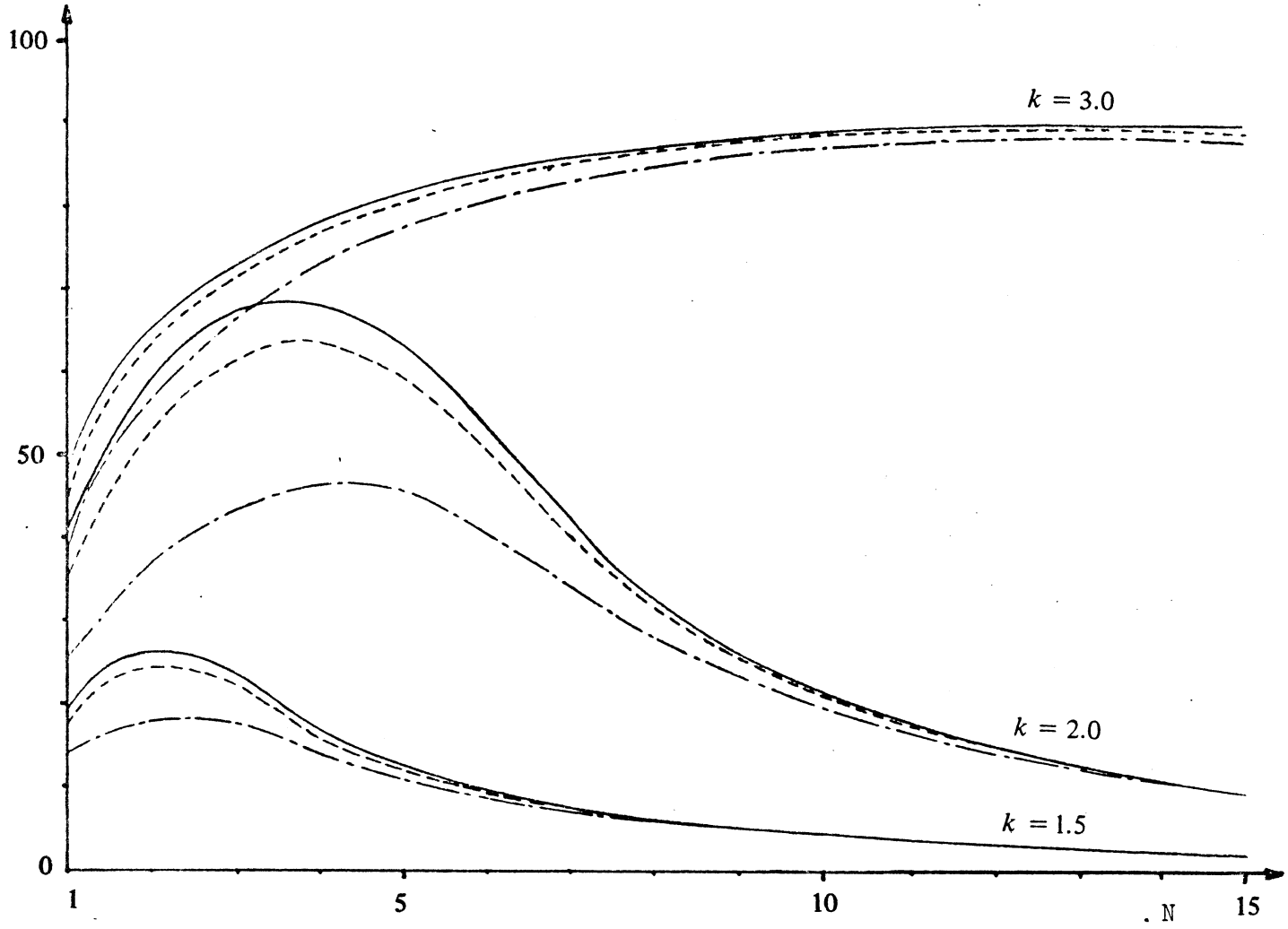


Figure 21 : $A_{UC}(n)$ pour les modèles FDVI et FDVR

——— T=5000 msec
----- T=1000 msec
—·—·— T= 250 msec

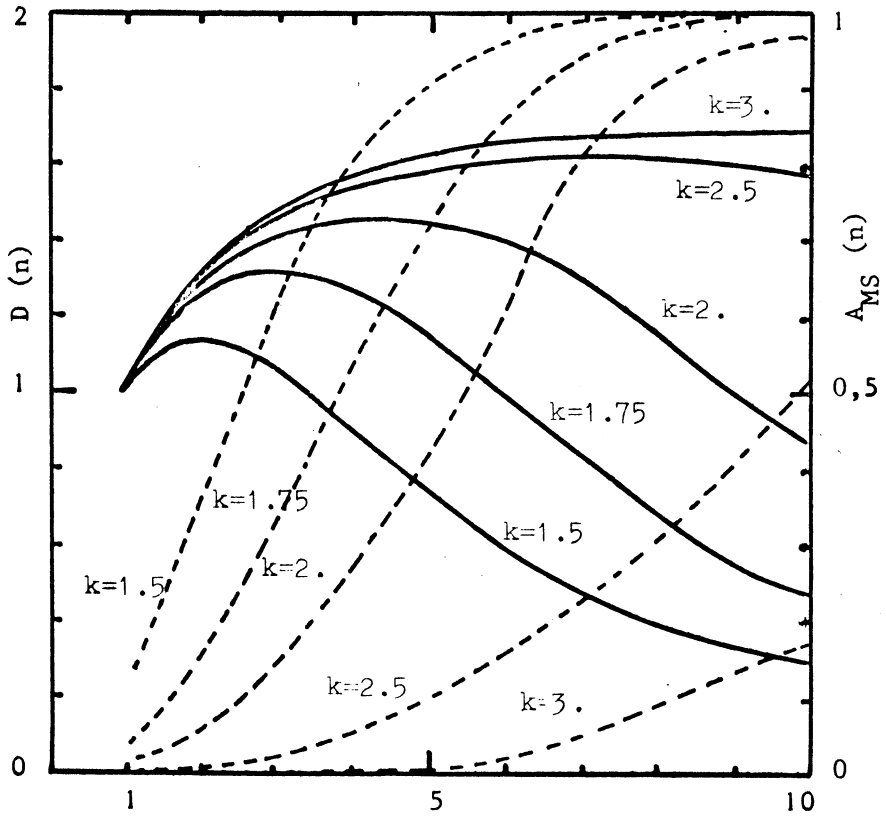


FIGURE 22

D (n) et A_{MS} (n) en fonction de n
(k = 1.5, 1.75, 2, 2.5, 3)

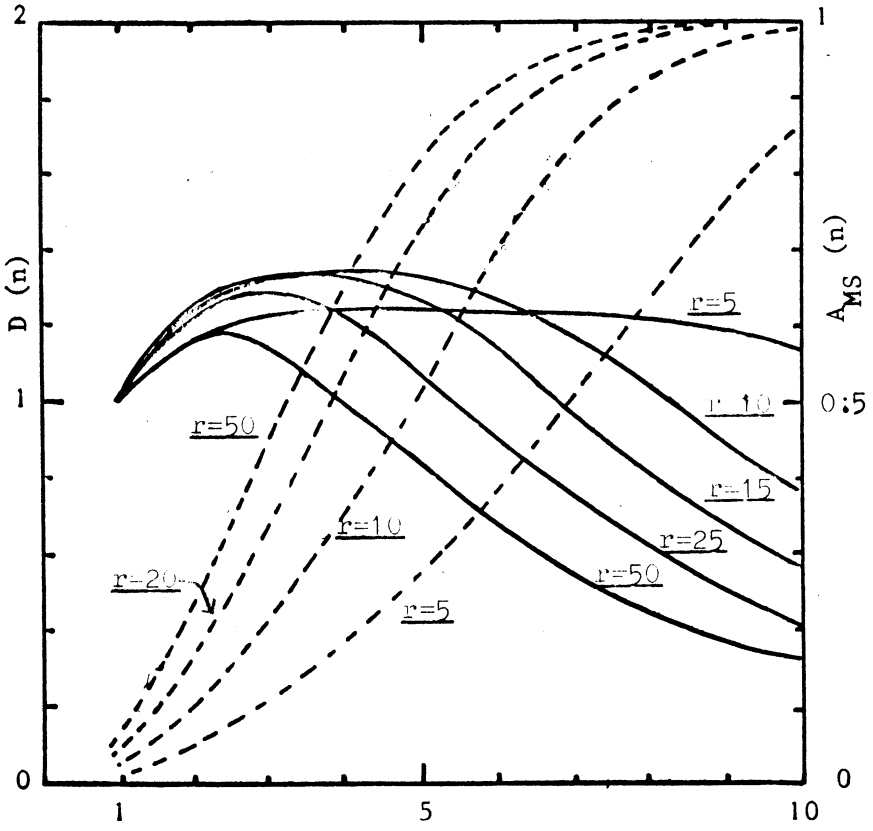


FIGURE 23

D (n) et A_{MS} (n) en fonction de n
(r = 5, 10, 15, 25, 50 (millisecondes))

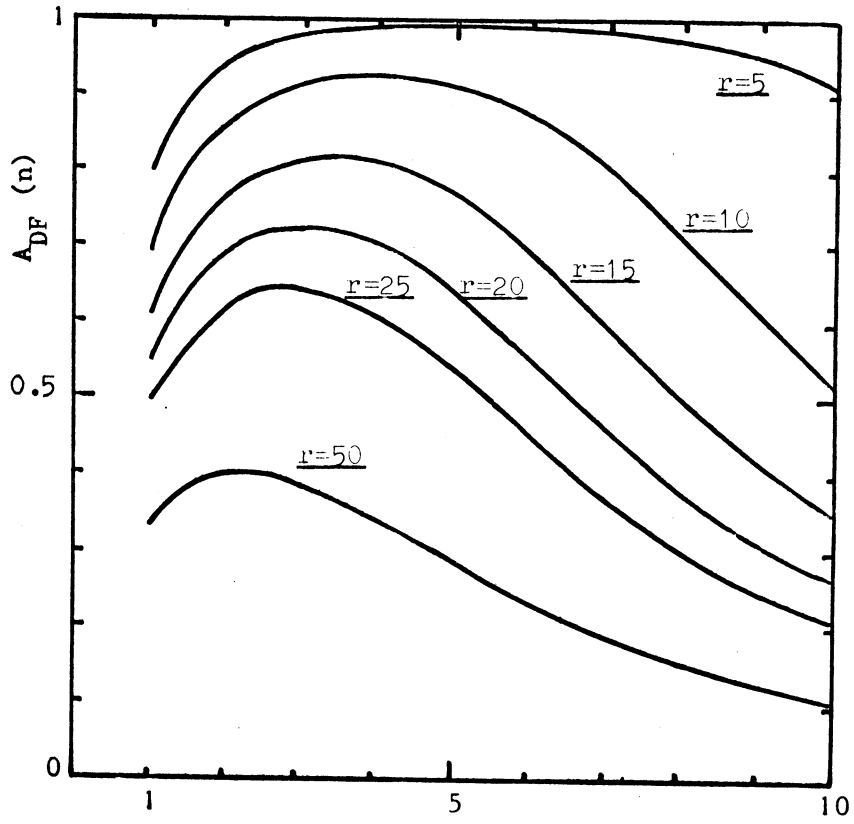


FIGURE 24

A_{DF} (n) en fonction de n
(r = 5, 10, 15, 20, 25, 50 (millisecondes))

3.3. - Influence du comportement des programmes sur les performances globales.

A l'aide du modèle défini précédemment, et en utilisant les résultats fournis par sa résolution analytique, nous pouvons étudier les performances d'un système multiprogrammé à mémoire virtuelle pour différents types de comportements de programme. Les performances du système sont simplement illustrées par la fonction D(n) qui représente la dilatation du temps réel apportée par la multiprogrammation en fonction du degré de multiprogrammation. Les paramètres de comportement de programmes qui interviennent dans les performances sont la fonction de durée de vie e(m) et le temps moyen entre deux entrées-sorties r.

La figure 21 illustre l'écart sur la mesure du taux d'utilisation $A_{UC}(n)$ qui est obtenu suivant que la FDVI ou la FDVR est utilisée. L'expérience a été faite en utilisant les résultats du paragraphe 2.2.2. en calculant $A_{UC}(n)$ d'abord avec la FDVI suivant le modèle $e = m^k$, ensuite vec la FDVR obtenue à partir de ce modèle pour différentes valeurs de T. Les résultats indiquent que l'utilisation de la FDVI tend à surestimer de façon importante les performances globales, particulièrement lorsque le degré de multiprogrammation est inférieur ou égal au degré optimal.

La figure 22 présente l'influence du paramètre k du modèle BK sur les performances globales, pour k variant entre 1.5 et 3. Les courbes montrent de façon très sensible le rôle déterminant de k, ou des propriétés de localité des programmes, sur les performances à la fois par la dilatation maximum qui peut être obtenue, et par le degré optimal de multiprogrammation atteint avant l'écroulement. Le rôle du paramètre k apparaît également directement si l'on considère le taux d'utilisation de la mémoire secondaire qui augmente d'autant plus vite avec n que k est petit.

L'influence du taux d'entrée-sortie, représenté par le paramètre r, temps moyen virtuel entre deux entrées-sorties consécutives d'un programme, est rapportée sur la figure 23 . De toute évidence, la multiprogrammation étant fondée sur la simultanéité entre le traitement des entrées-sorties par le DF et le déroulement des programmes sur l'UC, l'efficacité ne peut être supérieure à 1 que si le taux d'entrée-sortie est suffisant. C'est ce qui

apparaît sur les courbes de la figure 23 tracées avec k = 1.75. Lorsque r diminue, l'efficacité maximum augmente jusqu'à ce que le DF soit saturé comme le montrent les courbes $A_{DF}(n)$ en fonction de n de la figure 24 . Les variations correspondantes de $A_{MS}(n)$ en fonction de n sont représentées sur la figure 23 .

3.4. - Dispersion du critère D(n)

Les résultats présentés dans le paragraphe précédent concernent le critère D(n) défini comme la dilatation moyenne pour le degré de multiprogrammation n. La dispersion de la dilatation autour de sa valeur moyenne est également une grandeur importante pour juger de l'intérêt du critère D(n). D'après l'équation (34), le critère D(n) peut être simplement obtenu à partir de l'estimation de $A'_{UC}(n)$ et $A_{DF}(n)$. Pour simplifier la suite de l'exposé plaçons nous à n constant, en omettant de noter l'indice n dans les quantités considérées.

Soit $W'_{UC}(t)$ et $W_{DF}(t)$ les temps pendant lesquels l'UC et le DF ont été actifs sur un intervalle de temps $[o,t]$ (par UC active nous entendons une activité consacrée aux utilisateurs). Un estimateur simple de A'_{UC} et A_{DF} à l'instant t est alors donné par :

$$A'_{UC}(t) = W'_{UC}(t)/t \quad , \tag{41}$$

$$A_{DF}(t) = W_{DF}(t)/t, \tag{42}$$

et d'après les théorèmes de CHANG et LAVENBERG [12], $A'_{UC}(t)$ et $A_{DF}(t)$ sont des estimateurs non biaisés de A'_{UC} et A_{DF} pour t grand, c'est-à-dire :

$$\lim_{t\to\infty} E\left[W'_{UC}(t)/t\right] = A'_{UC}$$

$$\lim_{t\to\infty} E\left[W_{DF}(t)/t\right] = A_{DF}$$

Afin de pouvoir utiliser pratiquement les estimateurs ainsi définis, il est nécessaire de pouvoir connaître leur variance, ce qui permettra de calculer un intervalle de confiance. Cette question a été étudiée par LAVENBERG et SCHEDLER [24] dans le cas d'un réseau de files d'attente fermé à deux stations, et nous utilisons ici leurs conclusions pour apprécier la qualité de l'estimateur de D(n), c'est-à-dire :

$$\left[W'_{UC}(t) + W_{DF}(t) \right] / t.$$

Notons 1 et 2 les deux stations du réseau étudié par LAVENBERG et SCHEDLER. Les temps de service à la station 1 sont exponentiels de moyenne $1/\mu_1$ et distribués suivant une loi générale de moyenne $1/\mu_2$ et de variance $\mathrm{Var}[T]$ à la station 2. Soit $W_1(t)$ et $W_2(t)$ les durées d'activité des stations 1 et 2 sur un intervalle $[o,t]$. On montre simplement que $W_i(t)$, $i=1,2$ est un processus stochastique cumulatif et que l'on peut écrire alors :

$$\mathrm{Var}\left[W_i(t)/t\right] \sim \sigma_i^2/t,$$

où $\sim$ désigne une égalité asymptotique pour t grand. De plus $\left[W_i(t) - tA_i(t)\right]/\sigma_i\, t^{1/2}$ est asymptotiquement normalement distribué avec une moyenne nulle et une variance égale à 1. On en déduit pour t suffisamment grand un intervalle de confiance sur $A_i(t) = \lim_{t\to\infty} W_i(t)/t$ de la forme :

$$\left[A_i(t) - \varepsilon\sigma_i/\sqrt{t},\ A_i(t) + \varepsilon\sigma_i/\sqrt{t}\right] \qquad i=1,2,$$

où ε définit de façon classique la confiance souhaitée.

Si nous considérons à présent le processus $W_1(t) + W_2(t)$, on montre également qu'il s'agit d'un processus cumulatif et l'on a :

$$\mathrm{Var}\left[W_i(t) + W_2(t)\right] = \mathrm{Var}\left[W_1(t)\right] + \mathrm{Var}\left[W_2(t)\right] + 2\,\mathrm{Cov}\left[W_1(t),\ W_2(t)\right],$$

avec :

$$\text{Cov}\left[W_1(t),\ W_2(t)\right] = \sigma_{12}/t.$$

Les quantités σ_1, σ_2 et $\rho = \sigma_{12}/\sigma_1\sigma_2$ ont été calculées explicitement par SCHEDLER et LAVENBERG et plusieurs exemples numériques ont été traités que nous rapportons sur la figure 25 . Le temps de service $1/\mu_1$ est égal à 1 et les calculs sont faits pour différentes valeurs de $1/\mu_1$, Var[T] et N, nombre de clients dans le réseau.

Les résultats mettent en évidence la corrélation fortement négative entre $A_1(t)$ et $A_2(t)$, et indiquent donc que la qualité de l'estimateur de $A_1 + A_2$ est meilleure que celle de A_1 ou de A_2. Cette conclusion est utilisée par LAVENBERG et SCHEDLER pour proposer des estimateurs de A_1 et A_2 de la forme :

$$A_1(t,\beta) = \beta\ W_1(t)/t + (1-\beta)\ \frac{\mu_2}{\mu_1}\ W_2(t)/t,$$

$$A_2(t,\beta) = \mu_1/\mu_2\ A_1(t,\beta),$$

qui sont meilleurs que $W_1(t)/t$ et $W_2(t)/t$.

Ces observations faites sur un modèle plus simple que celui que nous considérons, mais de même nature, suggèrent que $\left[\ W'_{UC}(t) + W_{DF}(t)\right]/t$ est un bon estimateur de D(n), pour une valeur donnée de n, en raison même de la corrélation négative existant entre $W'_{UC}(t)$ et $W_{DF}(t)$. Si nous revenons aux exemples numériques tirés de [24], nous pouvons calculer et comparer les rapports σ_1/A_1, σ_2/A_2 et $\sigma/(A_1+A_2)$ où σ désigne l'écart type sur $W_1(t) + W_2(t)$ pour t grand, avec :

$$\sigma^2 = \sigma_1^2 + \sigma_2^2 + 2\sigma_1\sigma_2\ \rho$$

Les résultats sont rassemblés sur la figure 26 pour $\mu_1 = \mu_2 = 1$, N = 4 et Var[T] prenant les valeurs 0, 1, 4 et 25, et montrent bien que la dispersion de $A_1 + A_2$ est très sensiblement inférieure à celle de A, ou de A_2.

		Var [T] = 0					Var [T] = 1				
μ_1	N	A_1	A_2	σ_1	σ_2	ρ	A_1	A_2	σ_1	σ_2	ρ
.5	2	.468	.937	.398	.228	-.697	.429	.857	.507	.358	-.771
	4	.499	.998	.490	.055	-.311	.484	.968	.628	.222	-.555
	6	.500	1.000	.499	.011	-.109	.496	.992	.678	.120	-.401
1.	2	.731	.731	.423	.484	-.774	.667	.667	.609	.609	-.800
	4	.870	.870	.501	.537	-.615	.800	.800	.693	.693	-.667
	6	.914	.914	.527	.551	-.571	.857	.857	.728	.728	-.615
2.	2	.904	.452	.288	.558	-.741	.857	.429	.507	.716	-.771
	4	.992	.496	.105	.681	-.413	.968	.484	.314	.888	-.555
	6	.999	.500	.031	.704	-.199	.992	.496	.170	.959	-.401

		Var [T] = 0					Var [T] = 0				
μ_1	N	A_1	A_2	σ_1	σ_2	ρ	A_1	A_2	σ_1	σ_2	ρ
.5	2	.409	.818	.786	.493	-.887	.402	.803	1.82	.959	-.972
	4	.463	.925	.934	.376	-.708	.448	.896	2.13	.640	-.881
	6	.483	.965	1.01	.289	-.590	.467	.934	2.26	.516	-.786
1.	2	.625	.625	.991	.818	-.908	.605	.605	2.34	1.64	-.982
	4	.725	.725	1.11	.896	-.794	.668	.668	2.64	1.57	-.949
	6	.776	.776	1.16	.962	-.730	.692	.692	2.71	1.60	-.927
2.	2	.811	.405	.940	.988	-.895	.778	.389	2.37	2.01	-.984
	4	.907	.454	.827	1.17	-.759	.827	.414	2.43	2.10	-.962
	6	.949	.474	.677	1.29	-.663	.848	.424	2.36	2.19	-.941

Var(T)	0	1	4	25
σ_1/A_1	.575	.866	.1531	3.952
σ_2/A_2	.617	.866	1.235	2.350
$\sigma/(A_1+A_2)$	.262	.353	.465	.937

Figure 26.

3.5. - Comportement de la mémoire secondaire

Les courbes présentées précédemment mettent bien en évidence le rôle centrale de la mémoire secondaire pour les performances globales, et l'écroulement du système apparaît lié directement à la surchage de la mémoire secondaire. On peut donc se demander s'il n'est pas possible d'utiliser la charge de la mémoire secondaire comme un indicateur du fonctionnement satisfaisant de l'ensemble du système.

Cette idée n'est pas neuve, et c'est une opinion généralement partagée par les informaticiens familiers des systèmes multiprogrammés à mémoire virtuelle que les performances optimales sont atteintes lorsqu'il n'y a pratiquement pas de file d'attente derrière la mémoire secondaire. Ainsi, RODRIGUEZ-ROSELL remarque dans [36] que "les mesures effectuées sur les files d'attente des canaux de pagination et d'entrée-sortie indiquent qu'avant l'écroulement des performances, il n'y a pas de formation de files d'attente derrière ces canaux", et ADAMS [1] a observé que le système EMAS fonctionne au maximum de son efficacité lorsque l'occupation du canal de pagination est voisine de 50%. Ces résultats peuvent être vérifiés à partir de mesures faites sur le système IBM-CP67 [36] et les prédictions de performances réalisées pour le système MULTIC S [39]. Les résultats sont rassemblés sur les figures 27 et montrent également que l'occupation du canal de pagination est proche de 50% lorsque l'utilisation de l'unité centrale est maximum.

Nous pouvons essayer de répéter ces observations à l'aide du modèle. Dans ce but trois séries d'expériences ont été menées et les résultats sont présentés sur les figures 28 . Les expériences consistent à calculer $A'_{UC}(n)$ et $A_{MS}(n)$ pour différentes valeurs des paramètres les plus importants du modèle

n	1	2	3	4	5	6
Efficacité du système	.17	.28	.31	.29	.21	.16
Taux de défaut de page	.18	.33	.49	.61	.75	.85

Figure 27a : utilisation de la mémoire secondaire sur IBM - CP 67

(le temps moyen de service d'une requête de page est égal à 1.7 fois le temps moyen de service de la mémoire secondaire pour prendre en compte le déchargement des pages écrites, soit T_{MS} = 1.7 × 12 = 20.4 m sec.)

n	1	2	3	4	5	6	7	8
A'_{UC}	.55	.64	.66	.65	.64	.63	.61	.60
e (m sec.)	98	56	42	35	30	28	26	24
$A'_{UC/e}$	.56	1.1	1.6	1.9	2.1	2.3	2.4	2.5
$A_{MS}=A'_{UC}\frac{e}{T_{MS}}$	.20	.40	.55	.66	.73	.79	.83	.87

Figure 27b : utilisation de la mémoire secondaire sur MULTICS

le temps moyen de service de la mémoire secondaire T_{MS}, le temps moyen entre entrée-sortie r et le coefficient k du modèle de durée de vie, et à tracer sur un même graphe $A'_{UC}(n)$ et $A_{MS}(n)$ afin que le comportement de la mémoire secondaire puisse être simplement relié à $A'_{UC}(n)$ et donc à D(n).

Plusieurs remarques peuvent être faites sur les figures 28 . La première est que toutes les courbes $A_{MS}(n)$ ont une forme en S semblable, avec un point d'inflexion proche de la valeur $A_{MS}(n) = 0.5$ qui correspond sensiblement à la valeur maximum de $A'_{UC}(n)$. Afin de vérifier cette observation, les parties de courbes $A_{UC}(n)$ et $A_{MS}(n)$ correspondant aux valeurs de n pour lesquelles $0.4 \leqslant A_{MS} \leqslant 0.6$ ont été soulignées, et nous pouvons remarquer que les parties soulignées contiennent, pour toutes les expériences, l'extremum de $A'_{UC}(n)$. Nous en déduisons que la zone de fonctionnement optimal du système peut être simplement caractérisée par la charge de la mémoire secondaire. La seconde remarque que nous pouvons faire est importante par rapport à cette première conclusion : le fait que le point d'inflexion de $A_{MS}(n)$ soit situé dans le voisinage du point de fonctionnement optimal indique que la sensibilité $\Delta n/\Delta A_{MS}(n)$ est minimum dans cette région. La conséquence pratique de ce résultat est que le degré de multiprogrammation optimum peut être estimé de façon satisfaisante à partir de $A_{MS}(n)$.

La dernière remarque est que les observations précédentes ont été vérifiées pour un large ensemble de valeurs des paramètres de l'architecture et du comportement des programmes, qu'elles ne dépendent donc pas d'une combinaison particulière de ces paramètres, et que les conclusions que nous avons pu en tirer sont en bon accord avec celles dégagées par l'étude expérimentale. Nous pouvons en déduire qu'il s'agit là d'un aspect fondamental du comportement des systèmes à mémoire virtuelle qui peut se résumer par les deux principes suivants :

Principe I :

Dans un ordinateur multiprogrammé à mémoire virtuelle, les performances optimales sont atteintes lorsque l'utilisation de la mémoire secondaire est proche de 0.5.

courbe n°	1	2	3	4	5
r (msec.)	10	20	30	40	50

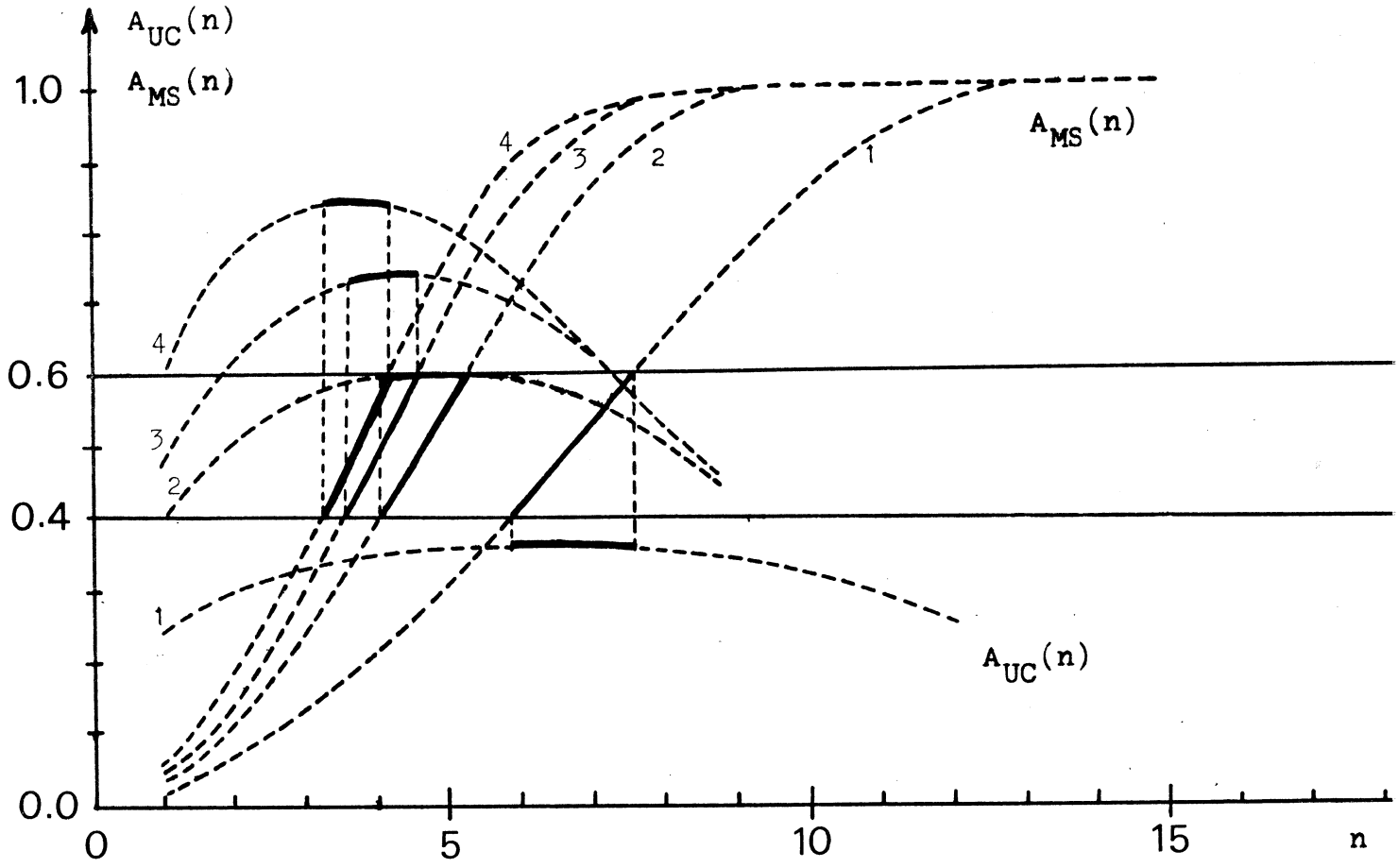


FIGURE 28a

M = 512, k = 1.5, α = .01 msec., T_{DF} = 30 msec., T_{MS} = 10msec

courbe n°	1	2	3	4	5
r (msec.)	10	20	30	40	50

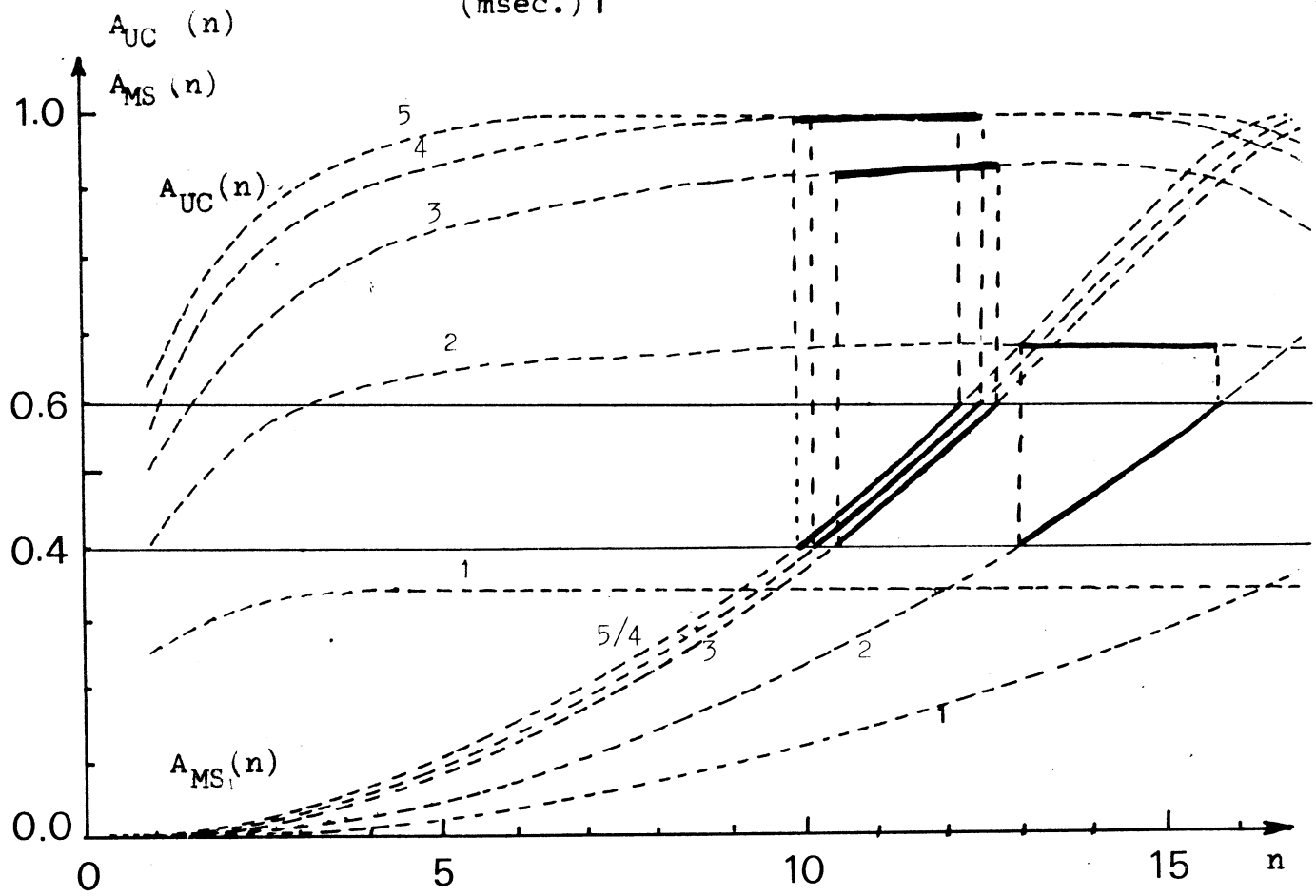


FIGURE 28b

M = 512, k = 2, α = .01 msec., T_{DF} = 30 msec.

T_{MS} = 10 msec.

courbe n°	1	2	3	4	5
T_{MS} (msec.)	5	10	20	30	100

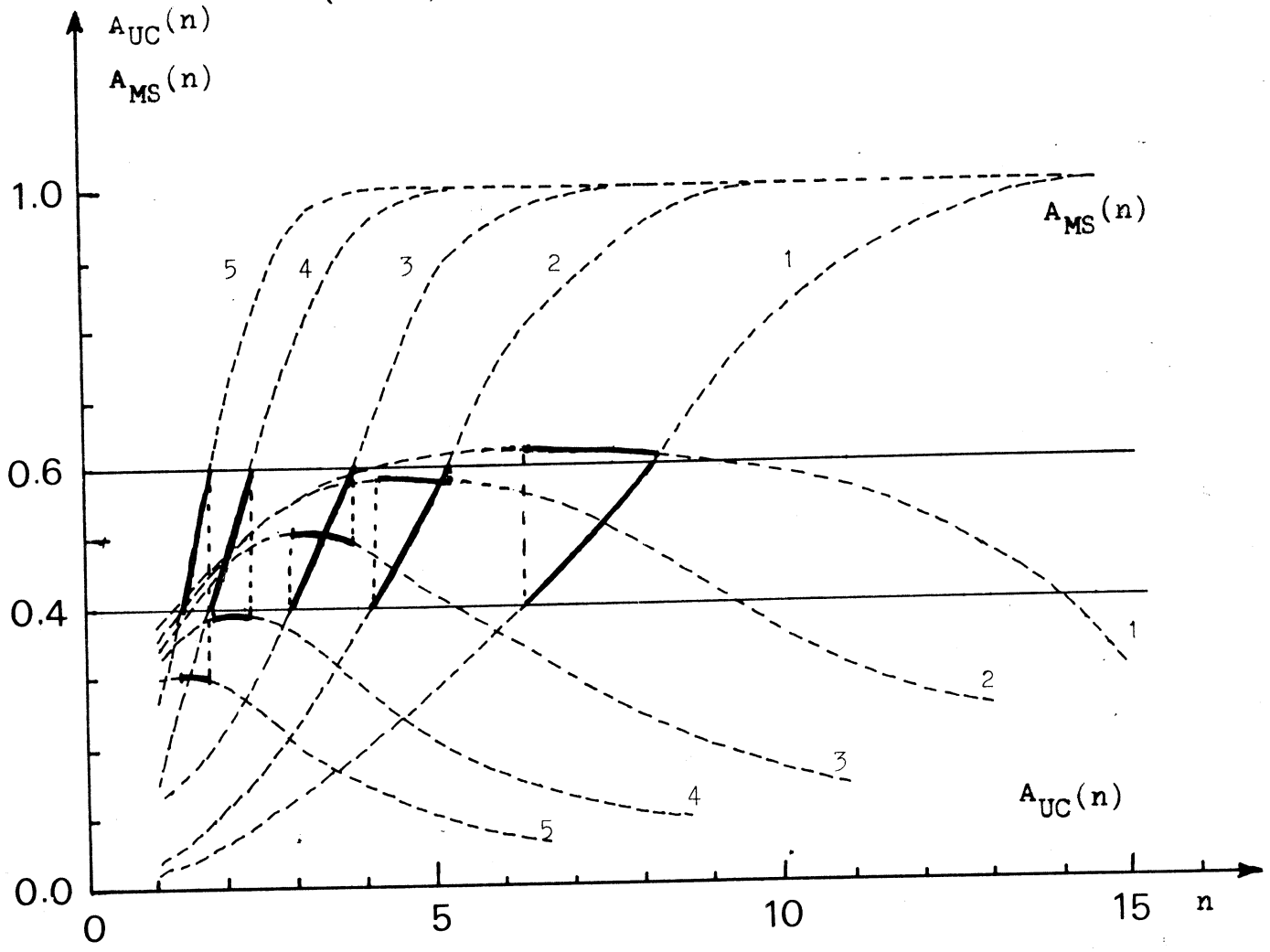


FIGURE 28c

M = 512, k = 1.5, α = .01 msec., r = 30 msec., T_{DF} = 30 msec

θ_c = 0.5 msec.
θ_r = 2.0 msec.
θ_e = 1.0 msec.

courbe n°	1	2	3	4	5
T_{MS} (msec.)	5	10	20	30	100

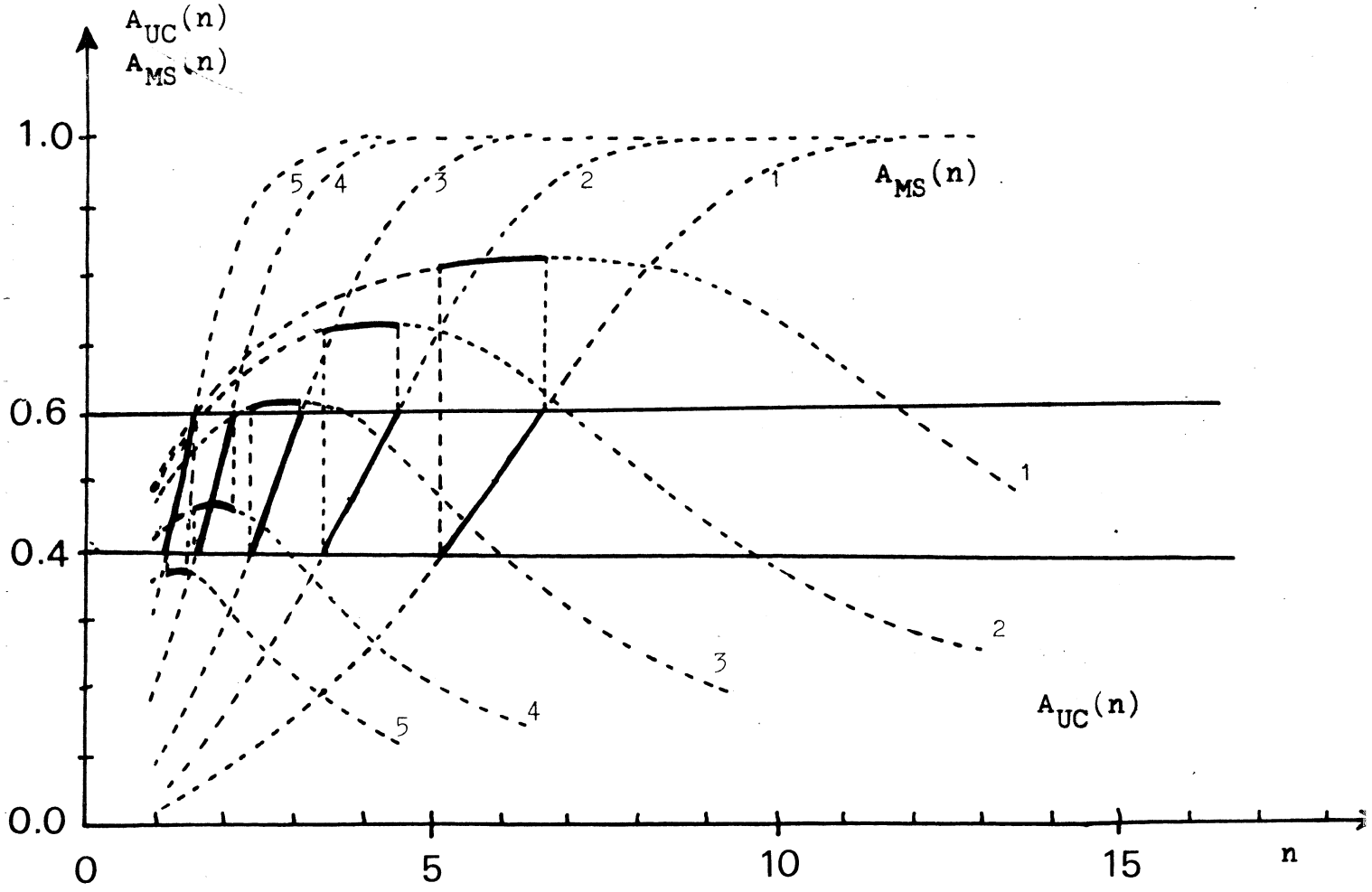


FIGURE 28d

M =512, k = 1.5, = .01 msec., r = 30 msec., T_{DF} = 30 msec.

Principe II

Au voisinage du point de fonctionnement optimum, la sensibilité du degré de multiprogrammation a l'utilisation de la mémoire secondaire est minimum.

4. - REGLES ET CRITERES DE CONTROLE DU DEGRE DE MULTIPROGRAMMATION

4.1. - Introduction

Le contrôle du degré de multiprogrammation dans les systèmes multiprogrammés à mémoire virtuelle peut se concevoir selon deux approches différentes. La première approche consiste à utiliser un des modèles de comportement de programme analysés plus haut pour prédire les besoins en mémoire du ou des programmes au cours de leur exécution. Il s'agit alors d'identifier un ou plusieurs paramètres de ce modèle de comportement et d'en déduire, par un algorithme approprié, le contrôle requis. Nous qualifierons de contrôle par comportement de programme cette méthode qui a l'intérêt d'opérer un contrôle à partir d'informations prises à la source même des perturbations -c'est-à-dire le comportement des programmes- qui affectent les performances du système et peuvent entraîner son écroulement. En revanche, rien ne garantit que le contrôle ainsi exercé optimise les performances, et il s'agit plutôt alors d'un contrôle préventif qui vise à limiter le degré de multiprogrammation et à éviter ainsi l'écroulement. Dans ce sens nous parlerons de contrôle sous-optimal, et c'est l'expérimentation qui permettra d'apprécier la valeur de ces principes de contrôle.

Dans cette catégorie de contrôles fondés sur cette approche entrent le principe de l'ensemble de travail [14] construit à partir du modèle de l'ensemble de travail d'un programme, et les règles dites du "genou" et "L=S"[16] dérivées du modèle de durée de vie. Ces trois principes ou règles de contrôle du degré de multiprogrammation sont dues à DENNING.

La seconde approche consiste à utiliser une des mesures de performance du système pour estimer le degré optimal de multiprogrammation et, à partir de cette estimation, à opérer une régulation afin de maintenir le degré réel proche du degré optimal estimé. Dans ce cas, le contrôle n'est pas déterminé à partir du comportement des programmes, mais indirectement à partir des conséquences de leur comportement sur les performances globales : nous parlerons alors de contrôle par comportement de système. Cette approche repose sur l'hypothèse que les variations de comportement des programmes se traduisent de façon aisément identifiables sur les performances globales du système, et que l'on peut mettre en évidence l'existence d'un degré optimal de multiprogrammation. Cette hypothèse a été largement vérifiée par l'analyse du modèle d'architecture à mémoire virtuelle présentée dans le paragraphe précédent, et des éléments supplémentaires de réponse seront fournis plus loin. L'intérêt de cette méthode est de fournir un contrôle optimal puisque la régulation est faite dans le but d'optimiser une mesure de performance. Dans cette catégorie de contrôle entrent l'algorithme de contrôle adaptatif fondé sur le maximum de critère de dilatation [2] et une variante de cet algorithme construit à partir des principes I et II énoncés dans le paragraphe [27]. Un premier développement de cette approche a été présenté en [17b].

Quels que soient la catégorie et le principe d'une règle de contrôle, sa valeur dépend finalement de ses propriétés de robustesse, des difficultés que posent sa mise en oeuvre sur un système, et de sa stabilité. Par robustesse, nous entendons la capacité d'une règle de contrôle à réaliser un contrôle satisfaisant quelles que soient les conditions de fonctionnement du système ; par stabilité, sa capacité à opérer une régulation qui ne soit pas source de perturbations pour le système. La robustesse d'une règle est liée à la justesse des observations expérimentales et les analyses qui la justifient, et c'est donc à ce niveau là qu'elle pourra être discutée. Les difficultés d'implémentation et de stabilité dépendent des solutions apportées aux trois types de problèmes posés par la mise en oeuvre d'un algorithme de contrôle sur un système :

- problème d'instrumentation : comme nous l'avons vu,toute règle de contrôle requiert l'identification de paramètres liés soit au comportement des programmes, soit au fonctionnement du système. La mesure de ces paramètres peut demander la mise en place d'outils de mesure internes au système représentant un investissement et une charge supplémentaire pour le système.

- problèmes d'estimation : quels que soient les paramètres mesurés ces paramètres sont en général hautement variables, alors que leur utilisation par l'algorithme de contrôle demande que des valeurs stables soient fournies sous peine d'instabilité. Les grandeurs mesurées doivent donc subir des traitements statistiques pour filtrer ces informations parasites et fournir à l'algorithme de contrôle des grandeurs représentatives du fonctionnement du système.

- problèmes d'algorithmes de contrôle : les règles de contrôle définissent le principe de la régulation à effectuer à partir de l'estimation des paramètres de comportement des programmes ou du système. Il reste en général à préciser l'algorithme, ou l'heuristique, qui réalisera cette règle, en tenant compte des risques d'erreurs sur l'estimation des paramètres, en s'assurant que le contrôle est effectué quelles que soient les conditions de fonctionnement, en particulier au démarrage du système, et en minimisant les instabilités possibles.

Nous présentons dans la suite de ce chapitre les différentes règles de contrôles citées plus haut, leurs propriétés de robustesse et les problèmes posés par leur mise en oeuvre.

4.2. - Contrôle par comportement de programme

Comme nous l'avons vu au début de ce chapitre, le principe du contrôle par comportement de programme est d'utiliser un modèle de comporteme de programme comme estimateur du comportement des programmes et plus précisément de leurs besoins en mémoire. Dans ce paragraphe nous présentons les règles de contrôle qui ont été proposées suivant ce principe, et nous discutons de leurs propriétés en fonction des observations expérimentales faites sur les modèles correspondants.

4.2.1. - Contrôle par le modèle de l'ensemble de travail

Les principes d'allocation de ressources aux programmes dans un système multiprogrammé utilisant la notion d'ensemble de travail ont été proposés et analysés en détail par DENNING dans [14]. En reprenant le schéma classique décrivant la situation d'un programme vis-à-vis des ressources qu'il requiert, les programmes peuvent être partagés en trois classes : actifs, prêts et bloqués, suivant qu'ils ont acquis les ressources UC et mémoire qu'ils demandent, qu'ils sont en attente d'une de ces ressources, ou qu'ils ne requièrent ni l'une ni l'autre de ces ressources.

A partir de cette classification, le principe de l'ensemble de travail s'énonce comme suit :

Un programme ne peut être activé que si son ensemble de travail est chargé en mémoire, et une page de la mémoire ne peut être déchargée si elle appartient à l'ensemble de travail d'un programme actif.

Plusieurs implémentations de ce principe ont été réalisées [32,37], et il a inspiré la conception des mécanismes de gestion de ressources dans de nombreux systèmes [1,6,21,30]. L'étude des différentes réalisations ou tentatives de réalisation montre que la difficulté principale rencontrée porte sur la mesure de l'ensemble de travail des programmes actifs, et que c'est cette difficulté qui a limité dans de nombreux cas la mise en oeuvre complète du principe de l'ensemble de travail.

En effet, en l'absence de dispositifs câblés appropriés, la mesure de l'ensemble de travail prend un temps d'unité centrale important, ce qui tend à limiter la fréquence de mesure [1] ou même à rendre les mesures impraticables [21]. Ainsi dans le système ESOPE l'impossibilité de mesurer la taille de l'ensemble de travail a conduit à une solution intermédiaire où à chaque usager est attribuée une "catégorie", ensemble d'une taille mémoire et d'un temps de résidence en mémoire, la catégorie d'un programme étant modifiée suivant le comportement du programme au cours de son exécution. Dans le système EMAS [1], les mêmes contraintes conduisent au même type de solution, avec cependant un calcul de l'ensemble de travail sur des intervalles de temps assez longs de l'ordre de la seconde permettant d'opérer un pré-chargement de l'ensemble de travail d'un programme avant son activation. Toutefois,

l'importance de la période de mesure conduit à une certaine inertie des réactions du système aux modifications des ensembles de travail par suite d'une prédiction insuffisamment précise du comportement futur du programme. L'implémentation réalisée par RODRIGUEZ et DUPUY [37] satisfait davantage au principe énoncé par DENNING, avec une période de mesure de 30 msec. rendue possible par l'utilisation d'une combinaison d'indicateurs software et hardware de l'utilisation d'une page.

En revanche, pour l'ensemble des implémentations du principe de l'ensemble de travail connues le problème d'estimation ne s'est pas posé, c'est-à-dire que c'est le résultat de la mesure qui est utilisé directement pour prendre une décision sur l'allocation de ressources. Cette observation indique bien la qualité du modèle de l'ensemble de travail comme estimateur des besoins de programme, sans qu'il y ait besoin d'un traitement supplémentaire pour filtrer les mesures. Les observations expérimentales sur la taille de l'ensemble de travail donnent les raisons de cette constatation puisque d'une part la dispersion de la taille est faible et que d'autre part le coefficient d'autocorrélation est élevé, même pour des périodes de mesure importantes. Il est toutefois évident que si l'ensemble de travail peut être utilisé directement avec succès comme l'ont montré les expériences d'implémentation, on peut améliorer la prédiction réalisée avec un estimateur plus complexe. C'est ce qu'à montré BRYAN en proposant différents algorithmes de prédiction de la taille de l'ensemble de travail [9] . Citons.

$$\text{I} \qquad \hat{W}_{i+1} = W_i$$

$$\text{II} \qquad \hat{W}_{i+1} = \mu_i$$

$$\text{III} \qquad \hat{W}_{i+1} = W_i + (W_i - W_{i-1})\rho_{\mu,i}$$

$$\text{IV} \qquad \hat{W}_{i+1} = \mu_i + (W_i - \mu_i)\rho_{w,i}$$

où $\hat{W}_{i+1}$ est la taille estimée pour la période i+1 à partir de la taille W_i mesurée pendant la période i, μ_i est la moyenne des W_i, $\rho_{\mu,i}$ et $\rho_{w,i}$ sont respectivement les autocorrélations entre $(\Delta W)_i$ et $(\Delta W)_{i+1}$ et W_i et W_{i+1}.

L'algorithme I est évidemment l'estimateur classique utilisant la mesure sans traitement supplémentaire. Les figures 29ab montrent les résultats de prédiction pour les algorithmes II et IV.

Les problèmes de contrôle posés par l'implémentation du principe de l'ensemble de travail sont assez classiques, et principalement liés d'une part aux contraintes imposées par le système à la mise en oeuvre du principe, d'autre part aux décisions d'allocation non résolues par le principe de l'ensemble de travail. L'implémentation présentée dans [37] illustre bien ce point en montrant les choix à faire par un concepteur en fonction du système et des objectifs d'exploitation pour atteindre une allocation équilibrée des ressources aux différents programmes.

4.2.2. - Contrôle par le modèle de durée de vie

L'étude du modèle de durée de vie met en évidence, comme nous l'avons vu, un certain nombre de propriétés de comportement des programmes et la forme et les paramètres de la fonction de durée de vie d'un programme peuvent fournir une indication sur les besoins en mémoire de ce programme. Il est donc tentant d'utiliser ce modèle pour aider à l'allocation de ressources, suivant l'approche utilisée pour l'ensemble de travail. Deux propositions ont été faites dans ce sens par DENNING [16] sous les noms de "Règle du Genou" et "Règle L=S". Ces deux règles ont été analysées par DENNING d'un point de vue statique, mais, à notre connaissance, aucune implémentation n'en a été réalisée. Nous présentons dans ce paragraphe la règle du genou telle qu'elle est exposée dans [16] pour examiner ensuite les problèmes posés par son implémentation à partir des données expérimentales sur le modèle de durée de vie présentées dans le paragraphe 2.2.

La règle du genou

L'observation expérimentale des fonctions de durée de vie montre que la forme de ces fonctions est habituellement constituée d'une partie approximativement convexe suivie d'une région approximativement concave, comme cela apparaît sur la figure 5 du paragraphe 2.2.1. Notons cependant que fréquemment plusieurs de ces régions convexes et concaves peuvent être mises en évidence.

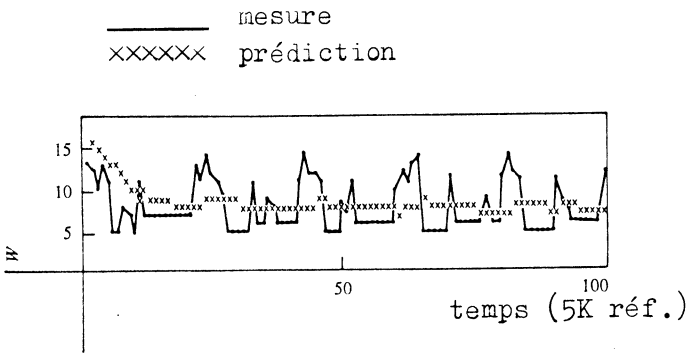


Figure 29a : algorithme II (T=5K)

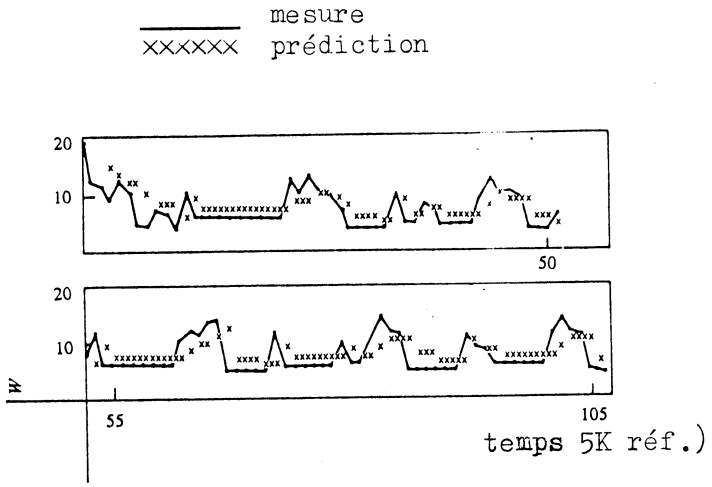


Figure 29b : algorithme IV (T=5K)

Le genou de la fonction est défini comme le point au-delà duquel la courbe tend à s'aplatir. Plus rigoureusement, on peut le définir comme le point de contact le plus élevé entre un rayon issu de l'origine et la courbe. L'effet d'écrasement au-delà du genou résulte de deux facteurs :

a) la gestion de mémoire tend à **maintenir** en mémoire les localités des programmes, si bien que la majorité des défauts de page se produisent durant les transitions du programme entre les localités, et ceci relativement indépendamment du nombre de pages allouées ;

b) l'effet de facteurs externes comme la réduction du temps moyen de résidence en mémoire des programmes à cause des entrées-sorties, et le chargement initial des pages comme analysé dans le paragraphe 2.2.2. et dans [33].

DENNING montre dans[16] qu'une propriété importante du genou est de maximiser l'intégrale espace-temps du nombre de pages utilisé par un programme par référence, et que cette propriété est une condition nécessaire et suffisante de maximisation du débit du système. A partir de cette observation, la règle du genou se déduit et consiste à réaliser l'allocation de mémoire à chaque programme de telle sorte que le nombre de pages chargées soit voisin du genou de leur fonction de durée de vie. La règle est validé d'un point de vue statique à l'aide d'un modèle analytique de système multiprogrammé semblable à celui présenté dans le paragraphe 3.

La mise en oeuvre de la règle du genou demande la mesure pour chaque programme de sa fonction de durée de vie, et ensuite l'estimation, à partir de ces mesures du genou de la courbe. En supposant que les problèmes de mesure ont été résolus, la difficulté principale nous paraît résider dans l'estimation du genou. Les données expérimentales montrent combien les intervalles de temps entre défauts de page sont dispersés pour une allocation de page donnée, et donc la nécessité de disposer d'un estimateur pour construire la courbe de durée de vie à partir des mesures, estimateur dont la variance risque d'être élevée en raison de la dispersion des mesures. Le calcul du genou apparaît donc délicat puisqu'il s'agit de détecter sur une courbe estimée avec une certaine imprécision, un changement de pente. L'expérience acquise sur le calcul du maximum d'une fonction estimée à partir de mesures dans le cas du système de contrôle décrit dans le paragraphe suivant montre qu'il est illusoire d'espérer calculer de façon satisfaisante le genou d'une courbe de

Plus généralement, nous pouvons penser, au vu des données sur les séries temporelles d'intervalles de temps entre défauts de page, que toute stratégie d'allocation fondée sur le modèle de durée de vie pose des difficultés d'implémentation importantes liées à la trop grande dispersion de la variable considérée. La raison en est que ces intervalles de temps entre deux défauts de page d'un programme représentent des informations extrêmement ponctuelles du comportement d'un programme, alors qu'en revanche, par exemple, l'ensemble de travail par sa définition même opère une certaine agrégation des phénomènes de comportement et possède donc de meilleures qualités statistiques et de prédiction.

4.3. - Contrôle par comportement de système

L'analyse du modèle de système multiprogrammé à mémoire virtuelle présenté dans le paragraphe 3 a mis en évidence l'existence d'un degré optimal de multiprogrammation, que nous noterons par la suite n_o, vis-à-vis du critère de dilatation D(n) :

$$D(n_o) \leqslant D(n) \quad , \quad n = 1, 2, \ldots$$

De plus nous avons montré que la valeur de n_o est liée aux propriétés de comportement de programme, c'est-à-dire qu'à chaque type de comportement correspond une valeur de n_o distincte. C'est sur ces deux remarques qu'est construit le contrôle du degré de multiprogrammation suivant le critère de dilatation maximum. Ce contrôle est organisé en trois fonctions, suivant le schéma général que nous avons proposé :

- prise de mesures sur le système,
- estimation de n_o,
- régulation de n suivant n_o.

L'estimation de n_o peut être faite de deux façons différentes. La première méthode consiste à partir de la définition de n_o, c'est-à-dire à identifier l'extremum de la fonction D(n). La seconde méthode est d'utiliser les résultats du paragraphe 3.5 sur le comportement de la mémoire secondaire, c'est-à-dire à mesurer l'activité de la mémoire secondaire et à définir n_o comme la valeur du degré de multiprogrammation pour laquelle cette activité atteint un seuil donné S voisin de 50%.

Nous présentons dans les pages suivantes les résultats d'une expérience de réalisation des différentes fonctions d'instrumentation,d'estimation et de contrôle pour les deux méthodes qui viennent d'être définies. Ces fonctions ont été implémentées dans un modèle de simulation d'un système multiprogrammé à mémoire virtuelle, et l'amélioration des performances apportée par le système de contrôle a été évaluée [2]. Nous décrivons tout d'abord le système de contrôle tel qu'il est conçu dans le cas où l'estimation de n_o est faite en identifiant l'extremum de D(n), et nous montrerons ensuite comme le système est modifié lorsque l'estimation de n_o est faite à partir de la règle des 50%.

4.3.1. - Estimation de n_o par l'extremum de D(n)

Le système de contrôle, ou controleur, est représenté sur la figure 30 . Il consiste en un dispositif de prise de mesures MES inséré dans le système ; un estimateur EST chargé d'estimer le critère D(n) à partir des mesures prises par MES ; un optimiseur OPT qui calcule, à partir de l'estimation de D(n), le degré de multiprogrammation optimale n_o ;et enfin un interrupteur K qui contrôle l'introduction de nouveaux programmes dans le système en fonction de la valeur courante n du degré de multiprogrammation et de n_o. Les hypothèses prises pour la conception du contrôleur sont les suivantes :

1) Le degré de multiprogrammation n est le seul paramètre sur lequel le contrôleur peut agir. Il s'agit d'une hypothèse simplificatrice destinée à limiter l'étude à la régulation du degré de multiprogrammation sans se préoccuper de faire un choix entre les programmes en attente dans la file Q. Ce dernier aspect est abordé par LEROUDIER [29]

2) Le critère pris en compte a un seul extremum. Cette hypothèse a été vérifiée par l'analyse du modèle.

3) Les variations des propriétés de comportement des programmes sont lentes vis à vis des variations internes au système. Autrement dit, le temps moyen mis par le système pour atteindre un état d'équilibre est petit devant l'intervalle de temps moyen correspondant à une variation significative du comportement des programme. Cette hypothèse a été vérifiée

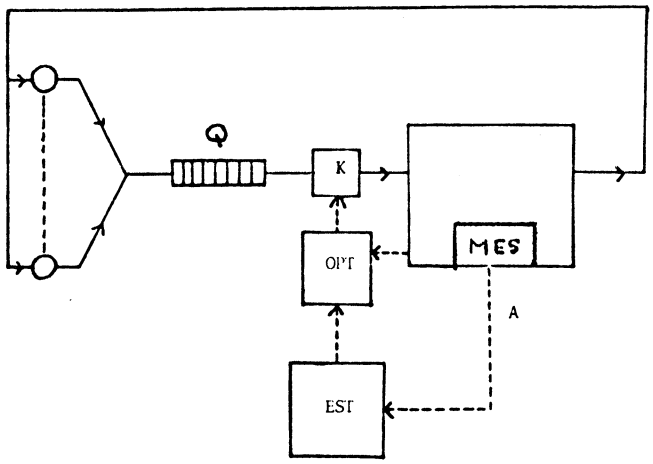


Figure 30

par l'analyse du régime transitoire du système. Les variations de comportement des programmes sont détectées par EST et transmises à OPT pour le calcul du degré optimal n_o. Les fonctions EST et OPT sont construites selon les principes généraux suivants :

1- Le degré de multiprogrammation n est maintenu à la valeur n_o. Le calcul de n_o est fait suivant une heuristique utilisant l'hypothèse 2. Cette heuristique prend en compte le fait que dans certains cas le critère maximum est atteint pour plusieurs valeurs consécutives de n, bien que ces valeurs de n ne soient pas équivalentes en ce qui concerne le temps de réponse : en raison des temps propre, le temps de réponse est en effet une fonction croissante de n.

2- Les mesures prises par MES sont pondérées par EST de façon à accorder un poids plus important aux événements récents qu'aux événements passés.

3- Les différentes fonctions du contrôleur sont activées à chacun des événements suivants : arrivée d'un programme dans la file d'attente q ; sortie d'un programme du système. Toutefois, pour assurer que des mesures sont prisesen toutes circonstances, l'activation de MES et EST est forcée périodiquement par une horloge de garde.

4- Le système de contrôle ne prend en compte que l'introduction des programmes dans le système et non leur sortie du système. La sortie d'un programme se produit donc indépendamment de toute action du contrôleur.

a) Le critère

Le critère de dilatation D(n) a été défini dans le paragraphe 3 comme le rapport du débit du système lorsque le degré de multiprogrammation est n au rapport de ce débit en monoprogrammation. Nous avons montré que D(n) pouvait s'exprimer, d'après 1'équation (34), sous la forme suivante :

$$D(n) = A'_{UC}(n) + A_{DF}(n) + A'_{UC}(n) \left\{ \frac{\theta_r}{r} + \frac{\theta_e}{e} + \frac{T_{MS}}{e_1} \right\}$$

L'équation (34) montre que D(n) s'exprime comme la somme de l'utilisation de l'UC consacrée aux programmes $A'_{UC}(n)$ et de l'utilisation du disque de fichier $A_{DF}(n)$, à laquelle s'ajoute un terme supplémentaire qui peut être négligé en première approximation. Nous en déduisons que, pratiquement, l'estimation de D(n) peut se faire à partir de mesures sur $A'_{UC}(n)$ et $A_{DF}(n)$, quantités facilement accessibles sur un système.

b) Réalisation des fonctions du contrôleur

Les fonctions du contrôleur ont été implémentés dans un modèle de simulation de système multiprogrammé à mémoire virtuelle, et les différents choix qui ont guidé leurs réalisations proviennent, à la fois de considérations théoriques simples et des résultats de différentes expériences de mise au point en simulation. L'objectif final étant d'obtenir un contrôleur aussi simple que possible et possédant de bonnes propriétés de stabilité.

Le principe de réalisation est le suivant : à chaque instant la valeur optimale estimée de n, notée $\bar{n}_o$ est connue et mémorisée. Cette variable ne peut être modifiée que par OPT. Lorsque une arrivée à la file d'attente Q, ou le départ d'un programme du système se produit, c'est-à-dire à chacun des événements définis plus haut, K est activé. Les mesures de $A'_{UC}(n)$ et $A_{DF}(n)$ sont prises après chacun de ces événements, ou sur le signal de l'horloge de garde de façon à ce que entre deux mesures consécutives n reste fixe. Si la valeur courante de n est inférieure à $\bar{n}_o$, K introduit un programme dans le système à partir de la file Q, dans le cas où celle-ci n'est pas vide, sinon aucun nouveau programme n'est introduit dans le système ; K appelle ensuite EST, puis est mis en attente. Nous renvoyons pour la description des fonctions EST et OPT à [2,29] où les mécanismes sont présentés en détail et justifiés.

4.3.2. - Estimation de n_o par la charge de la mémoire secondaire

La structure du contrôleur reste semblable à celle qui vient d'être décrite, et seuls sont modifiés les quantités mesurées et l'algorithme d'estimation de n_o.

La fonction MES prend des mesures sur l'activité de la mémoire secondaire, et ces mesures subissent dans EST un traitement identique à celui opéré dans le cas précédent afin d'obtenir un estimateur lissé et débiaisé de $A_{MS}(n)$ ainsi que la variance de cet estimateur. OPT met à jour l'estimation de n_o, $\bar{n}_o$ à partir de l'estimateur de $A_{MS}(n)$ calculé par EST. OPT utilise un seuil S qui est un paramètre du contrôleur choisi proche de 50 % d'après la règle des 50 %. Lorsque OPT est activé, il calcule $\bar{n}_o$ suivant l'algorithme suivant, où $[Z^+, Z^-]$ désigne l'intervalle de confiance construit autour de l'estimateur de $A_{MS}(n)$:

si $S \in [Z^+, Z^-]$ alors $\bar{n}_o \leftarrow \bar{n}_o$,

si $S > Z^+$ alors $\bar{n}_o \leftarrow \bar{n}_o + 1$,

si $S < Z^-$ alors $\bar{n}_o \leftarrow \bar{n}_o - 1$.

4.3.3. - Résultats numériques

Le fonctionnement et les performances des systèmes de contrôle par comportement de système ont été faites par simulation et les résultats de ces expériences sont rapportés dans [2] . Le simulateur a une structure semblable au modèle décrit dans le paragraphe 3 et c'est également les mêmes modèles qui sont utilisés pour caractériser le comportement des programmes. Les différentes fonctions du contrôleur sont également simulées en faisant intervenir la charge supplémentaire qu'elles imposent sur le système.

Pour toutes les expériences présentées, les paramètres suivants ont été utilisés :

- N = 20 et le temps de réflexion aux télétypes est distribué suivant une loi exponentielle de moyenne d = 8 secondes,

- temps U.C. total distribué suivant une loi exponentielle de moyenne c = 800 msec.,

- temps entre défauts de page et entrée-sortie distribués suivant une loi exponentielle,

- la fonction de durée de vie utilisée est :

$$e = \alpha(M/n)^k$$

avec :

$\alpha = 0.01$ msec^{-1} , M = 512 pages,

- le temps de service de la mémoire secondaire est distribué suivant une loi constante de moyenne T_{MS} = 5 msec.,

- le temps de service au disque de fichier est distribué suivant une loi exponentielle de moyenne T_{DF} = 30 msec.,

- les temps propres UC ont des valeurs constantes égales à θ_r = 2 msec., θ_e = 1 msec., θ_c = 0.5 msec., et chaque activation de EST, OPT, ou K, engendre un temps propre constant de 0.5 msec.

Afin d'étudier les performances du système de contrôle, deux charges différentes ont été considérées. La charge CH1 est définie par :

$$\text{CH1} \begin{cases} k = 1.5 \ , \\ r = 30 \text{ msec.}, \end{cases}$$

et la charge CH2 par :

$$\text{CH2} \begin{cases} k = 2, \\ r = 20 \text{ msec.} \end{cases}$$

Les figures 31 illustrent le comportement de chacune de ces charges vis à vis du critère D(n) tel qu'il est estimé par EST : la charge CH1 du fait de ses caractéristiques provoque l'écroulement des performances si le degré n n'est pas contrôlé, alors que le système peut supporter un nombre élevé de programmes de la charge CH2. L'indication de l'écart-type obtenu sur l'estimateur de D(n) montre également que cet écart est faible et qu'une bonne précision sur D(n) est fournie par l'estimateur.

Deux familles d'expériences ont été menées. Les premières consistent à simuler le fonctionnement du système avec chacune des charges CH1 et CH2 avec et sans le contrôleur. Les résultats sont rassemblés sur la figure 32 et montrent comment le mécanisme de contrôle permet une utilisation des ressources meilleure et plus équilibrée . Dans ce tableau l'efficacité E est définie par le rapport :

$$E = \frac{\Sigma \text{ temps de service}}{\Sigma \text{ Temps de service} + \Sigma \text{ temps d'attente}}$$

Pour la charge CH1, les overheads sont réduits de 24 % à 12 %, et le temps moyen de réponse et l'efficacité sont améliorés de 40 % et 80 %. La mise en oeuvre du contrôleur n'affecte pas les performances du système lorsqu'il exécute CH2, et réduit sensiblement le degré de multiprogram-

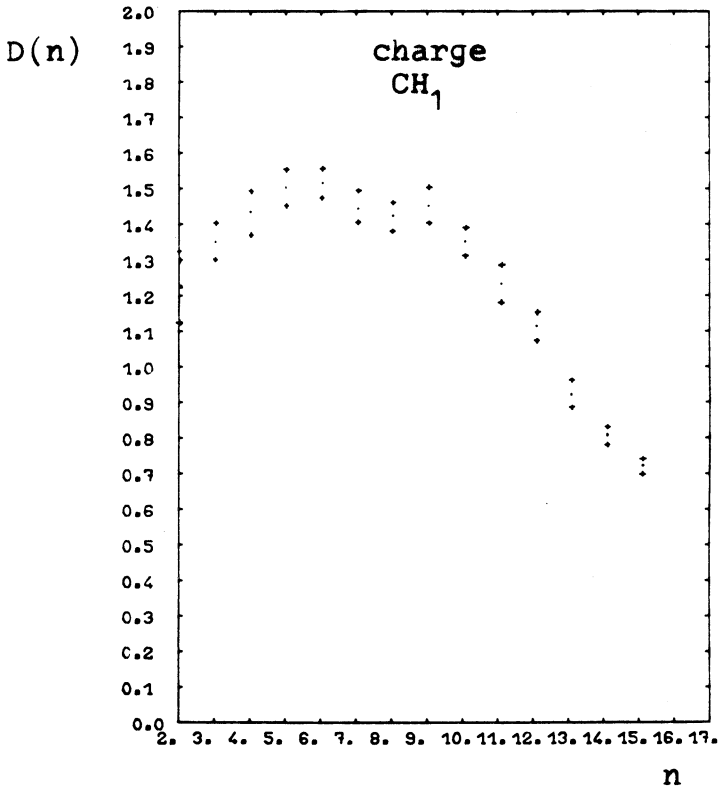


FIGURE 31a

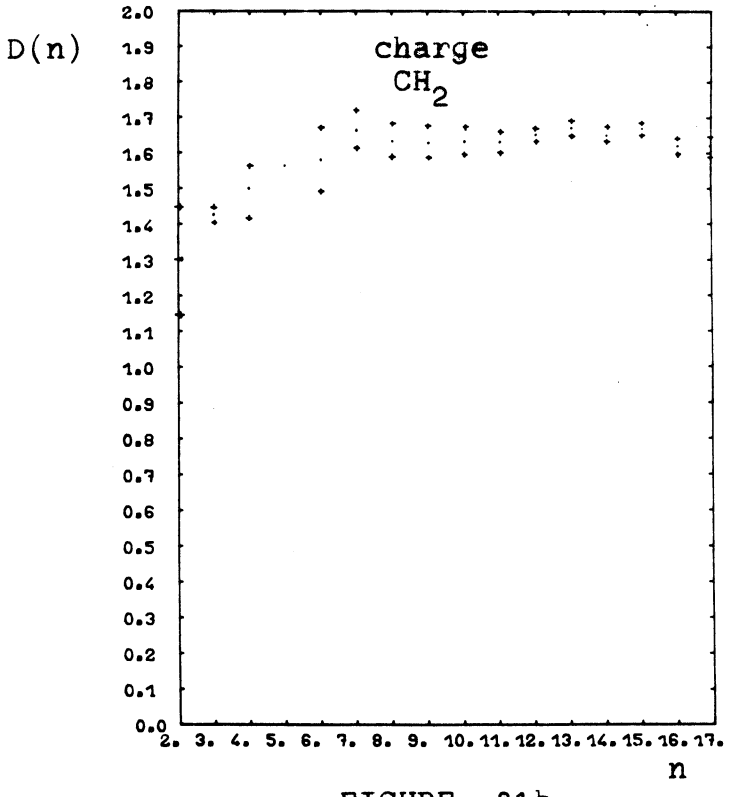


FIGURE 31b

Mesures de performance	CH_1		CH_2	
	sans régulation	avec régulation	sans régulation	avec régulation
A_{UC}	.59	.72	.78	.73
A'_{UC}	.35	.60	.66	.65
A_{MS}	.97	.29	.24	.11
A_{DF}	.53	.90	.99	.97
temps de réponse moyen	33.1	18.8	16.8	16.5
degré moyen	16.1	5.	13.5	9.
efficacité moyenne	.06	.11	.13	.12

Figure 32

	sans régulation	avec régulation
A_{UC}	.65	.89
A'_{UC}	.49	.68
A_{MS}	.54	.45
A_{DF}	.67	.88
temps de réponse moyen	25.3	18.7
degré moyen moyen	14.2	11.1
efficacité moyenne	.09	.19

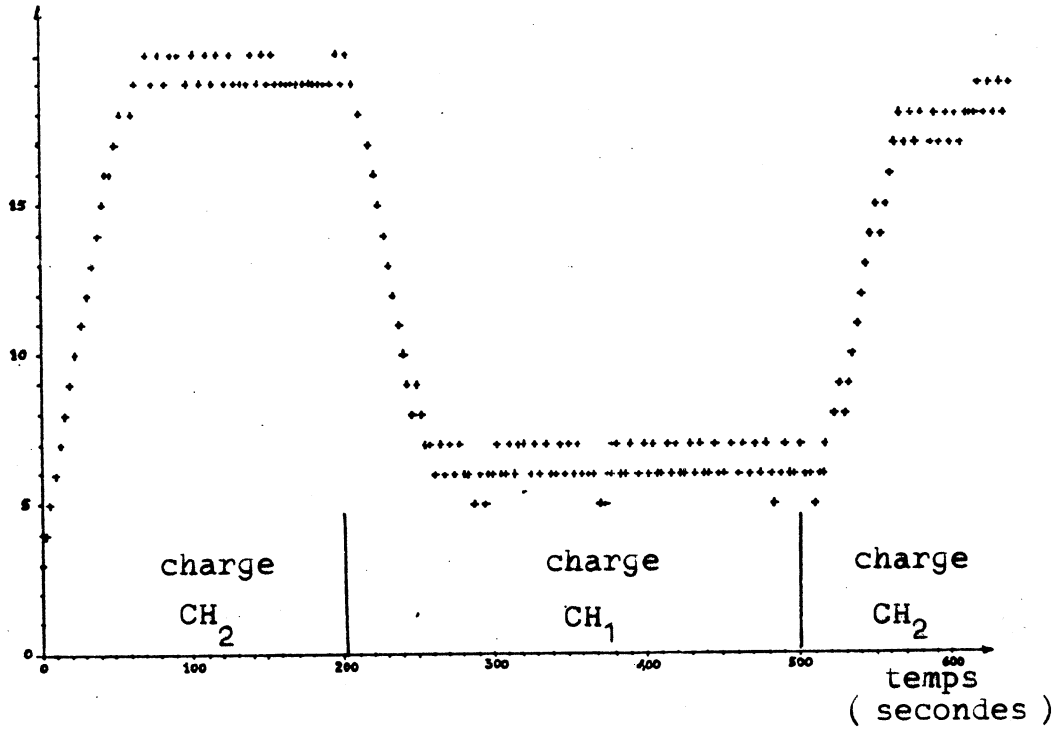


FIGURE 33a

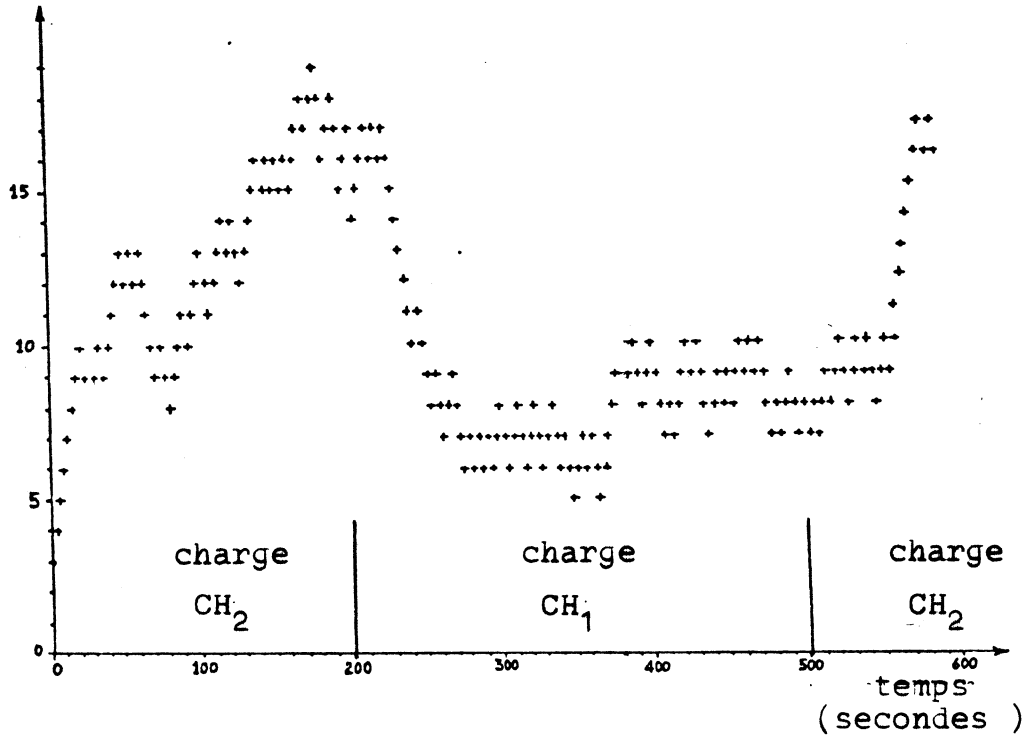


FIGURE 33b

ont été obtenus de la même façon aussi bien pour le contrôle par estimation de l'extremum de D(n) que pour le contrôle par estimation de la charge de la mémoire secondaire.

La seconde série d'expériences consiste à soumettre une charge variable au système et à observer la façon dont le contrôleur identifie les variations de charge et réagit sur le contrôle du degré de multiprogrammation. La charge variable consiste en la charge CH2 pendant 200 sec., CH1 pendant 300 sec., et de nouveau CH2 pendant 200 sec. Les performances du contrôleur sont évaluées en observant la valeur du degré optimum n_o estimée par le contrôleur, et en calculant les mêmes mesures que précédemment. Les résultats des simulations sont présentés sur les figures 33a,33b pour le contrôle suivant D(n) et suivant $A_{MS}(n)$ respectivement. La figure 34 donne les mesures de performances pour la charge variable obtenues identiquement pour les deux contrôles. Nous pouvons observer la différence entre les estimation de n_o suivant le mode de contrôle utilisé : dans le cas du contrôle par estimation de D(n) plusieurs oscillations apparaissent avant que la valeur estimée se stabilise à la valeur correspondant à la charge soumise au système, alors que pour le contrôle suivant $A_{MS}(n)$ la variation de n_o est pratiquement immédiate sans phénomène d'oscillations. La différence entre ces deux comportements est évidemment lié au type de grandeur estimée : dans le premier cas on estime le maximum d'une fonction, c'est-à-dire que l'on estime le point pour lequel la dérivée de cette fonction s'annule, alors que dans le second cas on estime le point pour lequel la fonction elle-même prend une valeur donnée. Ces remarques montrent combien les problèmes d'estimation conditionnent les performances d'un système de contrôle, et indiquent la difficulté qu'il y a à implémenter un principe de contrôle qui n'est pas construit sur une propriété simple d'un des indices de performance du système.

5. CONCLUSION.

Dans les deux chapitres précédents, nous avons étudié un problème d'allocation de ressources et nous avons pu caractériser simplement le comportement des usagers vis à vis de cette ressource : par la loi de distribution des temps de service pour la ressource U.C. ; par le taux d'arrivée des paquets vers une direction donnée pour l'allocation des tampons. C'est à dire que dans les deux cas le comportement des usagers est aisé à identifier puisqu'il s'agit en général d'estimer les paramètres d'une distribution, et les propriétés de ces modèles sont bien connues. Pour le problème de l'allocation de la mémoire qui a été l'objet de chapitre, la caractérisation du comportement des programmes vis à vis de cette ressource n'est plus aussi simple, puisqu'il s'agit de quantifier la répartition des références dans l'espace d'adressage, et l'on est amené à utiliser des modèles qui sont tous des caractérisations indirectes de cette répartition. Deux de ces modèles ont été présentés dans la première partie du chapitre : le modèle de l'ensemble de travail qui caractérise la concentration des références dans le temps en mesurant le sous-ensemble de l'espace d'adressage référencé pendant un nombre de références donné ; le modèle de durée de vie qui caractérise la concentration dans l'espace en mesurant le nombre de références entre deux références consécutives en dehors d'un sous-ensemble donné de l'espace d'adressage. En désignant par $\mathcal{P}(A)$ l'ensemble des parties de A, espace d'adressage du programme considéré, nous pouvons représenter, comme sur la figure 35, le modèle de l'ensemble de travail et le modèle de durée de vie comme des applications de $\mathbb{N}$ dans $\mathcal{P}(A)$ et de $\mathcal{P}(A)$ dans $\mathbb{N}$ respectivement.

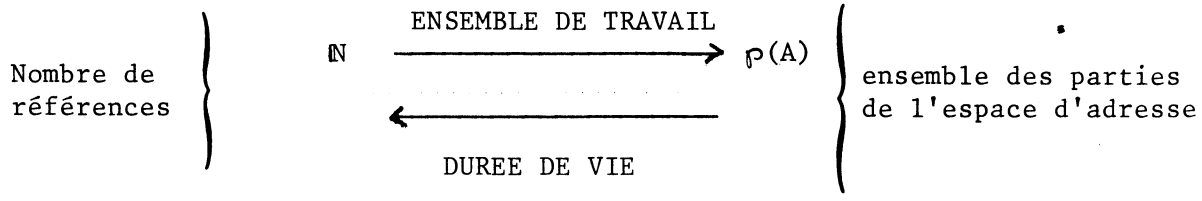


Figure 35.

Le modèle de l'ensemble de travail fournit un moyen de contrôler l'allocation de mémoire aux programmes par le paramètre de taille de la fenêtre de mesure et de prévoir cette allocation en raison des propriétés statistiques remarquables de la taille des ensembles de travail : faible dispersion et forte corrélation entre les tailles de deux ensembles successifs. Le modèle de durée de vie permet de mesurer les performances d'une allocation donnée en reliant le temps de séjour dans la partie de l'espace d'adressage considéré à la taille de cette partie, et les propriétés d'indépendance des durées de vie consécutives justifient l'emploi de ce modèle dans des modèles à réseaux de files d'attente.

C'est ce qui est fait dans la seconde partie du chapitre où le comportement du système considéré est analysé à l'aide d'un modèle à réseau de files d'attente. L'étude de ce type de modèles est assez classique, et les observations les plus importantes nous paraissent liées à la validation du modèle. Les résultats confirment les conclusions obtenues à l'issue du premier chapitre sur la robustesse des modèles à réseau de files d'attente vis à vis essentiellement des hypothèses de distribution et autorisent pratiquement l'emploi généralisé du modèle analytique de préférence au modèle de simulation, au moins pour le niveau de détail concerné. Nous avons également montré que le critère de performance retenu pouvait être estimé par un estimateur de faible variance, ce qui est essentiel lorsque l'estimateur est utilisé dans une boucle de contrôle.

La troisième partie du chapitre consacrée au contrôle du degré de multiprogrammation met de nouveau en évidence l'importance d'une analyse des propriétés statistiques des grandeurs mesurées. Les résultats obtenus sur le modèle de l'ensemble de travail et le modèle de durée de vie montrent le comportement fondamentalement différent de ces deux modèles et excluent à priori toute utilisation du modèle de durée de vie pour le contrôle de l'allocation mémoire ou du degré de multiprogrammation. Dans le cas de contrôle par comportement de système, il est nécessaire de mettre en oeuvre une procédure d'estimation du critère sur lequel est fondé le contrôle. Les observations faites sur les deux schémas de contrôle montrent encore combien les performances finales dépendent de la qualité de l'estimation obtenue, et illustrent le risque qu'il y a à opérer un contrôle à partir d'un critère difficile à estimer.

Nous avons volontairement orienté cette conclusion vers les aspects statistiques évoqués dans ce chapitre, plutôt que sur le problème étudié lui-même. Il nous semble en effet que la statistique peut devenir de plus en plus un outil d'analyse des systèmes informatiques et de récentes journées d'études sur le sujet [35]ont bien mis en évidence la multiplicité des applications de la statistique dans le domaine informatique. Cette évolution est un signe de la vitalité du domaine des mesures et de l'évaluation des performances des systèmes informatiques, et indique la nécessité fondamentale pour les études menées dans cette direction de s'appuyer sur des outils d'analyse puissants.

Annexe A

Modèle A

Lorsqu'un programme est réactivé après une interruption, il ne retrouve qu'une seule de ses pages. Autrement dit, il y a déchargement complet du programme suite à l'interruption. La matrice correspondante α^A_{ij} s'écrit donc :

$$(14) \qquad \alpha^A_{ij} = \begin{cases} 1 & \text{si } j=1 \\ 0 & \text{si } j>1 \end{cases} \qquad i=1,\ldots M$$

Nous en déduisons le diagramme de transition, présenté sur la figure 13 et la matrice P donnée par

$$(15) \qquad P_{ij} = \begin{cases} \Theta_i & , i=1,\ldots M-1,\ j=i+1 \\ 1-\Theta_i & , i=1,\ldots M\ ;\ j=1 \\ \Theta_M & , i=M\ ;\ j=M \\ 0 & \text{sinon} \end{cases}$$

A partir de l'équation (15) nous pouvons écrire

$$\begin{cases} \Pi_1 = \sum_{i=1}^{M} (1-\Theta_i)\,\Pi_i, \\ \Pi_2 = \Theta_1\,\Pi_1 \\ \text{----------} \\ \Pi_i = \Theta_{i-1}\,\Pi_{i-1} \\ \text{----------} \\ \Pi_M = \Theta_{M-1}\Pi_{M-1} + \Theta_M\,\Pi_M \end{cases}$$

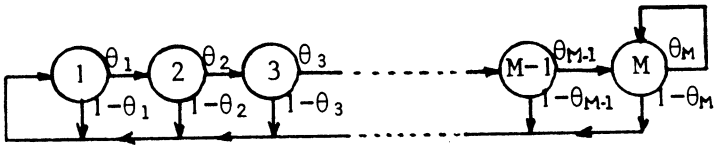


FIGURE 13

et nous obtenons :

$$(16)\quad \begin{cases} \Pi_i = \Pi_1 \prod_{j=1}^{i-1} \theta_j & i = 2,\cdots M-1 \\ \Pi_M = \dfrac{\Pi_1}{1-\theta_M} \prod_{j=1}^{M-1} \theta_j & \\ \sum_{i=1}^{M} \Pi = 1 & \end{cases}$$

Modèle B

Nous supposons pour ce modèle que lorsqu'un programme est interrompu dans l'état i, il sera réactivé dans un des états 1,2,...i avec la même probabilité 1/i. Nous avons donc

$$(17)\qquad \alpha^B_{ij} = \begin{cases} 1/i & j=1,\ldots i \\ 0 & j>i \end{cases} \qquad i = 1,\ldots M$$

et $P = (p_{ij})$ est donné par

$$P = \begin{cases} \theta_i & ,i = 1,\ldots M-1; j = i+1 \\ (1-\theta_i)/i & ,i = 1,\ldots M-1; j = 1,\ldots i \\ (1-\theta_M)/M & ,i = M\ ;\ j=1,\ldots M-1 \\ \theta_M + (1-\theta_M)/M & ,i = j=M \\ 0 & \text{sinon} \end{cases}$$

L'équation (18) s'écrit

$$\begin{cases} \Pi_1 = \sum_{j=1}^{M} \left(\dfrac{1-\theta_j}{j}\right) \Pi_j\ , \\ \Pi_2 = \theta_1 \Pi_1 + \sum_{j=2}^{M} \left(\dfrac{1-\theta_j}{j}\right) \Pi_j\ , \\ \text{----------} \\ \Pi_i = \theta_{i-1} \Pi_{i-1} + \sum_{j=i}^{M} \left(\dfrac{1-\theta_j}{j}\right) \Pi_j\ , \\ \text{----------} \\ \dfrac{1-\theta_i}{} \end{cases}$$

Nous faisons l'hypothèse que les pages sont déchargées une par une, durant l'interruption. En supposant que la durée de l'interruption et le temps qui sépare deux déchargements quand j pages sont en mémoire sont distribuées suivant des lois exponentielles de moyennes respectives I et τ_j, alors la probabilité de perdre une page supplémentaire est donnée par :

$$\delta_j = \frac{1/\tau_j}{1/\tau_j + 1/I},$$

et la matrice (α^D_{ij}) s'écrit :

$$(24) \quad \left\{ \begin{array}{ll} \alpha^D_{ii} = 1 - \delta_i, & \\ \alpha^D_{i,i-1} = \delta_i(1-\delta_{i-1}), & \\ \text{----------} & i = 1, \ldots, M \\ \alpha^D_{i,j} = \delta_i\,\delta_{i-1} \cdots \delta_{j+1}(1-\delta_j), & \\ \text{----------} & \\ \alpha^D_{i,1} = \delta_i\,\delta_{i-1} \cdots \delta_2 & \end{array} \right.$$

Pour caractériser les τ_i, nous supposons que le taux de vol de page est constant et également partagé entre les pages en mémoire. Nous en déduisons :

$$(25) \qquad \tau_i = \beta/i$$

où β peut-être défini comme le temps moyen entre deux vols de page par page. Il vient alors :

$$(26) \qquad \delta_i = i/(i+\nu),$$

avec :

$$\nu = \beta/I.$$

La solution explicite du système d'équations (25) ne peut être obtenue dans ce cas. Toutefois le calcul numérique des Π_i est immédiat en partant de Π_M et en utilisant la relation de récurence :

$$\Pi_{j-1} = \frac{1}{\theta_{j-1}}\left[\Pi_j - \sum_{i=j}^{M} (1-\theta_i)\alpha_{ij}^{D}\,\Pi_i\right], \tag{27}$$

et la condition de normalisation :

$$\sum_{i=1}^{M} \Pi_i = 1.$$

Bibliographie

[1] J.C. ADAMS, G.E. MILLIARD — "Performance measurements on the Edinburgh Multi-Access System" Proc. ICS 75, ACM/AFCET, Antibes (juin 1975), North-Holland Publishing Company.

[2] M. BADEL et al — "Adaptive Optimization of a time-sharing system's performance" IEEE Proceedings 63,6 (June 75)

[3] A. BATSON, P. MADISON — "Charateristics of program localities" CACM 19,5 (May 76)

[4] L.A. BELADY, C.J. KUEHNER — "Dynamic space-sharing in Computer Systems" CACM 12,5 (May 69)

[5] A. BENSOUSSAN et al — "The MULTICS Virtual Memory" Proceedings of the 2nd Symposium on Operating Systems Principles, ACM, Princeton, Oct. 68)

[6] D.G. BOBROW et al — "A paged time sharing system for the PDP-10" CACM 15,3 (March 72)

[7] B.S. BRAWN, F.G. GUSTAVSON — "Program behaviour in a paging environment" Proc. Fall Joint Computer Conference (1968)

[8] J. BRIAT, X. ROUSSET DE PINA, J.P. VERJUS — "Le projet OURS : ses principes" Congrès AFCET'76, Paris (Nov. 76)

[9] P. BRYANT — "Predicting working-set sizes" IBM J. Res. Dev. (May 75)

[10] J.P. BUZEN — "Queueing network models of multiprogramming" Ph. D. Thesis, Harvard University (1971)

[11] J.P. BUZEN — "Computational Algorithms for closed queueing networks with exponential servers" CACM 16,9 (September 1973)

[12] A. CHANG, S.S. LAVENBERG — "Work-rates in closed queueing network models" Op. Res. 22 (1974)

[12b] P.J. DENNING — "The working set model for program behaviour" C ACM 11, 5 (May 68)

[13] P.J. DENNING — "Thrashing : its causes and prevention" Proc. Fall Joint Computer Conference, (1968)

[14] PJ. DENNING — "Virtual Memory" Computing Surveys 2, 3 (September 70)

[15] P.J. DENNING, S.C. SCHWARTZ — "Properties of the working set model" CACM 15,3 (March 72)

[16] P.J. DENNING, K.C. KAHN, J. LEROUDIER, D. POTIER — "Optimal Multiprogramming" à paraître dans Acta Informatica

[17] E. GELENBE — "Modèles de comportement de systèmes informatiques" Thèse d'Etat, Université de Paris VI (Nov. 73

[17b] E. GELENBE, D. POTIER, A. BRANDWAJN, J. LENFANT — "Gestion optimale d'un ordinateur multiprogrammé à mémoire virtuelle" 5th Conference on Optimization Technique, Part 1. Lecture Notes in Computer Sciences,4,Springer Verlag,73

[18] E. GELENBE — "An unified approach to the evaluation of a class of replacement algorithm" IEEE - TC C22, 6 (1973)

[19] W.J. GORDON, G.F. NEWELL — "Closed queueing networks with exponential servers" Op. Res. 15 (1967)

[20] IBM Corp. — "Virtual machine Facility 1370 : Planning Guide IBM Corp.,Publ. n°GC 20-1801-0, 1972.

[21] J.R. JACKSON — "Job-shop like queuing systems" Management Science 10, 1 (Oct 63)

[22] T. KILBURN et al — "One level Storage System" IRE Trans. EC 11,2 (April 1962)

[23] S. KRAKOWIAK — "Conception et réalisation de systèmes à accès multiple : allocation de resssources" Thèse de Doctorat-ès-Sciences, Paris (1973)

[24] S.S. LAVENBERG, G.S. SHEDLER — "Derivation of confidence intervals for work-rate estimators in a closed queueing network" SIAM J. on Comp. 4, 2 (June 75)

[25] J. LENFANT — "Comportement des programmes dans leur espace d'adress Application à la gestion des mémoires hiérarchisées". Thèse d'Etat, Université de Rennes (Déc. 74)

[26] J. LENFANT — "Comparison of the Working-set and bounded locality intervals of a program " 2nd International Workshop on Modelling and Performance Evaluation of Computer Systems, EURATOM-ISPRA, Stresa, Italy (Oct. 76)

[27] J. LEROUDIER, D. POTIER — "Principles of optimality for multiprogramming" Proc. of the International Symposium on Computer Performance and Evaluation, ACM/SIGMETRICS, IFIP WG 7. Harvard University; Cambridge (Mass. (March 76)

[28] J. LEROUDIER, P. BURGEVIN — "Characteristics and models of program behaviour" Proc. ACM Annual Conference 1976, Houston (Texas).

[29] J. LEROUDIER — "Système adaptatifs à mémoire virtuelle" Thèse d'Etat, Université de Grenoble, 1977

[30] P.A. MEYER, L.H. SEAWRIGHT — "A virtual time-sharing system" IBM Syst. J. 9, 3 (1970)