UNIVERSITE SAAD DAHLAB DE BLIDA

Faculté des Sciences

Département d'Informatique

MEMOIRE DE MAGISTER

Spécialité : Systèmes d'information et de connaissances

Application de Lois de Commande Automatique à l'Amélioration de la QoS d'un Serveur Web

Par

Khelifa Kezrane

Devant le jury composé de :

Djamel Bennouar	**MCA, U. Blida**	**Président**
Nadjia Benblidia	**MCA, U. Blida**	**Examinateur**
Med Bachir Yagoubi	**MCA U. Laghouat**	**Examinateur**
Nacereddine Lagraa	**MCA, U. Laghouat**	**Examinateur**
Malik Loudini	**MCA, ESI. Alger**	**Promoteur**
Walid Khaled Hidouci	**MCA, ESI. Alger**	**Invité**
Narhimene Boustia	**MAA, U.Blida**	**Invité**

Blida, Septembre 2011

Dédicace

A ma mère,

A mes frères et sœurs,

A ma femme et mes enfants,

pour leur patience et soutien perpétuel qu'ils m'ont prodigué pour affronter tous les moments difficiles.

Khelifa.

Remerciements

Je tiens à exprimer toute ma reconnaissance à Messieurs Malik Loudini et Walid Khaled Hidouci, respectivement directeur et co-directeur de ce mémoire, pour m'avoir accueilli, encadré et mis à ma disposition tous les moyens nécessaires et logistiques, leur patience, également par leurs idées, et avec lesquels j'ai eu de nombreux échanges durant toutes les années passées à l'accomplissement de ce travail.

J'adresse tous mes remerciements à Monsieur Djamal Bennouar chef du conseil scientifique, pour avoir accepté de présider le Jury de soutenance, et pour m'avoir soutenu et conseillé.

Je remercie également Monsieurs Mohamed Bachir Yagoubi doyen de La faculté des sciences et des sciences de l'ingéniorat de l'université de Laghouat, pour son accord d'être membre du jury malgré ses occupations.

Je tien à remercier Mme Nadjiba Benblidia responsable de l'école doctorale et Mlle Narhimene Boustia pour leurs participation au jury ainsi que pour leurs conseils.

Nasreddine Lagraa, directeur du laboratoire LIM à l'université de Laghouat, en sa qualité de membre du Jury, et avec lequel ce fut un plaisir de travailler.

Je dois également exprimer ma gratitude à tous nos enseignants surtout Cherif Zahar Amine, et Mme Saliha Oukid Kouas pour tout le soutien durant ces années de recherche et leurs contributions à la réussite de l'école doctorale Informatique.

Hadj Mostefa Zergot et Hadj Guesmia Naouri pour leurs conseils, générosité, et compréhension.

Mohamed Lamine Benmaidi qui m'a accompagné tout au long de cette recherche et qui a su supporter avec bienveillance mon humeur des derniers moments.

Je n'oublierai pas Monsieur Mohamed Benyagoub pour son aide précieuse, à qui je témoigne toute mon amitié.

Je remercie mes amis pour leur soutien moral, et mes collègues de travail à la direction de l'éducation de Djelfa, surtout Monsieur Aissa Boutaleb.

ملخص

خوادم الإنترنت التي تستخدم في تنفيذ طلبات الزبائن، سياسة الأول فالأول و حسب ورود الطلب، تكون جودة الخدمة فيها سيئة خاصة خلال فترات الذروة ، مما يؤدي إلى أوقات انتظار غير مقبولة، و خاصة بالنسبة للمعاملات الحساسة. وقد اقترح نموذج تمييز الخدمة ، والمعروف باسم DiffServ ، كحل فعال وقابل للتطوير و يقدم وسيلة فعالة لاستغلال الخوادم من أجل تحسين الاتصال بالإنترنت. رغم ذلك، يمكن لهذا النموذج أن ينحرف عن الأهداف المسطرة له، لهذا اعتمدت أساليب التحكم لتوفير حلولا لهذا النوع من المشاكل. نتناول في هذه المذكرة النماذج الموجودة في مختلف البحوث المنشورة و التي تستخدم تقنية DiffServ مع التحكم ذو التغذية الرجعية، و نقترح مقارنة بين هذه النماذج على أساس تقييم تحليلي، لاختيار النموذج الأنسب في استغلال خوادم الانترنت، و هذا بهدف تحسين جودة الخدمة فيها.

Abstract

Internet servers using the FCFS scheduling models leads to a bad QoS. Especially during heavy load periods causing impredictable response delay which is not acceptable to time sensitive transactions. Differentiated service known as Diffserv has been proposed as an efficient and scalable solution to provide better services for internet communication. However, it is possible that some differentiation service policies objectives can deviate . For this, the feedback control methods are adopted to provide solutions to such problems. In this issue, we examine the existing allocation policies in the literature and applying the technique DiffServ with feedback control. We make a comparison between these policies to reallocate the Web server based on the analytical assessment. Thus we propose a control model to optimize the QoS Web Server.

Résumé

Les serveurs Internet utilisant la politique qui traitent les demandes de manière FCFS conçoivent des mauvaise QoS, surtout durant les périodes de surcharge, ce qui entraîne des délais de réponse inacceptables pour les transactions sensibles au temps. La différentiation des services, connue sous le nom de DiffServ, a été proposée comme une solution efficace et évolutive qui offre un meilleur service pour la communication Internet. Le modèle DiffServ crée un moyen performant pour la réallocation des serveurs. Cependant, il se peut que certaines politiques de différentiation des services puissent s'éloigner des objectifs visés. Pour cela, les méthodes de contrôle de rétroactions sont adoptées pour apporter des solutions à ce genre de problèmes. Cet article étudie les politiques d'allocation existant dans la littérature et qui appliquent la technique DiffServ avec un contrôle Feedback. Nous faisons une comparaison entre de ces politiques de réallocation du serveur en se basant sur l'évaluation analytique. Ainsi nous proposons un modèle de contrôle pour optimiser la QoS du serveur Web.

TABLE DES MATIERES

INTRODUCTION ……. 9

CHAPITRE I:
SERVEUR WEB ET QUALITE DE SERVICE
1.1 Introduction ……..…….. 13
1.2 Le Serveur Web ……..…….. 14
1.2.1 Serveur http …….. 14
1.3 L'architecture du serveur Web …….. 15
1.3.1Le serveur Web et la stratégie de concurrence ……..... 15
1.3.2 Le serveur Web multithread ……... 15
1.3.3 Le serveur Web modulaire ……. 16
1.4 La connexion persistante HTTP 1.1 …….. 17
1.5 Qualité de Service et la performance des serveurs Web …….. 19
1.5.1 Le système DiffServ et la QoS du serveur Web ……. 20
1.5.2 Les intentions légitimes de la QoS ……. 21
1.5.3 Les politiques de classification de services ……. 21
1.5.4 L'impact des délais de connexion sur la QoS 23
1.6 Conclusion …….. 23

CHPITRE II:
PRELIMINAIRE SUR LA COMMANDE AUTOMATIQUE
2.1 Introduction …… 24
2.2 Le système de commande …… 24
2.3 La rétroaction …….. 26
2.3.1 Les objectifs de la commande par rétroaction ……... 28
2.3.2 Les propriétés de la rétroaction ……... 29
2.4 Le Feedforward …… 30
2.5 Les lois de commande élémentaires …… 31
2.5.1 Action proportionnelle …… 32
2.5.2 Action intégrale …….. 33
2.5.3 Action dérivée …… 33
2.5.4 Régulateur, PD, PID …….. 35
2.6 Méthodes de calcul des gains …….. 36
2.7 Conclusion ……. 37

CHAPITRE III:
LES MODELES D'ALLOCATION DES SERVICES DANS LES SERVEURS WEB

3.1 Introduction …….. 38
3.2 Les modèles des files d'attente classiques ……..... 38
3.2.1 Généralités sur les files d'attente ……..... 39
3.2.1.1 Notations de Kendall …… 39
3.2.1.2 Notations utilisées pour les évaluations ……. 39
3.2.1.3 La formule de Little …….. 40
3.2.1.4 Indicateurs de qualité de service et évaluations des files ……... 41
3.2.2 Le modèle classique Best effort …… 41
3.2.2.1 Le modèle de file d'attente à taille infinie (M/M/1) ……. 41
3.2.2.1.1 Hypothèses ……... 41
3.2.2.1.2 Evaluations analytiques ……. 42
3.2.2.1.3 Evaluation expérimentale ……. 42
3.2.2.2 Le modèle de file d'attente à taille finie (M/M/1/K) ……. 44
3.2.2.2.1 Hypothèses ……... 44
3.2.2.2.2 Evaluation analytique …… 44
3.2.2.2.3 Evaluation expérimentale …….. 46
3.2.3 Les modèles classiques à services différenciés ……. 47
3.2.3.1 Le modèle de Gestion avec priorité absolue (M/M/1/∞/Priorité) ……... 47
3.2.3.1.1 Hypothèses ……. 48
3.2.3.1.2 Evaluation analytique …….. 48
3.2.3.1.3 Evaluations expérimentales ……. 49
3.2.3.2 Le modèle de Gestion avec priorité relative (pondérée) …… 51
3.2.3.3 Le modèle de Gestion hybride (Exemple : $M^k/GI^k/1/\infty$/Priorité + WFQ) 53
3.2.3.3.1 Hypothèses …….. 53
3.2.3.3.2 Evaluation analytique …..…… 54
3.2.3.3.3 Evaluations expérimentales …… 55
3.2.3.4 Le modèle à files d'attente avec seuils …….. 57
3.2.3.4.1 Exemple …… 57
3.3 Les modèles des serveurs avec contrôle …… 60
3.3.1 Problématique des contrôles du QoS dans les serveurs …….. 60
3.3.2 Architecture générale de contrôle dans les serveurs Web …….. 61
3.3.3 Le contrôle Feedback des systèmes non DiffServ …….. 62
3.3.4 Le contrôle dans les systèmes DiffServ ……..... 62
3.3.4.1 Le contrôle d'admission……..... 63
3.3.4.2 Le contrôle selon les formalités de performance des services ……... 65
3.3.4.2.1 Le contrôle absolu de performance …… 66
3.3.4.2.2 Le contrôle relatif de performance ……. 66
3.4 Conclusion ……... 70

CHAPITRE IV
ETUDE COMPARATIVE ET PROPOSITION DE SYSTEME

4.1 Introduction ……..……..……. 71
4.2 La QoS et le serveur Web GPS ……..…….. 72
4.3 Analyse du système DiffServ de délais relatifs ……..…….. 73
4.4 Analyse du système DiffServ WFQ ……..……. 76
4.5 Comparaison des modèles délais relatifs, WFQ & le Best effort …… 80
4.5.1 Justification analytique ……..…….. 83
4.6 Description du système adopté ……..…….. 84
4.6.1 La perception Feedforward du paramètre de QoS …….. 86
4.6.2 Principe de contrôle Feedback des délais absolus de connexions …….. 87
4.6.3 Le contrôleur des délais absolus de connexion ……... 89
4.6.3.1 Construction du modèle ……..…… 91
4.6.3.2 Formulation de l'action proportionnelle intégrale dérivative (PID)……. 91
4.6.3.3 Analyse des propriétés du système ……..…….. 93
4.7 Conclusion ……. 96

CHAPITRE V
SIMULATION ET INTERPRETATION DES RESULTATS

5.1 Introduction …… 97
5.2 Perception des Paramètres de contrôle PID ……... 98
5.3 Le modèle DiffServ des délais relatifs vis-à-vis de Best effort……. 100
5.4 Simulation du système proposé et interprétation des résultats ……. 102
5.4.1 Le contrôle des délais absolus de connexion à deux classes de service 102
5.4.2 Le contrôle des délais absolus à trois classes de service ..……... 105
5.4.3 Le Système de contrôle hybride ……..... 108

CONCLUSION ……. 111

LISTE DES FIGURES

Fig. I- 1: Une interaction client serveur simple 13
Fig. I-2: Architecture du serveur Web 16
Fig. I-3: Architecture d'un serveur Web modulaire 17
Fig. I-4: Architecture du serveur Web Apache 18
Fig. I-5: Le délai de connexion vis à vis du délai de réponse 23
Fig. II-1: Schéma d'un système à commander 25
Fig. II-2: Un système de commande manuelle MIMO 26
Fig. II-3: Les éléments de contrôle Feedback 27
Fig. II-4: Les propriétés d'un système automatique 29
Fig. II-5: Le modèle de contrôle en boucle ouverte 30
Fig. II-6: Schémas de régulateur rétroaction 31
Fig. II-7: Les graphes des régulateurs P, I et D vis-à-vis des valeurs de gains 34
Fig. II-8: Les procédés PI, PD et PID vis-à-vis du régulateur P 35
Fig. II-9: Mise en oscillation d'un système 37
Fig. III-1: Le modèle de la file d'attente Best effort infinie M/M/1 42
Fig. III-2: L'évaluation par simulation évènementielle et l'évaluation analytique de la charge des files 43
Fig. III-3: L´évaluation par simulation évènementielle et l'évaluation analytique de la charge des files en cas de variation de trafic 43
Fig. III-4: Le modèle de la file d'attente à taille finie M/M/1/K 44
Fig. III-5: L'évaluation par simulation évènementielle et l´évaluation analytique de la charge des files 46
Fig. III- 6: L'effet du paramètre K sur la charge N, le temps de réponse R, et le taux d'utilisation ρ dans une file d'attente 46
Fig. III-7: Le modèle de Gestion avec priorité M/M/1/∞/Priorité 47
Fig. III-8: Le modèle de Gestion avec priorité $M^K/M^K/1/\infty$/Priorité 47
Fig. III-9: Comparaison de l´évaluation par simulation événementielle et de l'évaluation analytique de la charge des files $M^2/M^2/1/\infty$/Priorité 50
Fig. III-10: La discipline de service GPS 51
Fig. III-11:Un modèle d'une Gestion hybride 53
Fig. III-12: L'évaluation par simulation évènementielle vis-à-vis de l'évaluation analytique de la charge des files du modèle $M^4/GI^4/1/\infty$/Priorité + WFQ 56
Fig. III-13: Modèle multi QoS classes du serveur Web 57
Fig. III-14: La politique d'allocation des services des files à seuils 59
Fig. III-15: Architecture générale d'un optimiseur à rétroaction 61
Fig. III-16: Architecture de contrôle du serveur apache 62
Fig. III-17 : Système de contrôle pour les systèmes à queue 63

Fig. III-18: Un modèle DiffServ d'un serveur Web 64
Fig. III-19: Modèle de contrôle feedback des Stratégies d'allocations 68
Fig. IV-1 : Processus d'allocation suivant la pondération 72
Fig. IV-2: Graphes du temps de connexion dans un système DiffServ des délais relatifs..... 75
Fig. IV-3: Graphes du temps de connexion A, B et C respectivement dans un système DiffServ WFQ..... 80
Fig. IV-4 : Comparaison entre les graphes du temps de connexion DiffServ et le Best effot.. 81
Fig. IV-5: Le dépassement des ressources disponibles dans le DiffServ WFQ 82
Fig. IV-6: Principe d'ajustement dynamique 87
Fig. IV-7 : Processus de contrôle des délais absolus 90
Fig. IV-8: La boucle du contrôle Feedback PID 94
Fig. V-1: Choix de paramètres par la méthode de Ziegler & Nichols 98
Fig. V-2: L'effet de la régulation du contrôleur PID 99
Fig. V-3: L'effet des perturbations sur la régulation du contrôleur PID 99
Fig. V-4: Le modèle DiffServ des délais relatifs vis à vis de Best effort..... 101
Fig. V-5: Le choix des taux de requêtes survenues aux deux classes 102
Fig. V-6: Les taux des services éventuels du contrôle des délais absolus à deux classes..... 103
Fig. V-7: Le contrôle des délais absolus à deux classes..... 104
Fig. V-8: Le choix des taux de requêtes survenues aux trois classes 105
Fig. V-9: Les taux des services éventuels du contrôle des délais absolus à trois classes ... 106
Fig. V-10: Le contrôle des délais absolus à trois classes 107
Fig. V-11: Activités des deux classes imposées par le système de contrôle hybride 109
Fig. V-12: Activités des trois classes imposées par le système de contrôle hybride 110

LISTE DES TABLEAUX

Tab. II-1 : Une comparaison entre les deux modes de contrôle 31
Tab. II-2 : Valeurs préconisées des Coefficients PID par Ziegler et Nichols 37
Tab. III-1 : Paramètres des tests de simulation du modèle M/M/1 43
Tab. III-2 : Paramètres des tests de simulation du modèle $M^2/M^2/1/\infty$/Priorité 49
Tab. III-3 : Paramètres des tests de simulation du modèle $M^k/GI^k/1/\infty$/Priorité+WFQ 55
Tab. IV-1 : Paramètres des investigations WFQ 78
Tab. V-1 : Paramètres de la simulation de DiffServ des délais relatifs et de Best effort 100

INTRODUCTION

L'Internet a fourni un cadre très agréable de publication et d'échange d'informations à faible coût. Les entreprises ont mis en place des sites Web pour le marketing et les interactions avec les clients. Le problème de l'imprévisibilité du temps de réponse est extrêmement important au cours de la recherche d'informations sur le Web notamment pour les applications avec des contraintes de temps qui utilisent le Web comme interface distribuée [**1**].

Cependant, de nombreuses entreprises ont trouvé que les premières générations de serveurs Internet étaient insuffisantes pour couvrir les transactions commerciales en raison des problèmes de fiabilité, d'évolutivité, de sécurité, de performance et d'incohérence.

Il existe de nombreux services Internet : les serveurs web, les serveurs de messagerie, les services de transmission multimédia en continu, les serveurs commerciaux ou encore les serveurs de base de données. Ces services reposent le plus souvent sur l'architecture client-serveur, dans laquelle plusieurs clients accèdent concurremment à un service en ligne fourni par un serveur (comme la lecture d'une page web, l'envoi d'emails ou le paiement d'un panier d'achats) **[2]**.

Un serveur de messagerie est par exemple soumis à une plus forte charge le matin que le reste de la journée, puisqu'il est commun de consulter ses emails en arrivant au travail. Dans le cas extrême, une forte charge peut provoquer un écroulement du serveur et entraîner une indisponibilité du service **[2]**. Les coûts engendrés par ce phénomène sont estimés à plus de 2 millions de dollars par heure pour les entreprises de télécommunication et les entreprises financières **[3].**

Cette raison a alimenté un immense enthousiasme dans la communauté de la recherche. Les défis demeurent à relever dans deux principaux fronts. Tout d'abord, les serveurs Web sont des entités centrales de livraison du contenu et de leurs ressources qui doivent être utilisées de façon appropriée. D'autre part, c'est l'Internet qui apporte finalement le contenu aux utilisateurs.

La fameuse évolution du matériel possédant des processeurs plus rapides, des systèmes d'exploitation plus efficaces, et l'augmentation de la bande passante, offrent des solutions pour améliorer le débit et le temps de réponse des serveurs Internet. Mais malgré cela, la nature des services et la croissance rapide du volume de trafic, exprimées par la croissance impressionnante du nombre d'internautes, rendent l'échelle de la bande passante du réseau et la capacité du serveur insuffisantes et coûteuses, en l'absence d'une politique efficace prenant en compte l'aspect de la qualité de service (**QoS**) du serveur Web **[4].**

Un serveur Internet utilisant la politique ''Best effort'', traite les demandes de manière FCFS (premier arrivé, premier servi), a conduit à une mauvaise répartition des ressources du serveur (processeur, mémoire) et réseau, surtout durant les périodes de surcharge et entraîne des délais de réponse inacceptables pour les transactions sensibles au temps **[1]**.

De ce fait, il est nécessaire de concevoir et d'évaluer des serveurs Internet qui peuvent fournir des réponses rapides à des tâches prioritaires, en réduisant au minimum la performance des tâches à faible priorité sans trop dégrader la performance globale du système.

La différenciation des services dans les réseaux informatiques, connue sous le nom de ''DiffServ'' a été proposée comme une solution efficace et évolutive qui offre un meilleur service pour la génération récente de la communication Internet. Le modèle ''DiffServ'' crée un moyen intéressant pour les grands trafics prioritaires par la sélection et la réallocation des ressources du réseau pendant les périodes de pointe **[5]**.

Le même principe a été inspiré par des études antérieures dans le but de fournir de meilleures qualités de service aux tâches prioritaires. Cette politique ''DiffServ'' permet de définir des niveaux de priorités entre les différents services demandés. En particulier, les délais de réponse des requêtes critiques qui peuvent être atteints par allocation de ressources supplémentaires en fonction de leurs priorités.

Les études d'optimisation des serveurs Web s'intéressent à trouver des solutions, et particulièrement d'incorporer le concept QoS afin de permettre la différenciation entre les différents trafics des services Web. Généralement ces recherches se focalisent sur des techniques dont les modèles à queue représentent la masse essentielle pour gérer les différentes politiques d'allocation. Cependant, il se peut que ces politiques puissent s'éloigner des objectifs visés surtout dans les cas de pointes (surcharges).

Récemment, la théorie du contrôle a été identifiée comme un fondement théorique de la performance et offre des solutions potentielles aux applications complexes, telles que la planification en temps réel, les serveurs Web, les gestionnaires de stockage, le contrôle de puissance de CPU, et le routage dans les réseaux informatiques. Ces solutions offrent une interface entre le calcul de performance et le contrôle du système, et automatise de nombreuses parties de la rétroaction afin d'obtenir une qualité QoS satisfaisante dans ces applications. En effet, ces méthodes fournissent des ''middlewares'' pour faciliter le développement des applications software robustes [**6**].

On peut citer trois types de problèmes d'optimisation par contrôle ciblant les serveurs Web et qui sont généralement considérés comme des domaines de recherches **[7]**:

- Le premier problème est le DiffServ. C'est là où notre travail s'inscrit, en établissant des différents niveaux de services, pour pouvoir répondre avec une certaine qualité aux différentes requêtes au serveur Web.
- Le deuxième problème est l'amélioration du temps de réponse en prenant en compte la configuration du système (nombre maximum de clients).
- Le troisième problème est le contrôle d'utilisation des ressources (Mémoire et CPU) selon une façon non excessive en raison de considération de fiabilité (par exemple, certains logiciels deviennent fragiles à de lourdes charges), ou à cause de la conception d'un système qui pose une réserve de capacité qu'il ne peut franchir.

Notre travail consiste à étudier les politiques d'allocation des ressources d'un serveur Web, particulièrement les politiques appliquant la technique DiffServ avec une commande rétroaction (''Feedback contrôle'').

Le but de cette étude est de tirer profit des avantages de la régulation pour avoir une bonne qualité de service au niveau des serveurs Web. Ainsi, on doit proposer une solution permettant l'ajustement dynamique de paramètres afin de contrôler les délais de connexions des classes de services. En effet, cette régulation doit être relative aux besoins des classes et doit permettre de favoriser certains flux (les requêtes prioritaires) lors des situations de surcharge.

L'étude est tributaire d'une évaluation analytique des politiques d'allocation des services existants, et qui appliquent le ''DiffServ'' avec un contrôle par l'action de rétroaction, pour

agir sur le système de façon à ce qu'il se comporte de la manière souhaitée. Le choix de l'approche adoptée est justifié par une comparaison de ces politiques de réallocation du serveur. Ainsi une modélisation du système de contrôle a été adaptée pour optimiser la QoS du serveur Web.

Ce mémoire est organisé comme suit :

- Chapitre 1 : décrit le Serveur Web, le rôle de la Qualité de Service (QoS) et l'impact du temps de connexion sur l'amélioration de la performance des serveurs Web.
- Chapitre 2 : définit les notions préliminaires sur les lois des commandes et le modèle feedback.
- Chapitre 3 : expose les différents modèles :

    - Les modèles à queue : évaluation statistique de certains modèles classiques des files d'attente et certains modèles établis par la politique ''DiffServ''.
    - Les modèles avec contrôle Feedback : en se basant sur les modèles classiques des files d'attente, et à l'égard de certains critères des politique d'ajustement rétablis sur les paramètres de réallocation du serveur, pour obtenir une meilleure qualité de service.

- Chapitre 4 : est consacré à l'étude comparative de certaines politiques de contrôle et à la proposition de notre solution :

    - Une étude avec des évaluations analytiques des paramètres de QoS des systèmes existants accompagnées d'une comparaison.
    - Une modélisation d'un système adopté pour améliorer la QoS de serveur Web.

- Chapitre 5 : est affecté à la simulation du système adopté et interprétation des résultats.

Enfin, une conclusion et des perspectives pouvant compléter ce travail.

CHAPITRE I
SERVEUR WEB ET QUALITE DE SERVICE

1.1 Introduction

L'Internet est le réseau informatique mondial qui rend accessibles au public des services variés comme le courrier électronique, la messagerie instantanée et le World Wide Web, en utilisant le protocole de communication IP (Internet Protocol) **[8].**

Le « www » est une application Client/Serveur, comprenant des navigateurs et des serveurs communicants, en utilisant le protocole HTTP (Hyper Texte Transfert Protocol) qui correspond à la couche la plus haute du protocole TCP/IP **[9].** Les navigateurs tels que l'Explorer de Microsoft, Mozilla…. fournissent une interface graphique et conviviale pour visualiser les pages Web et les documents sur Internet.

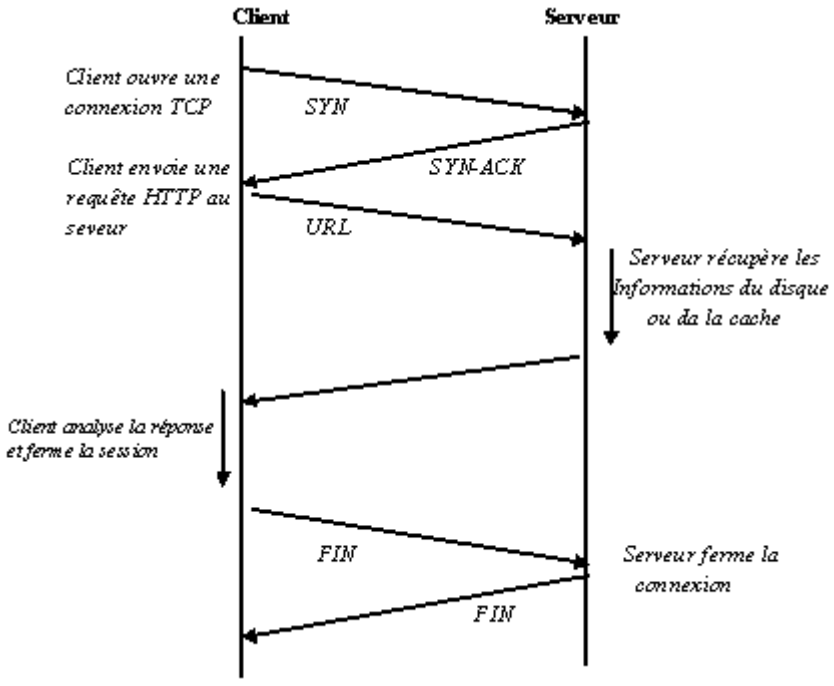


Fig. I- 1 : Une interaction client serveur simple [10]

La Figure I-1 montre une interaction Client/Serveur pour le protocole HTTP. Le client établit la connexion au serveur et interagit en utilisant différentes méthodes définies par le protocole

HTTP. Il envoie une requête pour un objet (exemple : document, image, recherche de base de données, etc…...). Un intervalle de temps variable est nécessaire au serveur pour traiter les requêtes et renvoyer l'objet ou les résultats de ces requêtes. Les objets sont référencés par leur URL (Uniform Ressource Locator).

1.2 Le Serveur Web

Un serveur Web peut être **[8]** :

- un ordinateur tenant le rôle de serveur informatique sur lequel fonctionne un logiciel serveur HTTP ;
- le serveur HTTP lui-même ;
- un ensemble de serveurs permettant le fonctionnement d'applications Web.

1.2.1 Serveur HTTP

Un serveur HTTP ou HTTPd (HTTP daemon) : est un logiciel servant des requêtes respectant le protocole de communication HTTP**.**

Les serveurs HTTP les plus utilisés sont **[8] :**

- Apache HTTP Server de « Apache Software Foundation », successeur du NCSA HTTPd ;
- Internet Information Services (IIS) de Microsoft ;
- Sun Java System Web Server de Sun Microsystems (anciennement iPlanet de Netscape, puis Sun ONE de Sun Microsystems) ;
- Zeus Web Server de Zeus Technology ;

Le serveur HTTP le plus utilisé est Apache HTTP Server qui sert environ 60% des sites Web en 2007 selon Netcraft1.

En outre, un serveur Web renvoie les informations (pages, images, etc...) vers les clients en réponse à leurs requêtes http. Il permet aussi l'intégration des sources variées d'informations par l'intermédiaire des programmes CGI (Common Gateway Interface) qui se chargent de la gestion des réponses aux requêtes des clients **[10]**.

Le serveur Web fait fonctionner plusieurs logiciels qui fonctionnent en parallèle. On retrouve la combinaison Apache (serveur HTTP), MySQL (serveur de base de données) et PHP, tous libres. Sous Linux, cette combinaison s'appelle LAMP (sigle de « Linux, Apache, Mysql,

PHP»); sous Windows, WAMP (« Windows, Apache, Mysql, PHP »); et sous Mac, MAMP («Macintosh, Apache, Mysql, PHP »).

La plupart des ordinateurs utilisés comme serveur Web sont reliés à Internet et hébergent des sites Web du World Wide Web. Les autres serveurs se trouvent sur des intranets et hébergent des documents internes d'une entreprise, d'une administration, etc...

1.3 L'architecture du serveur Web

Le choix de l'architecture des serveurs Web est adopté selon le besoin. On peut distinguer deux architectures des serveurs Web fréquemment utilisés. Avant de présenter ces deux architectures, il est préférable d'expliquer la notion de la stratégie de concurrence implémentée, peu récemment, sur la majorité des serveurs Web.

1.3.1 Le serveur Web et la stratégie de concurrence

Comme les systèmes d'exploitation multitâches, les serveurs Web en général implémentent cette sorte d'exécution, en établant la stratégie de concurrence, pour mieux répondre aux besoins des clients qui accèdent simultanément à leurs ressources. La politique de concurrence est réalisée par la création de processus légers appelés threads, pour chaque connexion entrante en attribuant un thread « thread per request ». Tandis que la création et la terminaison sont des opérations très coûteuses au système d'exploitation tels que UNIX, la croissance en nombre de processus peut entraîner une surcharge du serveur caractérisée par les temps de traitement indésirables des requêtes. En effet, pour réduire les frais généraux de service en évitant les processus dynamiques, une autre alternative consiste à maintenir les processus dans une ''piscine'' préexistante en attendant les demandes de connexion TCP au serveur [**11].**

1.3.2 Le serveur Web multithread

L'architecture d'un serveur Web travaillant avec une politique de concurrence « thread per request » est décrite dans la Figure I-2**.**

Un thread principale ''Listening thread'' reçoit les connections arrivant sur le port TCP 80, et crée un nouveau thread pour chaque nouvelle connexion **[12]**.

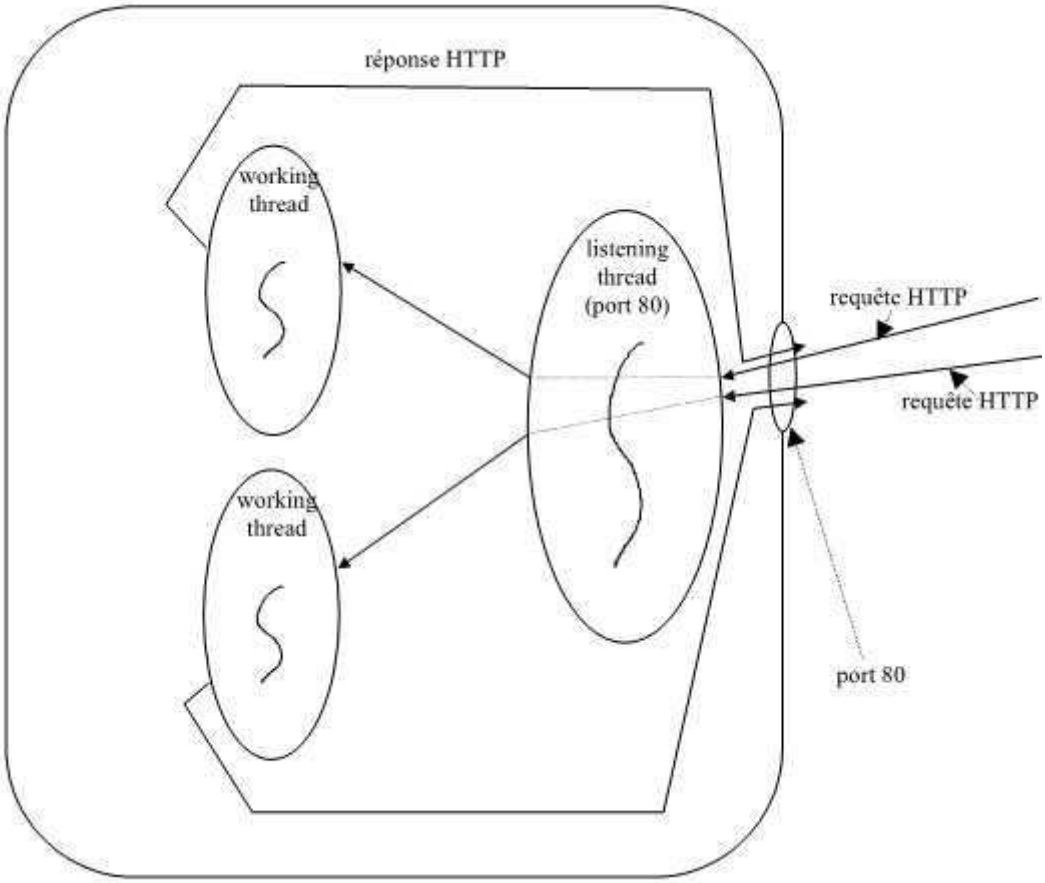


Fig. I-2 : Architecture du serveur Web (thread per request) [12]

1.3.3 Le serveur Web modulaire

Avec l'explosion du Web, les sites web les plus visités doivent être capables de répondre au grand nombre de requêtes. Par conséquent des solutions en utilisant plusieurs serveurs Web en même temps ont été mises en place. L'architecture de serveur Web modulaire est constituée de deux parties, semblables à la Figure I-3 :

- Les nœuds de stockage.
- Les nœuds de transfert (serveurs Web).

Le contenu de système peut être distribué sur tous les nœuds de stockages en utilisant les techniques adoptées par les Data Warehouse.

Cependant le module d'équilibrage des charges (Load Balancing) peut être utilisé pour répartir les requêtes survenues au système.

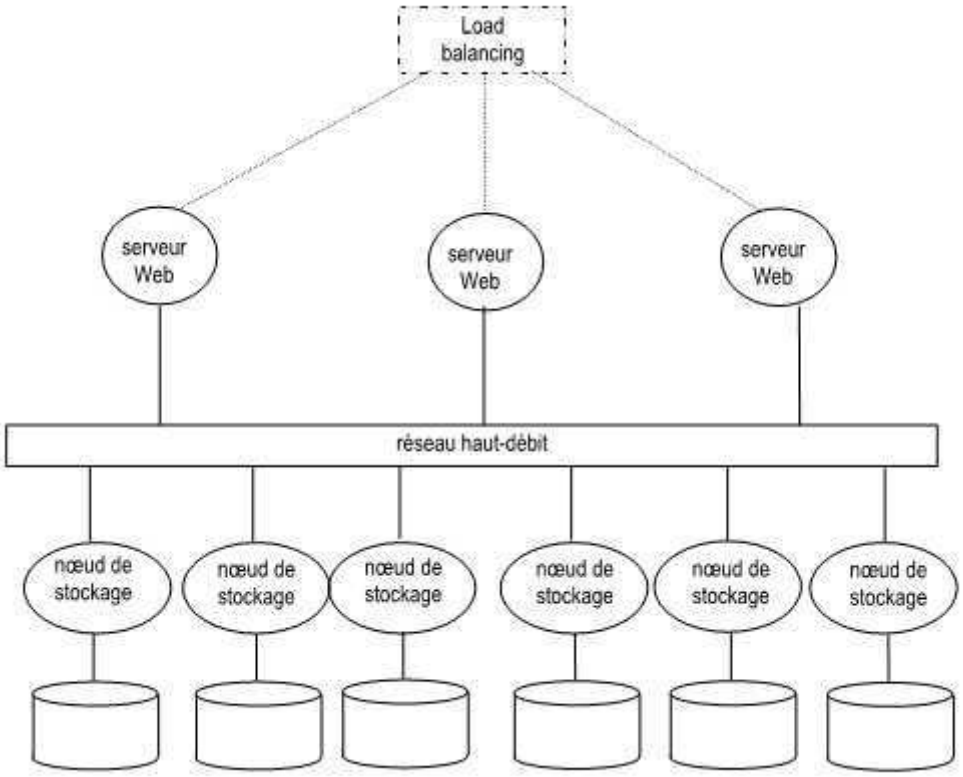


Fig. I-3 : Architecture d'un serveur Web modulaire [12]

1.4 La connexion persistante HTTP 1.1

Dans le protocole HTTP1.0, chaque requête http effectue une connexion TCP. Pour afficher le grand nombre d'objet (texte, image, script…) constituant une page Web, le navigateur doit faire plusieurs requêtes au serveur. Il en résulte un nombre excessif de connexions TCP simultanées. Pour remédier à ce problème, la version HTTP1.1, décrite dans **[13],** a réduit le nombre des liens TCP avec un mécanisme appelé la connexion persistante. Ce dernier permet à des requêtes multiples la réutilisation de la même connexion.

Ainsi, un client HTTP1.1 envoie d'abord sa demande de connexion TCP à un serveur Web, cette requête sera stockée dans une file d'attente sur un port du serveur, ensuite une connexion TCP sera établie entre le client et l'un des processus du serveur, et finalement la demande est effectuée. Le client peut envoyer ultérieurement ses demandes http et recevoir des réponses sur la même connexion.

Le protocole HTTP1.1 exige que la connexion TCP établie reste maintenue même après que la demande soit servie, c.-à-d reste à la prévision d'éventuelles futures demandes. S'il n'y a aucune demande parvenue sur ce sujet au sein du TIMEOUT secondes, la connexion est fermée et le processus qui en est responsable est retourné à la ‘’piscine’’.

Dans le serveur Web Apache HTTP par exemple, qui est structuré sous la forme d'un bassin de processus ou threads, lorsqu'une requête survient, elle est mise en attente dans la queue du serveur jusqu'à ce qu'un processus soit libre (Idle). A ce moment, le processus qui est chargé de traiter cette requête, doit être transmis à l'état occupé (Busy). Si le processus est dans l'état d'attente (Wait), le temps écoulé appelé le KeepAlive, représente le temps de réfléchir. Ce temps indique le temps en laissant la connexion ouverte au profit du même client pour une connexion persistante du protocole HTTP 1.1.

La Figure I-4 montre l'architecture du serveur Web Apache et le flux de la session d'une connexion.

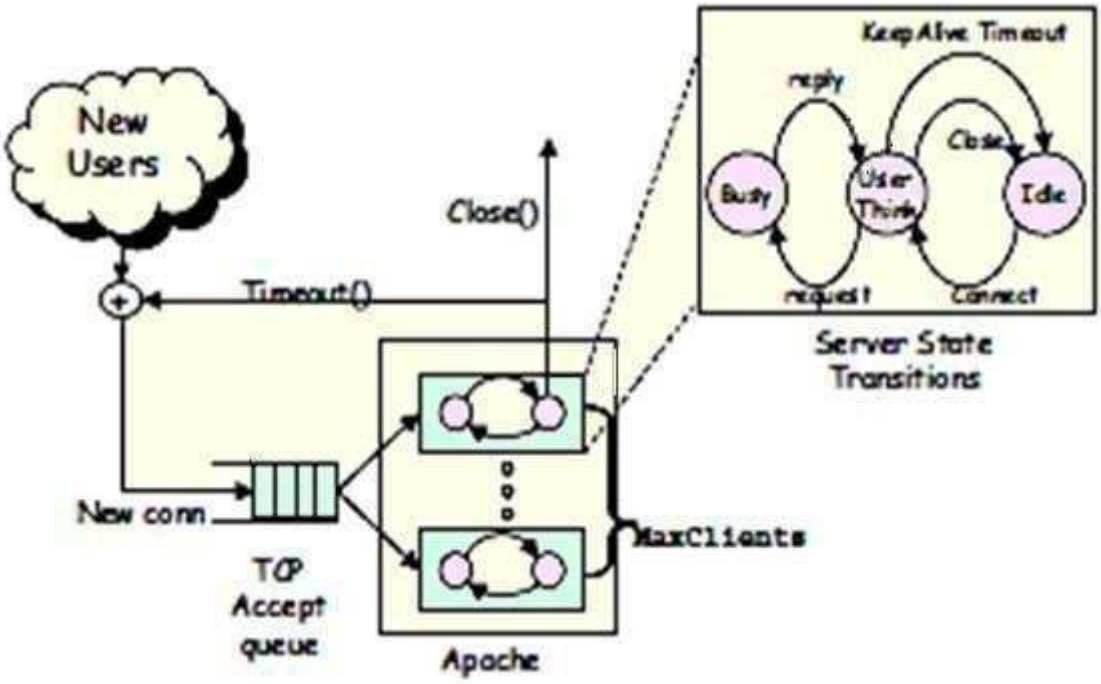


Fig. I-4 : Architecture du serveur Web Apache [15]

Les connexions persistantes produisent un nouveau type de goulot d'étranglement de la ressource sur le serveur, puisqu'un processus du serveur peut être lié à une connexion persistante, même après que la demande soit déjà servie, et qu'il peut mettre le serveur dans une situation de fonction déficiente.

Une façon de résoudre ce problème est d'augmenter le nombre de processus serveur. Toutefois, un trop grand nombre de processus peut causer la violation de la mémoire virtuelle **[14]**, et par la suite dégraderait considérablement les performances du serveur. Dans la pratique, une limite spécifiée par le système d'exploitation est imposée sur le nombre de processus serveur [**11**].

Enfin, on peut noter que les nouvelles connexions subissent des retards plus élevés sur les files d'attente, tandis que par la suite, les demandes qui arrivent sur les connexions sont servies presque immédiatement par leurs processus désignés.

1.5 Qualité de Service et la performance des serveurs Web

Le terme QoS (acronyme de « Quality of Service », en français QdS « Qualité de Service ») désigne la capacité de fournir un service conforme à des exigences en matière de temps de réponse et de bande passante **[16]**.

Les serveurs Web sont conçus pour gérer les différents types de demandes ayant des contraintes de performance. Ces dernières doivent être réunies pour maintenir les services au sens parfait. En conséquence, la qualité de service des serveurs Web comprend le critère de performance du temps et les politiques de classification.

Le temps de traitement d'une requête http dépend de la vitesse du serveur et de la complexité des requêtes. Le temps et la vitesse sont les unités de mesure de base pour la performance d'un serveur. La vitesse des requêtes http servies représente le débit du système. Le temps requis pour exécuter une requête est le temps de réponse du système. Et comme la variation de la taille des objets est significative, le débit est généralement mesuré en termes d'octets par seconde **[10].**

Le temps de connexion est susceptible d'être le critère principal de performance qui touche l'intérêt des clients. Ce temps est le délai entre le moment où une demande est présentée par un client et le moment où la demande est entièrement desservie par le serveur **[4].** Donc le temps de connexion comprend le temps d'attente des requêtes sur les files d'attente et le temps de réponse du serveur (traitement par le serveur). Le temps de connexion est considéré comme une mesure d'optimisation de la qualité de service du serveur web puisqu'il dépend de

la politique d'allocation contrairement au temps de traitement qui ne dépend que du serveur. A la fin de ce chapitre, un graphe montrera une comparaison entre ces deux temps.

Cependant, la tendance de développement technologique du serveur Web a donné que, pour garantir la QoS, on doit varier les délais de connexions pour les différentes catégories de demandes qui sont souvent exprimées en termes de moyennes statistiques **[4]**.

A noter qu'il est important que sur le champ d'application, le terme (QoS) peut avoir une signification beaucoup plus large que les paramètres de performance cités ci-dessus ; la sécurité et l'authenticité peuvent être également introduites dans le champ d'étude de la qualité de service.

1.5.1 Le système DiffServ et la QoS du serveur Web

Un serveur Internet traditionnel utilisant la politique ''Best effort'' traite les demandes de la manière FCFS. Au cours d'une période de forte surcharge, les tâches qui sollicitent des ressources limitées, comme l'ouverture des connexions au serveur et la bande passante des réseaux augmentent, surtout avec les "Retry" des clients impatients, ce qui aggrave la situation de la surcharge. Par conséquent, chaque tâche doit attendre dans une file d'attente pour un long moment avant de se servir.

Cette incommodité entraine l'exigence des programmes plus élaborés d'allocation des ressources, plutôt que le modèle de service ''Best effort'', qui doivent être adoptés pour fournir des services prévisibles au cours de périodes de forte surcharge [**1]**

En raison de la nature ouverte et dynamique des applications Web, la dernière décennie a été témoin d'une augmentation du besoin de provisions sur les différents niveaux de qualité de service (QoS). Et ce afin de répondre à l'évolution de la configuration du système et la disponibilité des ressources, ainsi qu'à satisfaire les différentes exigences du client [**17**].

La différenciation des services dans les réseaux informatiques, connue sous le nom de DiffServ dans la communauté IETF (l'Internet Engineering Task Force) a été proposée comme une solution efficace et évolutive qui offre un meilleur service pour la prochaine génération de la communication Internet **[5**]. Le modèle DiffServ crée un cadre parfait pour les grands trafics prioritaires par la sélection et la réallocation des ressources du réseau pendant les périodes de pointe [**18**].

De même, ce principe est inspiré par des études antérieures sous un nom autrement SDIS (Service Differentiating Internet Server) en but d'offrir des serveurs fournissant de meilleures qualités de service aux tâches prioritaires. En particulier, les délais de connexion des requêtes critiques qui doivent être limitées à l'allocation des ressources CPU ou I /O et à l'égard de leurs priorités.

De ce fait, il est nécessaire de concevoir et d'évaluer des serveurs Internet qui peuvent fournir une réponse rapide à des tâches prioritaires, en réduisant au minimum la performance des tâches à faible priorité sans dégrader la performance du système.

1.5.2 Les intentions légitimes de la QoS

Plusieurs causes motivantes adoptées par les serveurs SDIS peuvent être résumées comme suit **[1] :**

- Les applications médias, tels que les flux audio ou vidéo, ces types de données sensibles au temps qui doivent être respectés pour leur spécificité de livraison et ont besoin de service de priorité plus que d'autres services qui ne sont pas en temps réel.
- Le marché e-commerce prend en charge plusieurs types de transactions nécessitant le classement des services basés selon la génération de revenus.
- L'adoption éventuelle de services différenciés dans la prochaine génération des réseaux Internet rendra les serveurs ''Best effort'' inconcevables et peuvent défier le but principal de celui-ci. Ainsi, afin d'assurer une QoS de bout en bout, les serveurs doivent également fournir une différenciation des services.

1.5.3Les politiques de classification de services

La question de différenciation des services sur le serveur Web peut être accomplie par les différentes politiques de classification. Les politiques sont très différentes selon les objectifs du fournisseur des services ainsi que les types de contenus fournis. Il s'agit souvent d'une pratique commune pour accueillir les différents sites Web dans un seul serveur Web. On présente quelques exemples des politiques de classification qui peuvent être mises en œuvre **[4]** :

Source-based : Certains serveurs Web essayent de faire la distinction entre les demandes en fonction de l'origine de la demande. La politique peut faciliter un meilleur service pour une organisation ou une zone géographique. Le classificateur se penchera sur les adresses IP des hôtes pour déterminer les priorités des différentes classes.

Content-based : les objets sont généralement stockés sur des fichiers dans le serveur. Les types de fichiers peuvent varier d'une HTML simple à différents types de multimédias tels que les objets dynamiques, ces objets sont créés dans le serveur par une interface de programmation ou des scripts déclenchés par des demandes des utilisateurs. Autre exemple dans le e-commerce par exemple, les transactions par carte de crédit a plus d'importance que la demande de la navigation dans un magasin de vente en ligne. Le classificateur dans ce cas analyse les URL demandées et détermine le type du fichier ou l'opération demandée.

Économics-based : dans les centres d'hébergement partagé, le paiement peut déterminer la classification de la demande. Le centre d'hébergement pourrait atteindre un accord sur les niveaux de service avec les propriétaires du site. Les accords préciseront la structure de paiement sur la base de la qualité ou du niveau de service attendu. Le classificateur aura l'information sur la propriété des objets hébergés dans le serveur et fera la classification sur la base de cette propriété.

Popularity-based : la popularité de l'objet pousse souvent le classement des politiques dans de nombreux serveurs Web. Pour avoir une plus grande satisfaction du client, les services plus populaires doivent avoir plus de priorité que les services les moins populaires. Le classificateur peut maintenir les demandes distinctes sur la base des fourchettes de popularités sur l'ensemble de la collection.

Taille-based : La taille des objets demandés pourrait être utilisée pour le classement dans certains serveurs Web. L'attribution de priorités aux objets sur la base de leur taille a un impact sur les performances globales du serveur. Si les gros objets (en termes de données) ont une plus grande priorité, ils peuvent améliorer l'efficacité du serveur, mais les demandes pour les petits objets pourraient subir des retards indésirables. D'autre part, servir des fichiers de plus petites dimensions avec une plus grande priorité, réduit le nombre des requêtes en attente du serveur.

1.5.4 L'impact des délais de connexion sur la QoS

Dans une série d'expériences les auteurs **[11]** ont effectué des comparaisons entre le délai moyen de connexion et le temps moyen de réponse (par requête http) sur un serveur Web, ces expériences ont justifié l'utilisation du délai de connexion comme un indicateur de différenciation des services dans les serveurs Web. Dans ces expériences le ''DiffServ'' n'est pas pris en compte, c'est-à-dire toutes les connexions sont effectuées dans une même classe et tous les processus du serveur sont attribués à cette classe. Le délai de connexion est considérablement plus élevé que le temps de réponse, et augmente avec un rythme beaucoup plus rapide suivant le nombre d'utilisateurs. Si on géra également des expériences avec plus de processus le rapport entre le délai de connexion et le temps de réponse est presque toujours similaire comme celui présenté dans la Figure I-5.

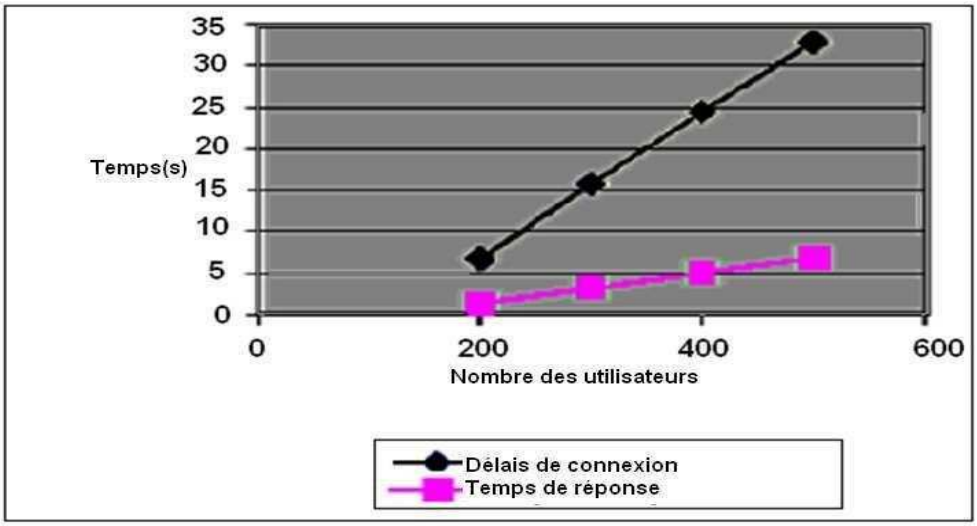


Fig. I-5 : Le délai de connexion vis à vis du délai de réponse [11]

Ces résultats certifient le choix légitime du délai de connexion pour la mesure de la qualité de service dans la politique de différenciation des services ''DiffServ'' utilisée par les serveurs Web.

1.6 Conclusion

Dans ce chapitre nous avons présenté quelques concepts fréquemment utilisés dans le monde des serveurs Web, et l'importance des systèmes DiffServ, ainsi que l'exigence d'introduire la notion de la qualité de service (QoS) comme indication pour achever une meilleure performance des serveurs Web.

En outre, nous avons estimé les délais de connexion pour faire office de paramètre de différenciation des services afin d'améliorer la qualité de service du serveur Web.

CHPITRE II
PRELIMINAIRE SUR LA COMMANDE AUTOMATIQUE

2.1 Introduction

La régulation automatique est l'une des techniques de l'ingénierie offrant les méthodes et les outils nécessaires à la prise de contrôle d'un système physique (installation de production, robot, alimentation électronique stabilisée, etc.) en vue d'imposer un comportement désiré. Cette prise de contrôle s'effectue par l'intermédiaire de certains signaux (grandeurs physiques) qu'il est alors nécessaire de mesurer afin de déterminer l'action à entreprendre sur le système.

Le contrôle étant automatique indique qu'aucune intervention humaine n'est nécessaire. Ces méthodes offrent la possibilité de modifier le comportement statique et dynamique d'une grandeur physique, afin qu'elle évolue conformément aux exigences d'une application.

Dans ce chapitre, on définit les notions courantes dans le domaine des commandes automatiques, ainsi que les différents type d'actions de commandes élémentaires P, I et D.

2.2 Le système de commande

Un système est un assemblage de constituants reliés les uns aux autres et forme une entité individualisée qui se comporte comme telle. Le mot commande est l'action d'agir sur un système de façon à ce qu'il se comporte de la manière souhaitée [**19**]. Donc un système de commande est un ensemble d'éléments en interaction les uns avec les autres de telle sorte qu'il puisse se commander, diriger ou se régler lui-même, ou bien commander, diriger, régler un autre système **[20]**. Les systèmes automatiques assurent en fait deux types de fonctions **[21]** :

- Maintenir la grandeur commandée, ou grandeur réglée, à une valeur de référence malgré les variations des conditions extérieures : on parle donc de l'aspect de la régulation,

- Répondre à des changements d'objectif, cette grandeur doit passer d'une valeur à une autre en un temps donné, voire avec un profil de variation imposé : on parle d'un fonctionnement d'asservissement.

La Figure II-1 représente les éléments qui peuvent intervenir dans un tel système qu'on peut commander:

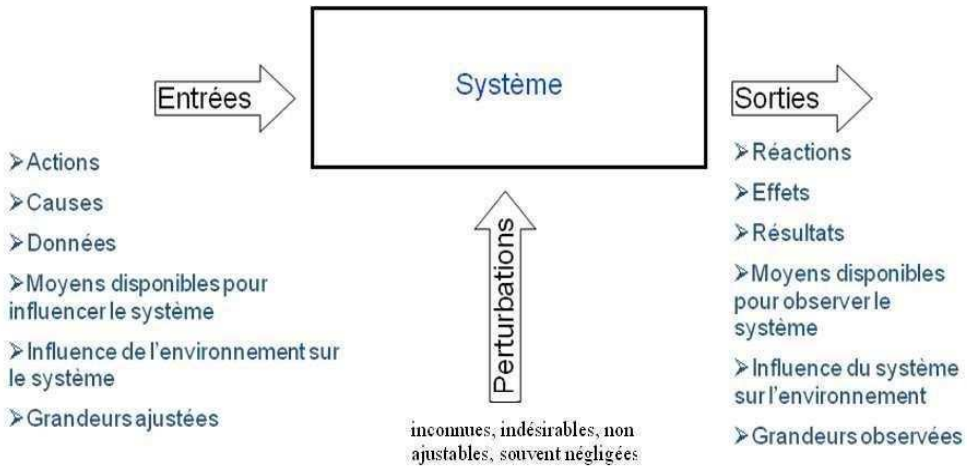


Fig. II-1 : Schéma d'un système à commander [19]

Un système de commande peut être manuel, c.à.d, avec un mode opérateur, ou automatique par un ensemble des méthodes permettant de conduire ce système sans intervention humaine mais sur la base des mesures liées à son comportement. En outre, le système peut être un système mono-variable SISO (Single Input, Single Output) ou multi-variables MIMO (Multiple Input, Multiple Output).

Un système SISO a un seul paramètre de sortie à mesurer et qui en rapport direct avec un seul paramètre d'entrée à commander, comme la température d'un thermostat dans une pièce ou dans un four**.**

Dans notre champ d'étude, ces notions sont utilisées chez un serveur Web. Par exemple pour contrôler le taux d'utilisation d'un serveur Web, cela nécessite un contrôle du paramètre Max clients. Supposons qu'on délimite le taux d'utilisation d'un serveur Web à un seuil de 66%, le paramètre de sortie à mesurer dans ce cas est le taux d'utilisation du CPU. Si le nombre de Max clients s'accroit, cela autorise plus de clients à se connecter, ce qui implique que le taux

d'utilisation du CPU s'accroit aussi, et l'inverse est vrai. Les perturbations ici sont l'exécution d'autres programmes que ceux du serveur qui peuvent se déclencher comme l'antivirus, ou pour le serveur lui même dans le cas des requêtes dynamiques CGI (Common Gateway Interface). Cette perturbation entraine le besoin de l'ajustement dynamique de Max clients suivant le contexte instantané.

Au contraire un système MIMO a plusieurs paramètres à commander. Dans ce type de système on possède plusieurs paramètres à contrôler et à achever.

2.3 La rétroaction

Les contrôles à rétroaction appelée aussi Feedback, sont utilisés pour réguler certains paramètres d'un système pour obtenir une performance désirée qui est la valeur en sortie, et qui doit disposer d'une unité d'évaluation ou de mesure afin de commander le système, appelée grandeur réglée. Tandis que les paramètres d'entrée déterminent les valeurs références à aboutir par le système, appelées consignes. La conduite d'une automobile décrit dans la Figure II-2 présente un exemple typique d'un système à rétroaction:

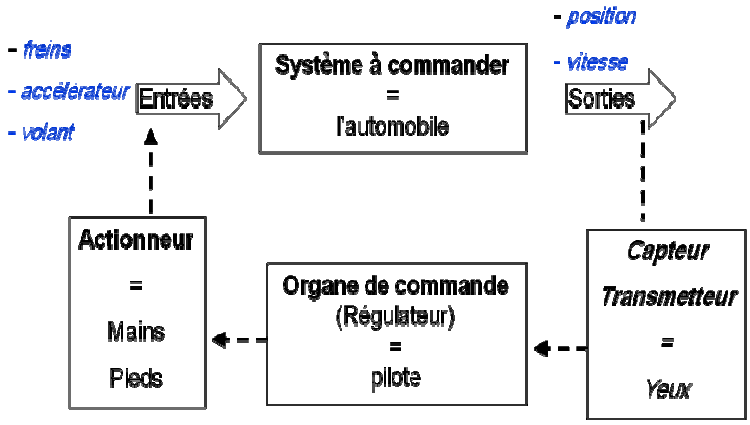


Fig. II-2 : Un système de commande manuelle MIMO [19]

Le flux d'information circulaire justifie l'utilisation du mot boucle fermée (closed-loop) dans la littérature pour désigner le système de contrôle Feedback. L'application de contrôle Feedback peut achever directement la valeur du paramètre de sortie désirée, en spécifiant sa valeur comme référence en entrée (reference input, dénommée consigne), et en évitant de manipuler le paramètre de contrôle d'entrée (contrôle input) pour aboutir à la valeur désirée de sortie.

La Figure II-3 décrit le schéma général avec les éléments du contrôle Feedback:

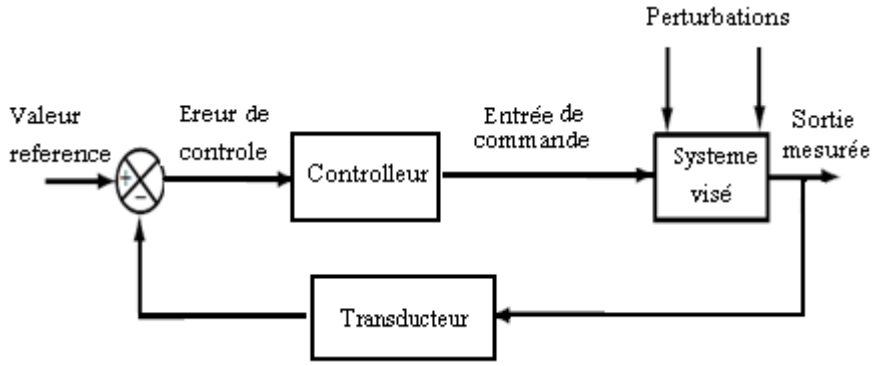


***Fig. II-3 : Les éléments de contrôle Feedback** [7]*

En effet les éléments qui décrivent le système Feedback sont :

- Erreur de commande: la différence entre la référence d'entrée (Référence Input) et la sortie mesurée (Measured Output).
- Entrée de commande: c'est le paramètre qui influe sur le comportement de système visé et peut être ajusté dynamiquement (comme le paramètre Max Clients dans le serveur HTTP Apache).
- Contrôleur : détermine le type de contrôle dont on a besoin pour achever à la valeur du (reference input). Le contrôleur calcule les valeurs de Control Input en basant sur les valeurs précédentes et actuelles du Control Error.
- Perturbations: qui ont l'effet de changement sur le control input influent sur le measured output.
- Sortie mésurée : c'est la grandeur réglée ou à mesurer du system visé, tel que le taux d'utilisation de CPU ou le temps de réponse d'une demande http.
- Valeur référence : c'est la consigne ou la valeur désirée à aboutir en sortie, par exemple le taux d'utilisation de CPU doit être 66%.
- Transducteur : qui transforme le measured output de telle sorte qu'on peut le comparer avec le reference input.

Dans une régulation en boucle fermée, une bonne partie des facteurs perturbateurs sont automatiquement compensés par la contre-réaction à travers le procédé Feedback. Autre

avantage, il n'est pas nécessaire de connaitre avec précision les lois, le comportement des différents composants de la boucle, et notamment du processus, bien que la connaissance des aspects statistiques et dynamiques des divers phénomènes rencontrés soit utile pour le choix des composants régulateurs.

Parmi les inconvénients d'une régulation en boucle fermée, il faut citer le fait que la précision et la fidélité de la régulation dépendent de la fidélité et de la précision sur les valeurs mesurées et sur la consigne. Autre inconvénient, sans doute le plus important, c'est le comportement dynamique de la boucle qui dépend des caractéristiques des différents composants de la boucle, notamment du processus. En fait, un mauvais choix de certains composants peut amener la boucle à entrer en oscillation **[21]**.

2.3.1 Les objectifs de la commande rétroactive

Les contrôleurs sont conçus pour certains buts prévus. Les objectifs les plus courants sont [**7**] :

- l'optimisation : Obtenir la "meilleure" valeur de la production mesurée. Par exemple, un contrôleur ajuste le MaxClients du serveur http Apache de façon à minimiser les temps de réponses.
- Le rejet de perturbation: Assurer que les perturbations agissant sur le système n'aient aucune influence sur la sortie mesurée. Par exemple, si la référence de taux d'utilisation d'un serveur est de 66%, lorsqu'une sauvegarde ou une analyse antivirus est exécutée sur un serveur Web, l'utilisation globale du système doit être maintenue à 66%.
- Le contrôle de la régularité : les contrôleurs sont utilisés pour veiller à ce que la sortie mesurée soit égale (ou proche) à la valeur du point de référence. La modulation est mise sur le point de référence s'il change fréquemment. Cela diffère du contrôle pour le rejet de perturbation à la mesure où ici nous nous concentrons sur les changements effectués sur le point de référence, et non sur les changements effectués par les perturbations sur les entrées.

L'objectif de contrôle Feedback est d'assurer que la valeur de paramètre de sortie soit égale ou au voisinage de la valeur de la référence d'entrée. Et de maintenir en plus cette valeur quelque soient les perturbations survenues.

2.3.2 Les propriétés de la rétroaction

Quatre propriétés de contrôle Feedback doivent être considérées dans l'étude d'un tel système. Ces propriétés sont les paramètres de performances du système à régulation automatique [**19**].

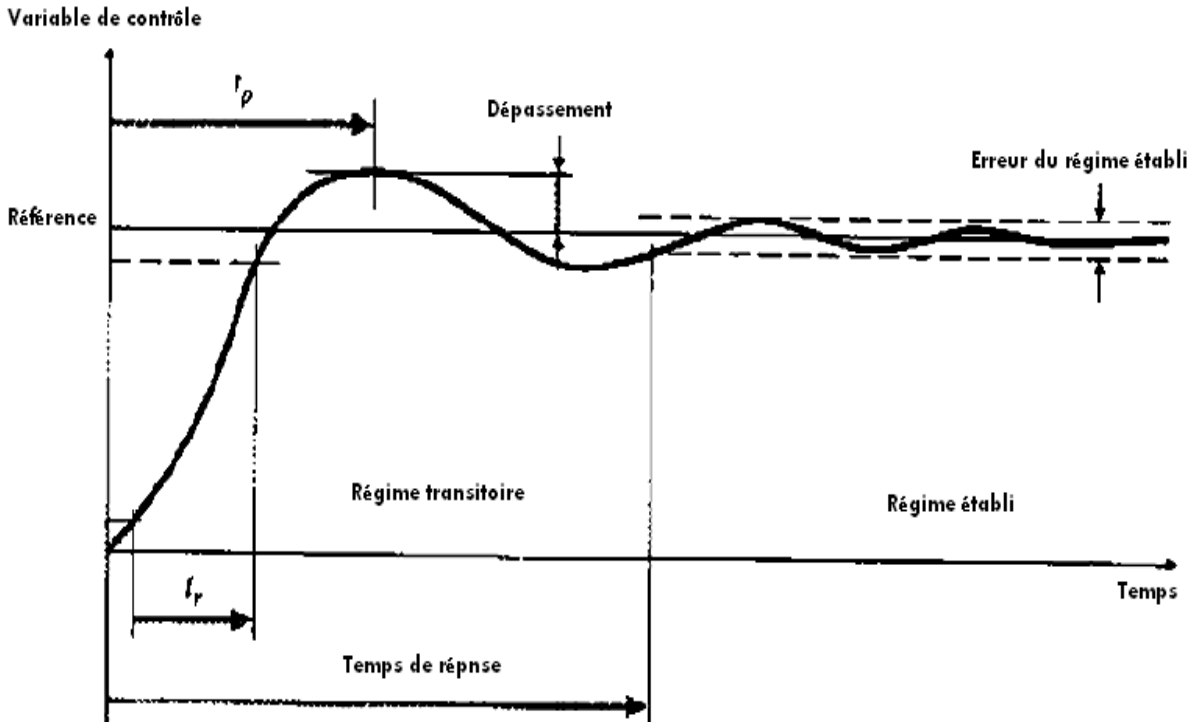


***Fig. II-4 : Les propriétés d'un système automatique** [7]*

La Figure II-4 ci dessus montre les propriétés d'un système automatique pouvant être expliquées par:

- Stabilités du système : le système doit être stable à la présence des perturbations survenues au système, car un système instable ne peut pas être utilisé pour une mission critique.

- Précision : le système est précis si la valeur mesurée converge vers la valeur consigne

- Convergence (settling time) : présente le temps écoulé pour que le système converge à la valeur référencée. Théoriquement, c'est le temps nécessaire pour que le régime transitoire ait totalement disparu [**19**].

- Dépassement (overshoot) : il est recommandé pour certain système d'achever ses objectifs sans un dépassement ennuyeux. Par exemple si la valeur maximum mesurée est Max et Yss est la valeur référencé, donc la valeur d'overshoot Mp=(Ymax / Yss)–1 doit être minimum.

2.4 Le Feedforward

Le contrôleur Feedforward, appelé aussi contrôleur à boucle ouverte, est utilisé pour ajuster le paramètre d'entrée contrôlé, en évitant l'utilisation de paramètre de sortie mesuré. Le contrôleur feedforward utilise le paramètre d'entrée référence et des fois en présence des perturbations pour ajuster le paramètre d'entrée contrôlé pour achever la valeur désirée de paramètre de sortie. Une régulation en boucle ouverte ne peut être mise en œuvre que si l'on connait la loi régissant le fonctionnement du processus (autrement dit, il faut connaitre la corrélation entre la valeur de sortie à mesurer et la valeur d'entrée à contrôler).

Au niveau des inconvénients, la régulation en boucle ouverte impose de connaitre la loi régissant le fonctionnement du processus, et il est très fréquent que l'on ne peut pas connaitre la loi en question. Autre inconvénient sérieux, il n'y a aucun moyen de contrôler, ou de compenser les erreurs, les dérives, les accidents qui peuvent intervenir à l'intérieur de la boucle. La Figure II-5 illustre ce modèle de contrôle.

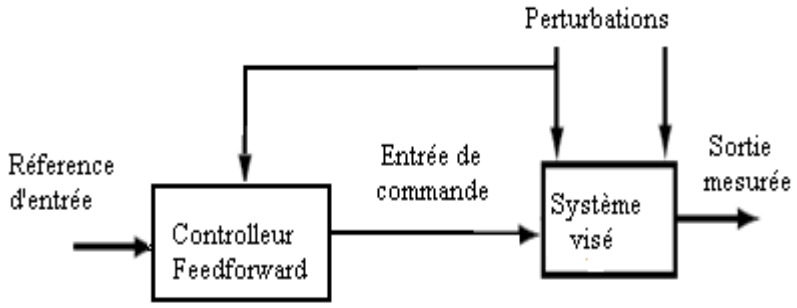


Fig. II-5 : Le modèle de contrôle en boucle ouverte [7]

Le Tableau II-1 ci-dessous, montre une comparaison entre les deux modes de contrôle :

Caractéristique	Feedforward	Feedback
Mettre stable un système instable	Non	Oui
Eviter l'utilisation de paramètre de sortie mesuré	Oui	Non
Simplicité du modèle utilisé	Non	Oui
L'adaptation à la perturbation	Non	Oui

Tab. II-1 : Une comparaison entre les deux modes de contrôle [7]

2.5 Les lois de commande élémentaires

Le principe de contrôle est conduit par des lois de commandes. La Figure II-6 présente le schéma de régulation en rétroaction :

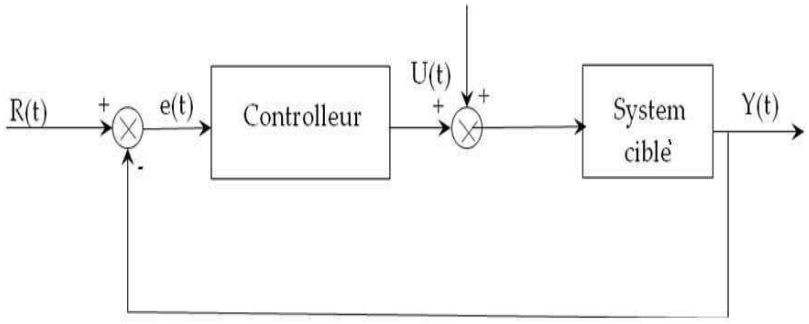


Fig. II-6: Schémas de régulateur rétroaction

En fait, on distingue deux types de lois de commande. Premièrement, celle qui ne présente que deux états appelée commande tout-ou-rien (on/off) semblable aux feux de signalisation par exemple:

- $U(t) = \begin{cases} U_{max} & \text{pour} \quad e(t) > 0 \\ U_{min} & \text{pour} \quad e(t) \leq 0 \end{cases}$ **…..…. E(II.1)**

Les deuxièmes types sont les régulateurs P, I et D (Proportionnel, Intégral, Dérivé), qui sont de plus en plus utilisés dans l'industrie, car ils permettent de régler les paramètres des

performances d'un processus, notamment les systèmes plus ou moins complexes, ayant un comportement qui peut être modélisé en deuxième ordre ou au voisinage de celui ci.

Par conséquent, les régulateurs P, I et D sont bien adaptés à la plupart des processus de type industriel, car d'une part ils sont relativement robustes par rapport aux variations des paramètres du procédé, et d'autre part il n'est pas trop exigé de connaitre la loi exacte gouvernant le procédé dans la boucle fermée comme celle de la boucle ouverte. On définit les différents contrôles de base P, I et D [**20]**:

- On appelle contrôle proportionnel (P): l'action de correction ou le signal de commande u(t) est à tout instant proportionnel au signal d'erreur e(t):

  $u(t)=K_P\, e(t)$ **…..…. E(II.2)**

  Où K_P est la constante de proportionnalité.
- On appelle contrôle par dérivation (D): l'action de correction où le signal de commande u(t) est obtenu par dérivation mathématique du signal d'erreur e(t):

  $u(t)=K_D\, de(t)/dt$ **…..…. E(II.3)**

  Où K_D est la constante de dérivation.
- On appelle contrôle par intégration (I): l'action de correction où le signal de commande u(t) est obtenu par intégration mathématique du signal d'erreur e(t):

  $u(t)=K_I \int e(t)dt$ **…..…. E(II.4)**

  Ou K_I est la constante d'intégrité.
- On appelle contrôle PD, PI, PID les actions dues aux corrections que l'on peut mettre en œuvre par combinaison des contrôles proportionnels, par intégration et par dérivation.

Pour effectuer un choix judicieux, il faut connaitre les effets des différentes actions PID: proportionnelle, intégrale et dérivée.

2.5.1 Action proportionnelle

L'action proportionnelle P crée un signal de commande u(t) proportionnel au signal d'erreur e(t). Le facteur de proportionnalité K_P est son gain statique. L'action proportionnelle permet de jouer sur la vitesse de réponse du procédé et d'améliorer la précision. Plus le gain est élevé, plus la réponse s'accélère, plus l'erreur statique diminue (en proportionnel pur), mais

plus la stabilité se dégrade. Donc, Il faut trouver un bon compromis entre la précision et la stabilité [**22]**.

2.5.2 Action intégrale

L'action intégrale I crée un signal u(t) qui est l'intégrale du signal d'erreur e(t). L'action intégrale permet d'annuler l'erreur statique (écart entre la mesure et la consigne). Plus le gain d'intégrité K_I est élevé, plus la réponse s'accélère et plus la stabilité se dégrade. Il faut également trouver un bon compromis entre la vitesse et la stabilité [**21]**.

2.5.3 Action dérivée

L'action dérivée D crée un signal de commande u(t) qui est la dérivée du signal d'erreur e(t). L'action dérivée est anticipatrice. En effet, elle ajoute un terme qui tient compte de la vitesse de variation de l'écart, ce qui permet d'anticiper en accélérant la réponse du processus lorsque l'écart s'accroît et en le ralentissant lorsque l'écart diminue. Plus le gain de dériveté K_D est élevé, plus la réponse s'accélère. Là encore, il faut trouver un bon compromis entre vitesse et stabilité [**21]**.

Notons que l'action dérivée ne peut être utilisée seule. On la fait appel, lorsque le signal de commande u doit être particulièrement efficace [**22]**.

La Figure II-7 montre les effets de choix des valeurs des gains sur les régulateurs. Dans la Figure II-7.a, l'erreur statique apparait.

Dans la Figure II-7.b, le $K_I=1/T_I$ car il est en fonction du temps. Et il en va de même pour $K_D=T_D$ dans la Figure II-7.c

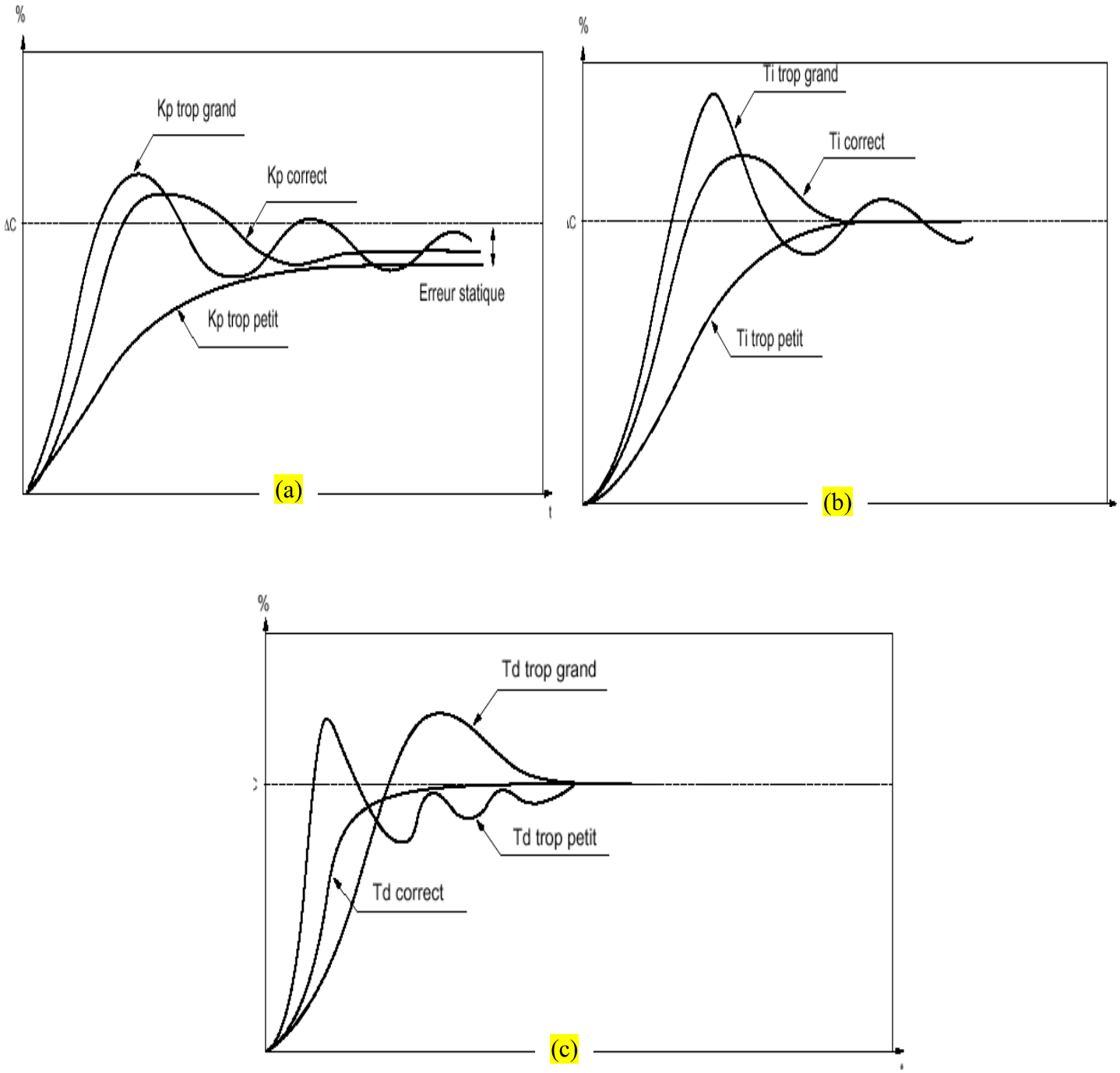


Fig. II-7: Les graphes des régulateurs P, I et D vis-à-vis des valeurs de gains [21].

2.5.4 Régulateurs PI, PD et PID

En effet des caractéristiques des régulateurs décrites précédemment, on fait appel à la combinaison des ceux-ci pour aboutir à un système rapide, précis et stable, donc chaque type de correcteur P,I et D a pour fonction:

- Le terme P assure la rapidité.
- Le terme I annule l'erreur statique.
- Le terme D améliore la stabilité.

Donc la loi de la commande de type PID employé en parallèle peut s'écrire:

$$U(t) = K_P \cdot E(t) + \frac{1}{T_I} \cdot \int_0^t E(t)dt + T_d \cdot \frac{dE(t)}{dt} \quad \text{..... E(II.5)}$$

La Figure II-8 ci dessous présente les procédés des différentes combinaisons P, I et D.

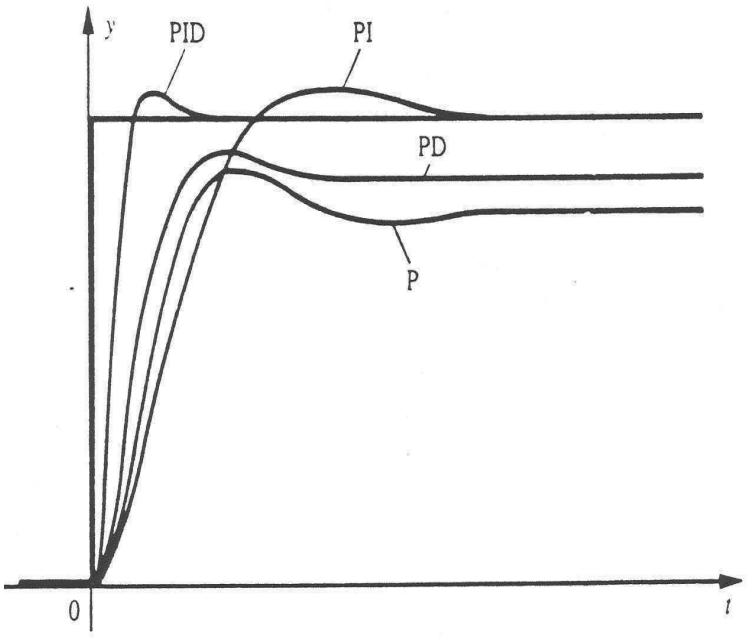


Fig. II-8: Les procédés PI, PD et PID vis-à-vis du régulateur P [21]

2.6 Méthodes de calcul des gains des régulateurs

Rappelons que le rôle d'un régulateur est de maintenir le gradeur régulée à une valeur consigne malgré la présence des perturbations. Donc, on doit disposer d'un modèle mathématique du processus à contrôler qui peut être décrit sous la forme d'un ensemble d'équations mathématiques.

En résolvant ces équations, il est alors possible de savoir comment va réagir le processus, suite à une modification d'une de ses entrées ou à l'arrivée d'une perturbation externe. A partir de là, en connaissant ce comportement, il est possible de définir les caractéristiques du régulateur qui permettra de commander au plus près le processus [**21].**

Malheureusement, il y a un fossé de la théorie à la pratique. Les descriptions mathématiques des processus sont en effet souvent très complexes et exigent de grandes compétences **[7]**.

Pour calculer les paramètres des régulateurs, d'autres techniques doivent être utilisées. Il en existe plusieurs, toutes basées sur des essais expérimentaux, parmi elles, on peut citer deux méthodes [**21]**:

– La méthode par approches successives : cette technique consiste à modifier les actions (sur le processus) et à observer les effets pour la mesure enregistrée, jusqu'à obtenir la réponse optimale. On règle dans l'ordre l'action proportionnelle, l'action dérivée puis l'action intégrale.

Du fait de sa simplicité, c'est une méthode très utilisée ; toutefois, son application devient longue sur les processus à grande inertie. Son principal avantage est de ne pas nécessiter de connaissance approfondie du processus et du régulateur.

– La méthode de Ziegler et Nichols : elle nécessite l'observation de la réponse du processus et la connaissance de la structure du régulateur. C'est une méthode qui permet de calculer les coefficients des actions PID, sans la détermination des paramètres du processus.

La méthode de Ziegler & Nichols **[23],** consiste à mettre en oscillations entretenues la boucle de régulation. A partir du gain critique Grc en action proportionnelle seule qui permit d'obtenir une oscillation non amortie, et de la période T_{OSC} de cette oscillation comme il est indiqué à la Figure II-9, il est possible de choisir les paramètres de réglage du régulateur. Cette méthode est utilisable pour les procédés stables et instables.

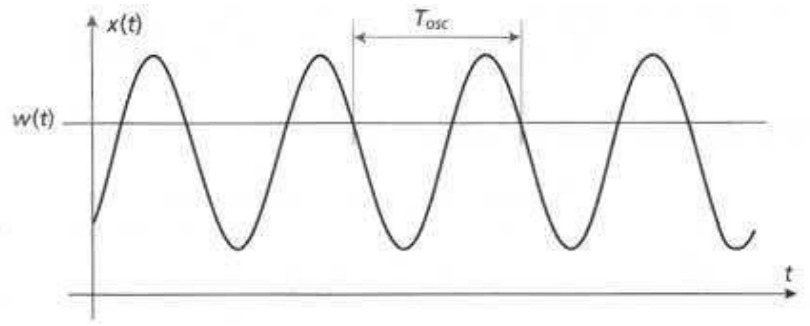


Fig. II-9: Mise en oscillation d'un système [23]

Le tableau ci-dessous montre les valeurs préconisées par Ziegler et Nichols d'un PID parallèle.

Modes de régul. / Actions	P	PI	PID
G_r	$\frac{G_{rc}}{2}$	$\frac{G_{rc}}{2,2}$	$\frac{G_{rc}}{1,7}$
T_i	Maxi	$\frac{2.T_{osc}}{G_{rc}}$	$\frac{0,85.T_{osc}}{G_{rc}}$
T_d	0	0	$\frac{T_{osc}.G_{rc}}{13,3}$

Tab. II-2 : Valeurs préconisées des Coefficients PID par Ziegler et Nichols

Notons que les valeurs préconisées par Ziegler et Nichols peuvent conduire à un dépassement qui arrive jusqu'à 40%. Il faut donc modifier légèrement ces réglages **[23]**.

2.7 Conclusion

La régulation automatique est une technique, qui permet d'offrir des outils à la prise de contrôle d'un tel système en vue d'imposer un comportement désiré. Ce chapitre présente les notions courantes utilisées dans les commandes automatiques, ainsi que les différents types d'actions de commandes élémentaires P, I et D. Nous préférons traiter d'autres notions au niveau des appendices, tels que la construction de modèles et la transformation en Z et la fonction de transfert.

CHAPITRE III
LES MODELES D'ALLOCATION DES SERVICES DANS LES SERVEURS WEB

3.1 Introduction

Les études d'optimisation des serveurs Web s'intéressent particulièrement à incorporer le concept de la qualité de service (QoS), pour permettre la différenciation entre les différents trafics des services Web.

Comme il a été énoncé à l'introduction, les modèles à queues représentent la masse essentielle pour gérer les différentes politiques d'allocation du serveur Web.

Pour cela, nous présentons dans ce chapitre, en premier lieu quelques modèles des systèmes à files d'attente classiques, ensuite les modèles des serveurs avec contrôle Feedback qui existent dans les littératures.

Le délai moyen d'un flux dans un système à files d'attente (on l'a nommé le temps de connexion) est la moyenne des délais des paquets constituant ce flux. Le délai d'un paquet est le temps de transit du paquet entre l'entrée et la sortie du système. Il s'agit donc du temps d'attente du paquet dans la file d'attente et du temps de service du paquet dans le serveur.

Le taux de perte d'un flux est le rapport du nombre de paquets du flux rejetés par le système sur le nombre total de paquets constituant le flux. Nous considérons que les paquets ne sont rejetés que dans le cas où la taille de la file est limitée et qu'il ne reste plus de place pour accepter un paquet.

3.2 Les modèles des files d'attente classiques

Nous présentons quelques modèles de systèmes de files d'attente classiques, pour lesquels nous présentons les évaluations analytiques des délais moyens des flux dans ces systèmes, le nombre moyen des requêtes ainsi que le taux de perte des paquets s'il existe dans le système.

3.2.1 Généralités sur les files d'attente

3.2.1.1 Notations de Kendall

Communément, la notation utilisée pour définir un système de files d'attente est la notation de Kendall. Sa forme complète est *A/S/m/K/P/D* et sa forme abrégée est *A/S/m* **[24]**. Une file d'attente simple est caractérisée par 5 paramètres:

- un processus d'arrivée (*A*) des paquets,
- un processus de service (*S*) des paquets,
- un certain nombre de serveurs (*m*),
- une taille du système (*K*), qui est le nombre de places dans le système (dans la file d'attente et dans le serveur), par défaut est ∞,
- une taille de la population (*P*), par défaut ∞,
- une discipline de service (*D*), par défaut *FIFO.*

Les processus d'arrivée (*A*) et de service (*S*) peuvent être :

- *M* : loi exponentielle. Cette loi est dite sans mémoire. Elle caractérise les phénomènes aléatoires indépendants des événements passés et des phénomènes précédents.
- *D*: loi déterministe (constante). Cette loi caractérise les flux constants (c'est-à-dire les flux dont les paquets sont émis à intervalles constants pour le processus d'arrivée), ou caractérise le service à durées constantes pour le processus de service. Les phénomènes aléatoires sont négligés.
- *GI* : loi générale (avec indépendance des événements). Celle-ci regroupe les lois à événements indépendants. Les lois *M* et *D* sont des lois particulières de type *GI*.
- *G*: loi générale (avec dépendance des évènements). Celle-ci regroupe les lois à événements dépendants. La loi *GI* est une loi particulière de type *G*.

3.2.1.2 Notations utilisées pour les évaluations

Nous présentons ici les notations utilisées pour évaluer le temps d'attente dans une file et dans le système, ainsi que le taux de perte dans le système. Les évaluations sont toutes établies dans le cadre du régime permanent **[24]**. Soient :

- $\boldsymbol{\lambda i}$**:** le taux d'arrivée des paquets dans le système de files, ou encore le débit du flux, pour un flux de classe i,
  $1 / \lambda i$: intervalle moyen séparant deux arrivées de même classe.
- $\boldsymbol{\mu i}$: le taux de service d'un serveur d'un paquet de classe i,
  $1 / \mu i$: durée moyenne de service.

Dans un système de files d'attente, nous cherchons à mesurer :

- Le nombre moyen de paquets présents dans le système (file d'attente + serveur) : N,
- Le nombre moyen de paquets dans la file d'attente : Q,
- Le taux d'utilisation du serveur : U,
- Le temps moyen T de séjour dans le système (file + service) : $T \geq 1/\mu$.
- Le temps moyen de service : S,
- Le temps moyen d'attente dans la file : W,
- Le taux de perte dans le système s'il existe: P,

Pour un système stable en régime stationnaire où le taux d'arrivée doit être inferieur au taux de service $\lambda < \mu$,

On a: $T = W + S$

$N = Q + mU$

Où m est le nombre de serveurs.

Du fait que $m = 1$, on obtient :

$$N = Q + U.$$

Dans le cas d'un système ne génère pas de perte, le taux d'utilisation du serveur est:

$U=\rho=\lambda/\mu$,

Où ρ est l'intensité (ou le débit) du système.

3.2.1.3 La formule de Little

Pour un système stable en régime stationnaire, la formule de Little exprime le fait que le nombre moyen de paquets présents dans le système (respectivement, le nombre moyen de paquets présents dans la file) dépend du taux d'arrivée des paquets et du temps moyen de séjour dans le système (resp. du temps moyen d'attente dans la file) **[25]**, Donc :

- Au niveau du système :

$N = \lambda T$ …….***E(III.1)***

Où T: Le temps moyen de séjour dans le système.

- Au niveau de la file d'attente :

$Q = \lambda W$ …….***E(III.2)***

Où W: Le temps moyen d'attente dans la file.

3.2.1.4 Indicateurs de qualité de service et évaluations des files

Le lien est étroit entre les évaluations des indicateurs de qualité de service et les évaluations des paramètres présentés précédemment **[24]**:

- ☒ Le délai *T* correspond au temps moyen de séjour d'un paquet dans le système (temps moyen de séjour dans la file plus le temps de service).
- ☒ Le taux de perte *P* correspond au taux de non acceptation des paquets dans la file lorsque la file est pleine (dans le cas où la file a une taille limitée). Dans le cas où l'on ne limite pas la taille de la file, le taux de perte est nul.

3.2.2 Le modèle classique Best effort

Le modèle Best effort représente le serveur Internet traditionnel ayant une seule file d'attente et qui utilise la politique best effort et traite les demandes de manière FIFO "First In, First Out". C'est la technique par défaut dans tous les serveurs Web. L'inconvénient majeur est que ce type de serveur ne fait pas la différence entre les services. De ce fait, les services sensibles au temps sont retardés, ce qui apporte une mauvaise qualité de service pour ce type de serveur Web. L'avantage de ce modèle est qu'il est plus facile à mettre en place **[26].**

3.2.2.1 Le modèle de file d'attente à taille infinie (M/M/1)

3.2.2.1.1 Hypothèses

Une file M/M/1 est caractérisée par :

- les lois d'arrivée et de service sont poissoniènnes,
- la file est de taille infinie,
- il existe une seule classe de service en FIFO.
- Le système est considéré comme stable à partir du moment où il y a moins d'arrivées que de départs (ou services) *: $\rho < 1$.*

L'effet que la file d'attente soit de taille infinie conduit à un taux de perte nul. La perte des paquets apparaît lorsqu'un nouveau paquet arrive au système mais ne peut y être stocké à cause de la saturation de la file d'attente. La Figure III-1 présente la file M/M/1:

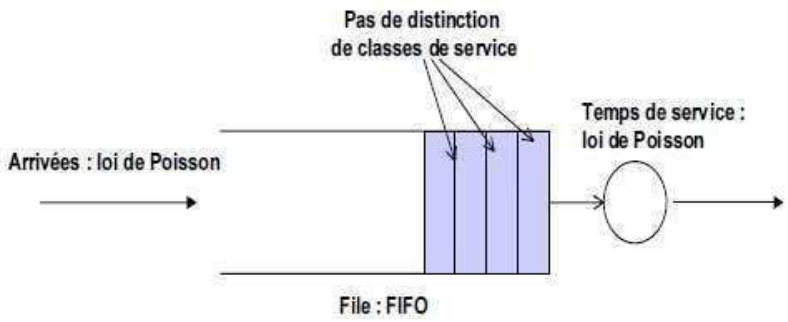


Fig. III-1: Le modèle de la file d'attente Best effort infinie M/M/1.

3.2.2.1.2 Evaluations analytiques

Nous rappelons quelques caractéristiques du système de file M/M/1 [**27**], [**28**]:

- L'intensité du trafic est représentée par:

$$\rho = \lambda / \mu \quad \text{.....} \textbf{E(III.3).}$$

- Le taux d'utilisation du serveur est :

$$U = \rho = \lambda/\mu. \quad \text{.....} \textbf{E(III.4).}$$

- Le nombre moyen de paquets dans le système vaut:

$$N = \rho / (1 - \rho) \quad \text{.....} \textbf{E(III.5).}$$

- Le nombre moyen de paquets dans la file est :

$$Q = N - U = \rho^2 / (1 - \rho) \text{.....} \textbf{E(III.6).}$$

- Le temps moyen de séjour dans le système (resp. dans la file) se calcule à l'aide de la formule de Little l'équation E(III.1).

  Donc le délai moyen dans un système M/M/1 vaut :

$$T = \frac{1}{\mu - \lambda} \quad \text{.....} \textbf{E(III.7).}$$

3.2.2.1.3 Evaluation expérimentale

Nous montrons la correspondance de validité avérée de l'évaluation analytique de la charge moyenne de la file M/M/1 à celle obtenue avec 10000 simulations de Monte-Carlo en régime stationnaire à travers des graphes [**28**]. Les Figures III-2 et III-3 illustrent la comparaison de l'évaluation analytique de la charge moyenne d'une file M/M/1 par rapport à la charge fournie par les séries de simulations. Par extension, l'évaluation analytique du délai moyen représente bien la réalité.

Les paramètres des tests sont indiqués dans le Tableau III-1 ci dessous:

Tests	λ	μ	P
Test1	0.1	1	0.1
Test2	0.5	1	0.5
Test3	0.3 / 0.6	1	0.3 /0.6

Tab. III-1 : Paramètres des tests de simulation du modèle M/M/1

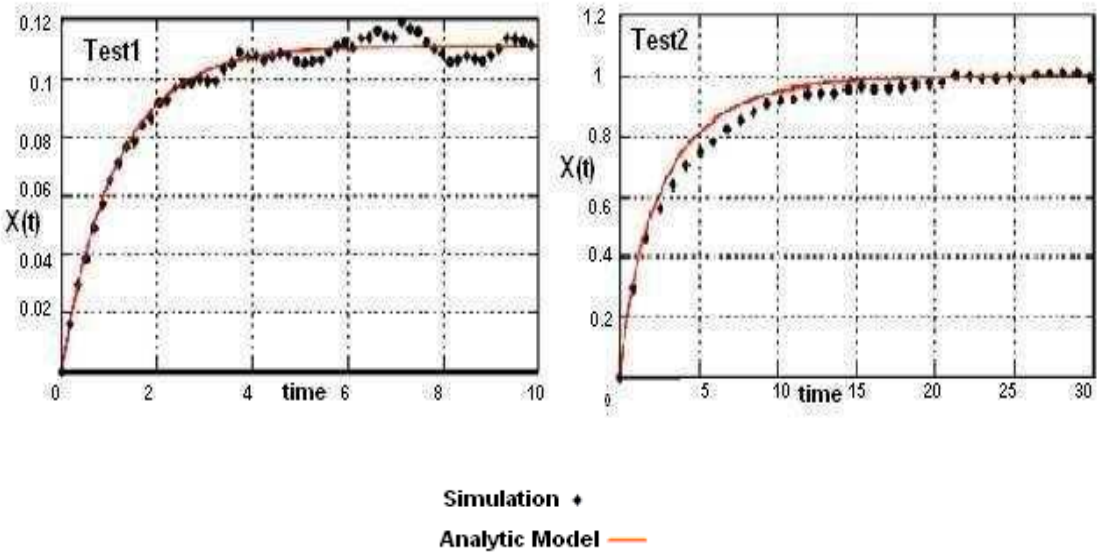


Fig. III-2 : L'évaluation par simulation évènementielle et l'évaluation analytique de la charge de file (M/M/1) [28].

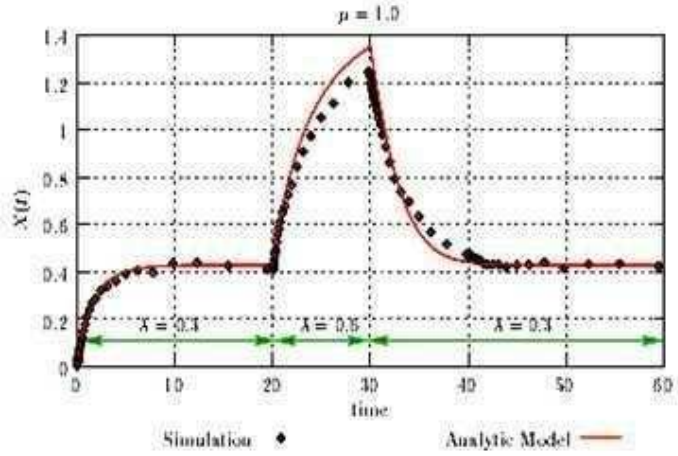


Fig. III-3: L'évaluation par simulation évènementielle et l'évaluation analytique de la charge des files en cas de variation de trafic [28].

3.2.2.2 Le modèle de file d'attente à taille finie (M/M/1/K)

Ce type de modèle de files d'attente ne se diffère du modèle précédent que par la taille limitée de la file d'attente possédant K-1 places, donc le système peut engendrer des pertes quand un paquet arrive et qu'il n'y a plus de place dans la file d'attente, par conséquent le paquet sera détruit.

3.2.2.2.1 Hypothèses

Les paramètres de la file d'attente M/M/1/K sont :

- les lois d'arrivée et de service sont poissoniènnes,
- la file d'attente est de taille finie K
- il existe une seule classe de service en FIFO.

Les paquets ont un taux d'arrivée de valeur λ et un taux de service de valeur μ. La taille de la file est limitée. Ainsi pour que le système ne s'engorge pas et reste stable à partir du moment où il y a moins d'arrivées que de départs (services) : $\lambda < \mu$. La Figure III-4 est une représentation du modèle du file d'attente M/M/1/K.

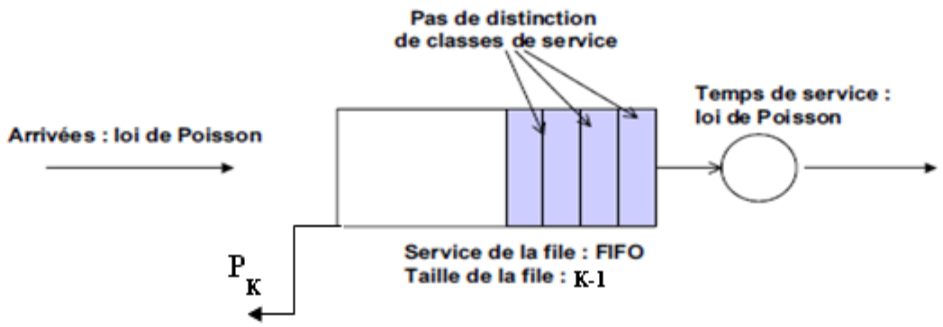


Fig. III-4: Le modèle de la file d'attente à taille finie M/M/1/K.

3.2.2.2.2 Evaluation analytique

Les évaluations que nous présentons sont détaillées dans **[29]** .

-L'intensité du trafic vaut toujours: $\rho = \lambda/\mu$. …….***E(III.8)***

-Le taux de perte est le même pour tous les flux, à savoir :

$$P = \begin{cases} 1 - \dfrac{1-\rho^{K}}{1-\rho^{K+1}} & \textbf{Si} \quad \rho \neq 1 \\ \dfrac{1}{K+1} & \textbf{Si} \quad \rho = 1 \end{cases} \qquad \text{…….}\textbf{\textit{E(III.9)}}$$

-Le trafic qui transite effectivement dans le système est diminué du taux de perte, soit : $\lambda (1 - P)$, tel que $1 - P$ le taux de conservation du flux.

-Le taux d'utilisation U du serveur n'est plus égal à: $\rho = \lambda/\mu$ mais vaut: $U = \lambda(1 - P)/\mu$.

- La charge moyenne N vaut :

$$N = \begin{cases} \dfrac{K\rho^{K+2} - (K+1)\rho^{K+1} + \rho}{1 - \rho - \rho^{K+1} + \rho^{K+2}} & \textbf{Si} \quad \rho \neq 1 \\ \dfrac{K}{2} & \textbf{Si} \quad \rho = 1 \end{cases} \qquad \text{…….}\textbf{\textit{E(III.10)}}$$

-Le délai moyen T dans le système est calculé en appliquant l'équation E(III.1) de Little:

$$T = N / \lambda\ (1 - P)$$

Donc le délai moyen d'un paquet (non perdu) vaut :

$$T = \begin{cases} \dfrac{1}{\mu(1-\rho)} - \dfrac{K\rho^{K}}{\mu(1-\rho^{K})} & \textbf{Si} \quad \rho \neq 1 \\ \dfrac{K+1}{2\lambda} & \textbf{Si} \quad \rho \neq 1 \end{cases} \qquad \text{…….}\textbf{\textit{E(III.11)}}$$

Notons que, quand N tend vers l'infini et que $\rho < 1$, nous retrouvons bien les résultats d'une file M/M/1.

3.2.2.2.3 Evaluation expérimentale

Comme précédemment, les graphiques comparatifs de l´évaluation analytique et la simulation de la charge moyenne de la file M/M/1/N sont aussi les évaluations obtenues des simulations de Monte-Carlo (10000 réalisations). Tandis que la Figures III-5 illustre la comparaison de l´évaluation analytique de la charge moyenne d'une file M/M/1/N par rapport à la charge fournie par la série de simulations, la Figure III-6 montre l'effet de la capacité de la file d'attente K sur la charge, le temps de réponse, et le taux d'utilisation avec un taux de services μ=4 et les différents taux d'arrivés.

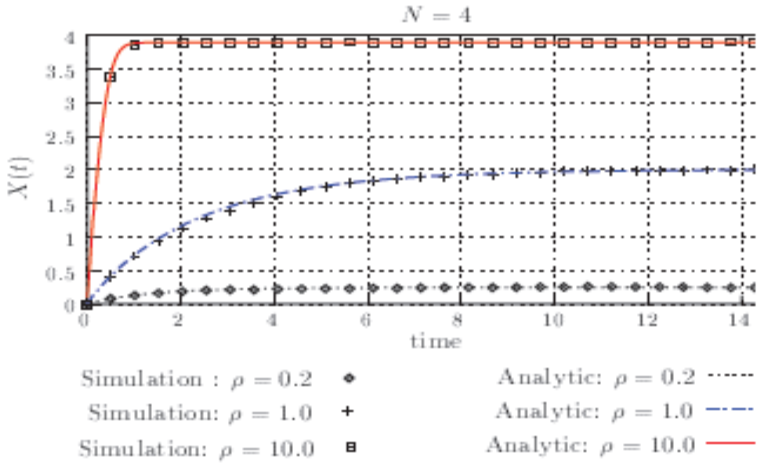


Fig. III-5 : l'évaluation par simulation évènementielle et l´évaluation analytique de la charge de file (file M/M/1/K) [28].

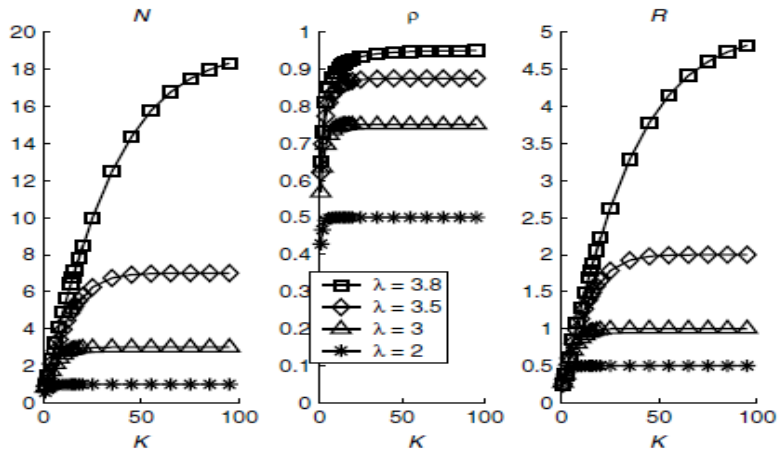


Fig. III- 6: L'effet du paramètre K sur la charge N, le temps de réponse R, et le taux d'utilisation ρ dans une file d'attente M/M/1/K [7].

3.2.3 Les modèles classiques à services différenciés

Le Système à services différenciés DiffServ détient le pivot des recherches d'optimisation des qualités de service dans les serveurs Web. Dans ce type de système, le serveur est en opposition du modèle Best effort, il possède plusieurs classes, toute classe a sa propre file d'attente avec une certaine priorité. Le serveur favorise les classes ayant les plus hautes priorités.

Dans les serveurs DiffServ on peut traiter les requêtes avec différentes manières, parmi elles on cite:

- ☒ *-Gestion avec priorité absolue.*
- ☒ *-Gestion avec priorité relative (pondérée).*
- ☒ *-Gestion hybride.*
- ☒ *-Gestion avec files d'attente à seuils.*

3.2.3.1 Le modèle de Gestion avec priorité absolue (M/M/1/∞/Priorité)

Le serveur permet de classer les demandes selon leurs priorités. La Figure III-7 représente un système à file d'attente M/M/1/∞/Priorité :

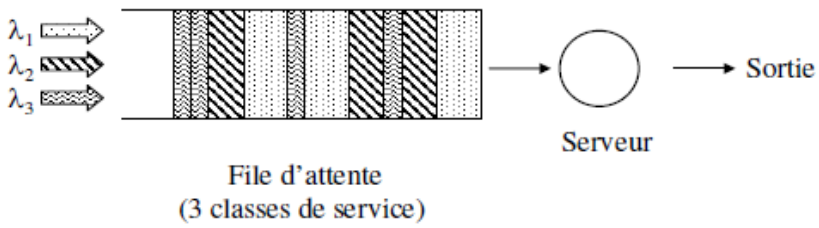


***Fig. III-7: Le modèle de Gestion avec priorité* M/M/1/∞/Priorité**

Cela peut s'apparenter à un système multi files $M^K/M^K/1/\infty$/Priorité, semblablement au Figure III-8, où il y a une file pour chaque classe et chaque file est gérée en FIFO.

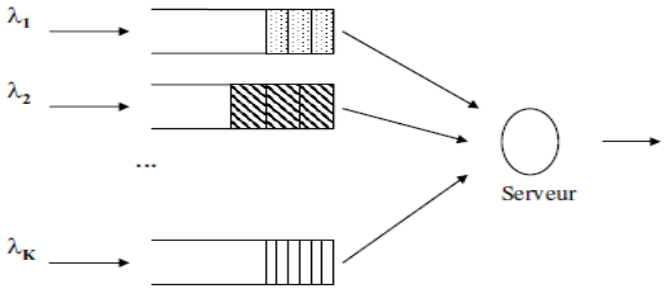


Fig. III-8: Le modèle de Gestion avec priorité $M^K/M^K/1/\infty$/Priorité

Le serveur traite en priorité la première file (de classe supérieure) en FIFO puis, lorsque celle-ci est vide, la deuxième file, etc. Si au cours du service de la j^{ieme} file, un paquet arrive dans une des files 1 à j−1 (donc de priorité plus élevée), le serveur finit le service en cours (pas de préemption) et arrête de servir la file j pour revenir à la file de classe supérieure dans laquelle s'est présenté le paquet.

3.2.3.1.1 Hypothèses

Les paramètres du système sont :

- la loi d'arrivée des paquets et la loi de service sont poissoniènnes,
- les files sont à des capacités illimitées,
- il existe plusieurs classes de service et ces classes définissent des niveaux de priorité.

Si tous les paquets dans la file appartiennent à la même classe, ils sont servis en FIFO. Toutefois, si un paquet de classe supérieure arrive, il sera mis en tête de file et sera servi après que le service en cours soit fini, mais avant les autres paquets présents dans la file de classe moins prioritaire.

3.2.3.1.2 Evaluation analytique

Ce modèle a été présenté par G .Pujolle et E.Gelenbe dans **[27]** sous le nom M/GI/1 avec priorité, la loi d'arrivée des paquets est poissonienne et la loi de service est en générale indépendante (cela peut être par exemple une loi M ou D),

Lorsque la file n'est pas vide à l'arrivée d'un paquet de classe *i*, son temps d'attente moyen Wi se décompose de la manière suivante :

- Wi0 : les temps de fin de service du paquet en cours de service (peu importe sa classe), ce temps est aussi appelé Résiduel Time Life (RTL).
- Wi1 : les temps de service des paquets plus ou également prioritaires au paquet de classe *i* qui se situent avant lui dans la file à son arrivée.
- Wi2 : les temps de service des paquets de classes supérieures arrivant dans la file pendant l'attente du paquet de classe *i*.

Le temps d'attente moyen du paquet de classe *i* est donc la somme de ces temps d'attente :

$$Wi = Wi0 + Wi1 + Wi2$$

Pour une classe *i*, soit ρi la probabilité que le serveur soit occupé. Nous avons : $\rho_i = \sum_{j=1}^{i} \rho_j$

Posons λi est le taux d'arrivée pour une classe *i*, μ_i^{-1} et σ_j^2 sont la moyenne et le carré du coefficient de variation de la distribution du temps de service des paquets de classe *i*. Le temps moyen d'attente d'un client de classe *i* dans sa file d'attente vaut :

$$W_i = \frac{\sum_{j=1}^{k} \lambda_j \sigma^2}{2(1-\rho_{i-1})(1-\rho_i)} \qquad \text{.....E(III.12)}$$

Donc le temps d'attente moyen dans le système (ou délai moyen) pour un paquet de classe *i* est la somme du temps moyen de service $1/\mu_i$ et du temps moyen d'attente dans la file *Wi*, soit:

$$T_i = \frac{1}{\mu_i} + \frac{\sum_{j=1}^{k} \lambda_j \sigma^2}{2(1-\rho_{i-1})(1-\rho_i)} \qquad \text{.....E(III.13)}$$

3.2.3.1.3 Evaluations expérimentales

L'équipe RST (Réseaux et Systèmes de Télécommunications) du LAAS-CNRS a développé un simulateur de réseaux pour l´évaluation événementielle de divers systèmes de files d'attentes **[24].**

Les paramètres des tests sont indiqués dans le Tableau III-2 ci dessous:

Tests	Nb échantillons	Num Classe	λ	μ	Taille paquet	ρ
Test 1	200000	—	—	—	—	—
		1	10	100	10	0.1
		2	100	200	5	0.5
Test 2	500000	—	—	—	—	—
		1	4	20	50	0.2
		2	2	10	100	0.2
Test 3	500000	—	—	—	—	—
		1	2	20	50	0.1
		2	1	10	100	0.1

Tab. III-2 : Paramètres des tests de simulation du modèle $M^2/M^2/1/\infty$/Priorité

La Figure III-9 présente les graphiques comparatifs de l´évaluation analytique et de l'évaluation par simulation de la charge moyenne pour le modèle file $M^2/M^2/1/\infty$/Priorité à deux classes.

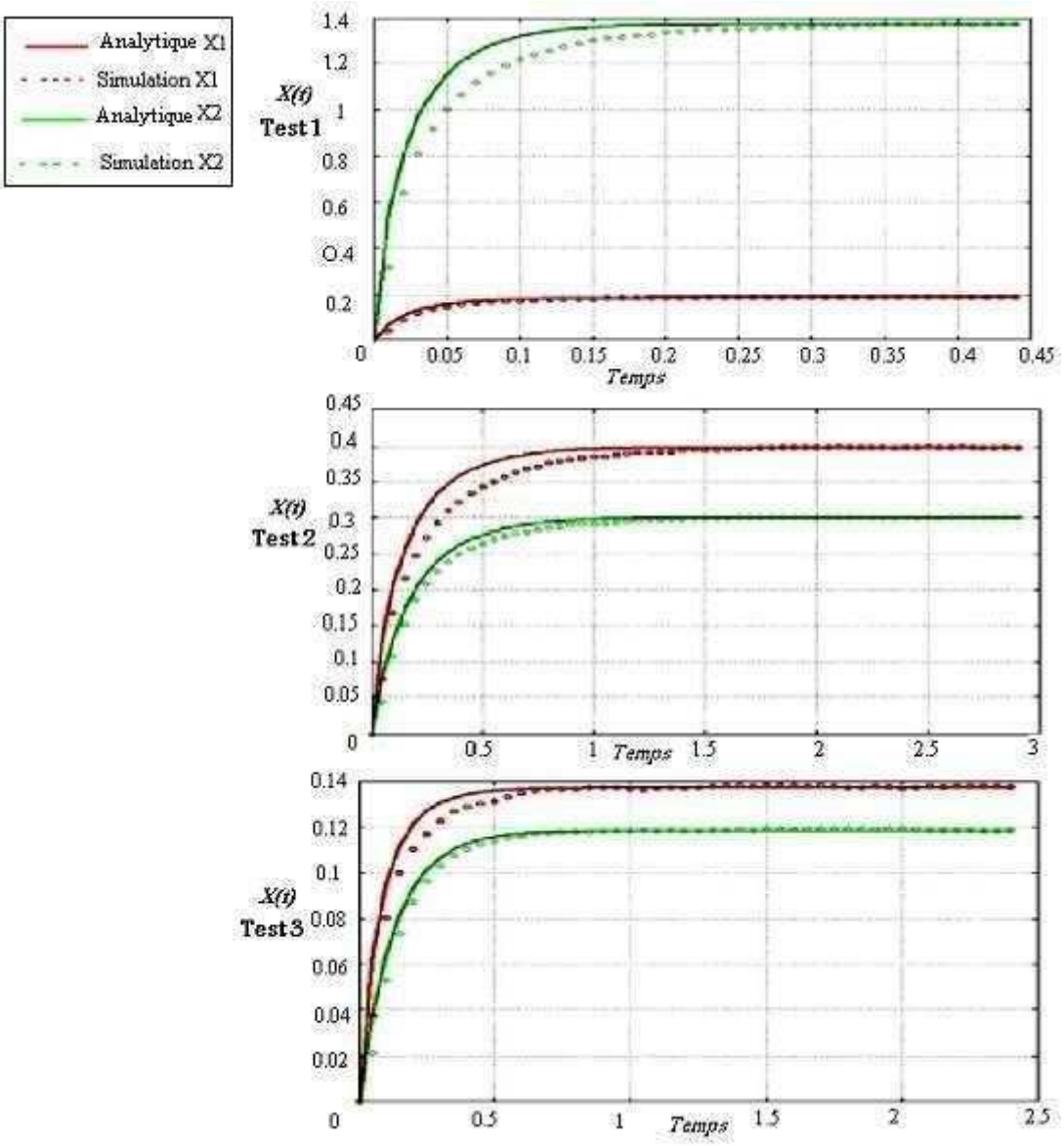


Fig. III-9 : Comparaison de l´évaluation par simulation événementielle et de l'évaluation analytique de la charge des files (file $M^2/M^2/1/\infty$/Priorité) [24].

3.2.3.2 Le modèle de Gestion avec priorité relative (pondérée)

Dans le même champ de système DiffServ, ce type de gestion est connu dans les réseaux informatique sous le nom WFQ- Weighted Fair Queuing – qui signifie la discipline de file d'attente tentant d'obtenir une utilisation optimale de la capacité en pondérant les priorités de passage selon la charge associée aux tâches **[30].** Les demandes qui surviennent, sont alors classées en premiers temps dans une file d'attente qui correspond à son poids. Par conséquent, les demandes des classes sont servies de façon équilibrée en suivant leur poids respectifs.

Cette discipline de service est connue aussi sous le nom de GPS (Generalized Processor Sharing) et PGPS qui est la version «paquet» de GPS et qui correspond exactement à WFQ. La discipline de service GPS est un modèle théorique de multiplexage de flux qui se charge de transmettre les paquets bit par bit en fonction de leurs poids.

La Figure III-10 montre le déroulement de la discipline de service GPS d'un système à trois files d'attente WFQ, et les poids de chaque file sont respectivement: 0.5, 0.25, 0.25, le serveur répartira le service à 50% pour la file 1, à 25% pour la file 2 et à 25% pour la file 3 quand les trois files sont simultanément occupées.

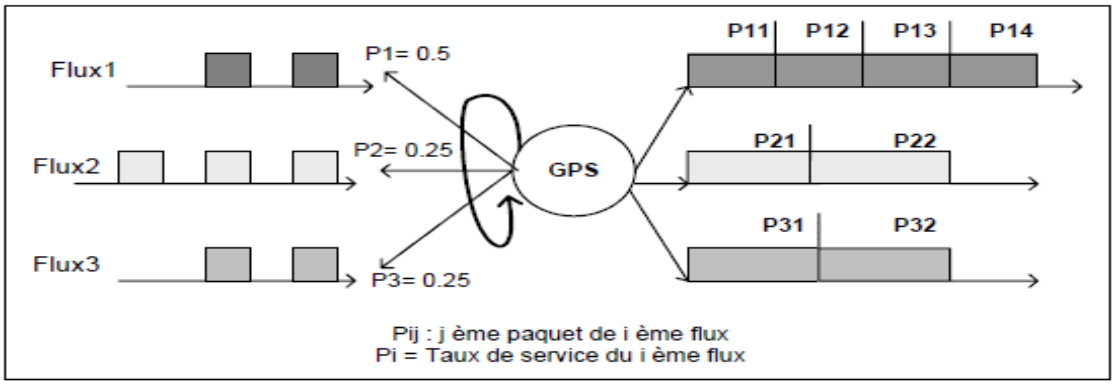


Fig. III-10 : La discipline de service GPS

Uns serveur GPS conserve cette discipline de service tant qu'il reste des requêtes dans les files. Si une file est vide, alors le serveur servira la prochaine file et répartira proportionnellement le taux de service affecté à la file vide sur les autres files pleines. En effet, si une file se vide temporairement, alors les autres files seront servies pendant ce temps

à un taux plus élevé. Cette propriété de GPS est particulièrement adéquate et va permettre d'optimiser les ressources.

Plus formellement, un serveur GPS est caractérisé par les taux de service affectés, $\{\varphi_{1,}.....\varphi_n\}$, à chacun des *N* flux actifs. Un flux est dit actif dans l'intervalle de temps [t1, t2], si la file d'attente de ce flux est toujours pleine durant cet intervalle, Avec : $\sum_{j=1...N}^{n} \varphi_j = 1$

Supposons que le serveur opère à capacité fixée C. Notons par Wi (t1, t2) la quantité de service allouée au flux *i* dans l'intervalle de temps [t1, t2]. Le serveur GPS est définit tel que:

$$\frac{W_i(t_1,t_2)}{W_j(t_1,t_2)} \geq \frac{\varphi_i}{\varphi_j} \qquad j=1....N$$

Pour tous les flux actifs *i* ayant des paquets en attente et sur tout intervalle [t1, t2], en faisant une somme sur $j = 1...N$ pour une valeur de i fixée, nous avons :

$$W_i(t_1,t_2)\cdot\varphi_j \geq W_j(t_1,t_2)\cdot\varphi_i$$

On obtient: $W_i(t_1,t_2)\cdot\sum_{j=1}^{n}\varphi_j \geq \sum_{j=1}^{n} W_j(t_1,t_2)\cdot\varphi_i$

D'où : $C = \sum_{j=1}^{n} W_j(t_1,t_2)$

On a : $W_i(t_1,t_2) \geq \frac{\varphi}{\sum_{j=1}^{n}\varphi_j}\cdot C\ (t_1,t_2)$

Ainsi, le flux actif reçoit un service minimum égal à W_i tant qu'il a des paquets en attente de service c'est le service garanti au $i^{ème}$ flux est :

$$W_i = \frac{\varphi}{\sum_{j=1}^{n}\varphi_j} C\ (t_1,t_2) \quad .....\textbf{\textit{E(III.14)}}$$

L'ordonnancement GPS, et son émulation WFQ, lorsqu'une des pondérations tend vers un, les deux systèmes ne tendent pas vers le même comportement. En effet, GPS étant idéal, lorsque le poids d'une classe tend vers un, celle-ci est complètement isolée (similaire à une file M/M/1) alors que pour WFQ, du fait de la granularité des paquets, elle tendra à être similaire

à une file prioritaire **[31]**. En conséquence le system WFQ appui sur les flux qui ont subi un taux d'arrivée élevée malgré qu'ils n'aient pas des poids adéquats.

Dans la section suivante, on présente un système d'ordonnancement hybride d'une file prioritaire avec un ensemble des files WFQ, dont on trouve une évaluation analytique et expérimentale d'un système WFQ.

3.2.3.3 Le modèle de Gestion hybride (Exemple : Mk/Glk/1/∞/Priorité + WFQ)

Le serveur peut utiliser une gestion mixte en combinant les trois méthodes, par exemple une file d'attente pour les priorités absolues, et certaines files d'attentes avec des priorités relatives. Dans ce cas le serveur favorise la première file d'attente, dés que la file est vide, le serveur traite les requêtes des files d'attente avec des priorités relatives à l'égard de leurs poids.

3.2.3.3.1 Hypothèses :

Ici nous présentons dans la Figure III-11 un modèle à 5 files **[24]**, dont la première file d'accueil *File0* sert à limiter le nombre total des requêtes, une autre est une file d'attente *File1* prioritaire - Priority Queueing(PQ)(i.e. les flux de cette file sont traités avant ceux des autres files) et les 3 autres files sont gérées par la politique WFQ.

La politique WFQ octroie à chaque file i un poids αi, la somme des poids valant 1. Si la file prioritaire est vide, le serveur va servir chaque file en fonction de ce poids. Si un paquet arrive dans la file prioritaire alors que le serveur traite une des files WFQ, le serveur finit son service puis traite la file prioritaire, qu'il reste ou non des paquets dans une des files WFQ. Le taux de pertes est commun pour toutes les classes de services.

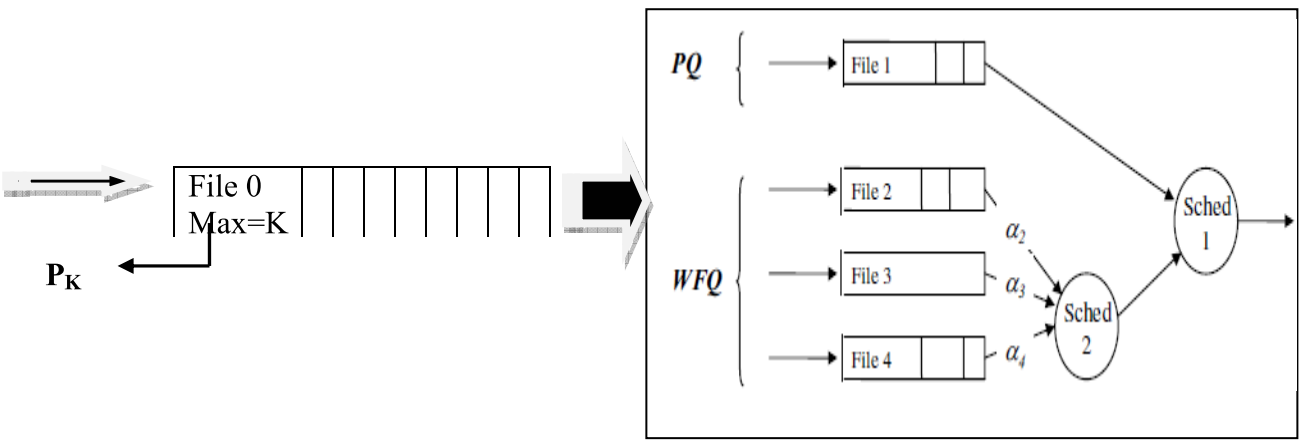


Fig. III-11:Un modèle d'une Gestion hybride

3.2.3.3.2 Evaluation analytique :

Les évaluations suivantes sont basées sur les évaluations de l'équipe du LAAS-CNRS dans le cadre du projet OPIUM (Optimisation de la Planification des Infrastructures de réseaUx Mobiles) labellisé par le RNRT (Réseau National de Recherche en Télécommunications) et ont été présentées dans **[31]**.

Le délai dans la première file d'accueil :

La file d'accueil se comporte comme si elle était la seule dans le système :

-Le taux de trafic est : $\rho = \lambda / \mu$, avec : $\lambda = \lambda_1 + \lambda_2 + \lambda_3 + \lambda_4$

- Le taux d'occupation du serveur par ce type de paquet est : $\rho = \lambda(1-P) / \mu$
- Le taux de perte P est calculé par l'équation E(III-9).
- Le nombre moyen de paquets est le même que dans l'équation E(III-10).
- Le délai moyen d'un paquet dans le système est bien sur l'équation E(III-11).

Le délai dans une file prioritaire :

La file prioritaire, peut aussi être considérée comme si elle était la seule dans le système dans le cas où le service est préemptif.

- Le taux d'occupation du serveur par ce type de paquet est : $\rho_1 = \lambda_1 (1-P) / \mu$ ……***E(III.15)***
- Le nombre moyen de paquets prioritaires vaut : $N = \rho_1 / (1 - \rho_1)$. ……***E(III.16***
- Le délai moyen d'un paquet prioritaire dans le système vaut :

$$T_1 \approx \frac{1}{\mu - \lambda_1(1-P)} \qquad \text{…….}\boldsymbol{E(III.17)}$$

Où P est le taux de perte.

Le délai dans une file WFQ :

Le délai des flux dans une file WFQ est influencé par la présence de flux dans la file prioritaire mais aussi, de manière pondérée par les poids φ_i, par les flux présents dans les files WFQ.

Notons, tout d'abord, pour tous les $i = 2, 3, 4$, le taux d'occupation total: $\rho = \sum_{i=2..4} \rho_i$

Le taux d'occupation du lien par du trafic venant des files WFQ dépend de la bande passante restante après le service de la file prioritaire 1, soit ($\mu - \lambda 1$). Donc :

$$\forall i = 2,3,4 \qquad \rho_i = \frac{\lambda_i (1-P)}{\mu - \lambda_1} \qquad \text{……}\boldsymbol{E(III.18)}$$

L'évaluation de la charge moyenne de la file dépend du poids de la file, du taux d'occupation par ce flux, mais aussi des taux d'occupation par les flux des autres files WFQ. Ce qui donne :

$$Q_i \approx \varphi_i . \frac{\rho_i}{1-\rho_i} + (1-\varphi_i)\left(\frac{\rho}{1-\rho} - \frac{\sum_{j=1..N, i\neq i} \rho_j}{1-\sum_{j=1..N, i\neq i} \rho_j}\right) \qquad \text{……E(III.19)}$$

Où : $\frac{\rho_i}{1-\rho_i}$: est le taux d'occupation probable par le trafic venant des files *i.*

$\frac{\rho}{1-\rho}$: est le taux d'occupation global par le trafic venant des files WFQ

$\frac{\sum_{j=1..N, i\neq i} \rho_j}{1-\sum_{j=1..N, i\neq i} \rho_j}$: est le taux d'occupation par le trafic venant des files WFQ autre que *i.*

Le délai dans la file WFQ *k* est déduit de la formule de Little E(III.2).

3.2.3.3.3 Evaluations expérimentales

Les courbes présentées sont fournies par le LAAS-CNRS **[24]**. Les tests dans la Figure III-12 illustrent la comparaison de l'évaluation analytique de la charge moyenne d'une file M^k/GI^k */1/∞/Priorité + WFQ* (une file prioritaire et trois files WFQ) par rapport à la charge moyenne fournie par une série de simulations. Les paramètres des tests sont indiqués dans le Tableau III-3 ci dessous :

Tests	Type de file	λ	μ	Poids WFQ
Test 1	Prio	0.1	1	—
	WFQ	0.05	1	0.8
	WFQ	0.02	1	0.1
	WFQ	0.03	1	0.1
Test 2	Prio	—	—	—
	WFQ	0.2	1	0.5
	WFQ	0.02	1	0.25
	WFQ	0.01	1	0.25
Test 3	Prio	0.2	1	—
	WFQ	0.15	1	0.5
	WFQ	0.1	1	0.333
	WFQ	0.05	1	0.166

Tab. III-3 : Paramètres des tests de simulation du modèle M^k/GI^k /1/∞/Priorité + WFQ

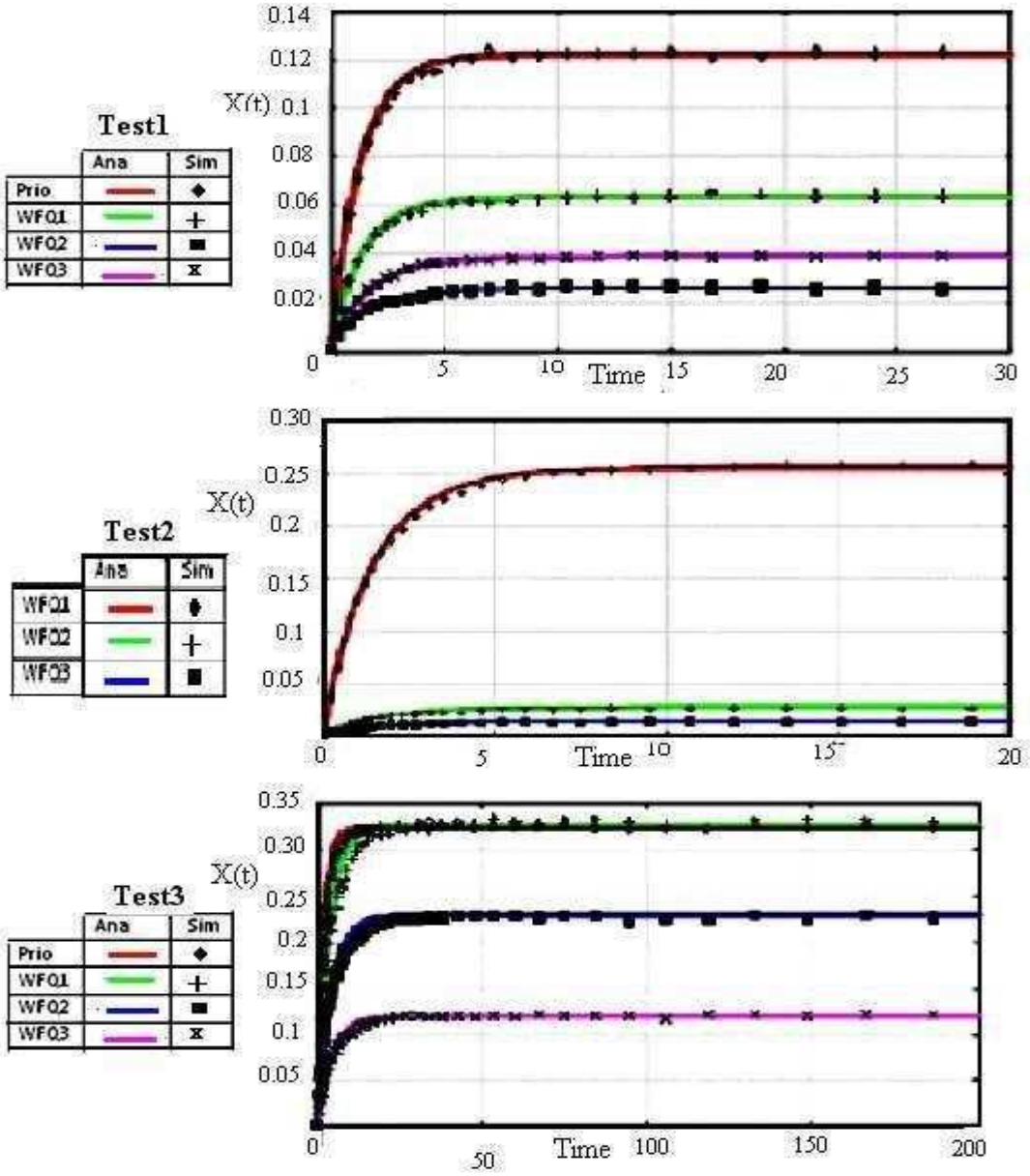


Fig. III-12: L'évaluation par simulation évènementielle vis-à-vis de l'évaluation analytique de la charge des files du modèle ($M^4/GI^4/1/\infty$/Priorité + WFQ) **[24].**

3.2.3.4 Le modèle à files d'attente avec seuils

Un autre modèle de système de différenciation des services (Modèle à multiple QoS Classes), est un système multi-modèle à files d'attente avec seuils. Pour ce système multi-modèle, les files d'attente ayant des priorités décroissantes, de la file d'attente 1 jusqu' à la file d'attente K. En outre, chaque file d'attente a un point de seuil qui n'est pas la limite de la file d'attente, mais son rôle est d'annoncer que la file d'attente devient prioritaire, si ce seuil est atteint **[4]**.

3.2.3.4.1 Exemple

La Figure III.13 présente un exemple de ce modèle avec K files d'attente où K - 1 d'entre elles sont des classes QoS servies avec leurs priorités et la $K^{ième}$ file est préservée aux services sans priorités qui vont être traitées par le modèle Best efforts.

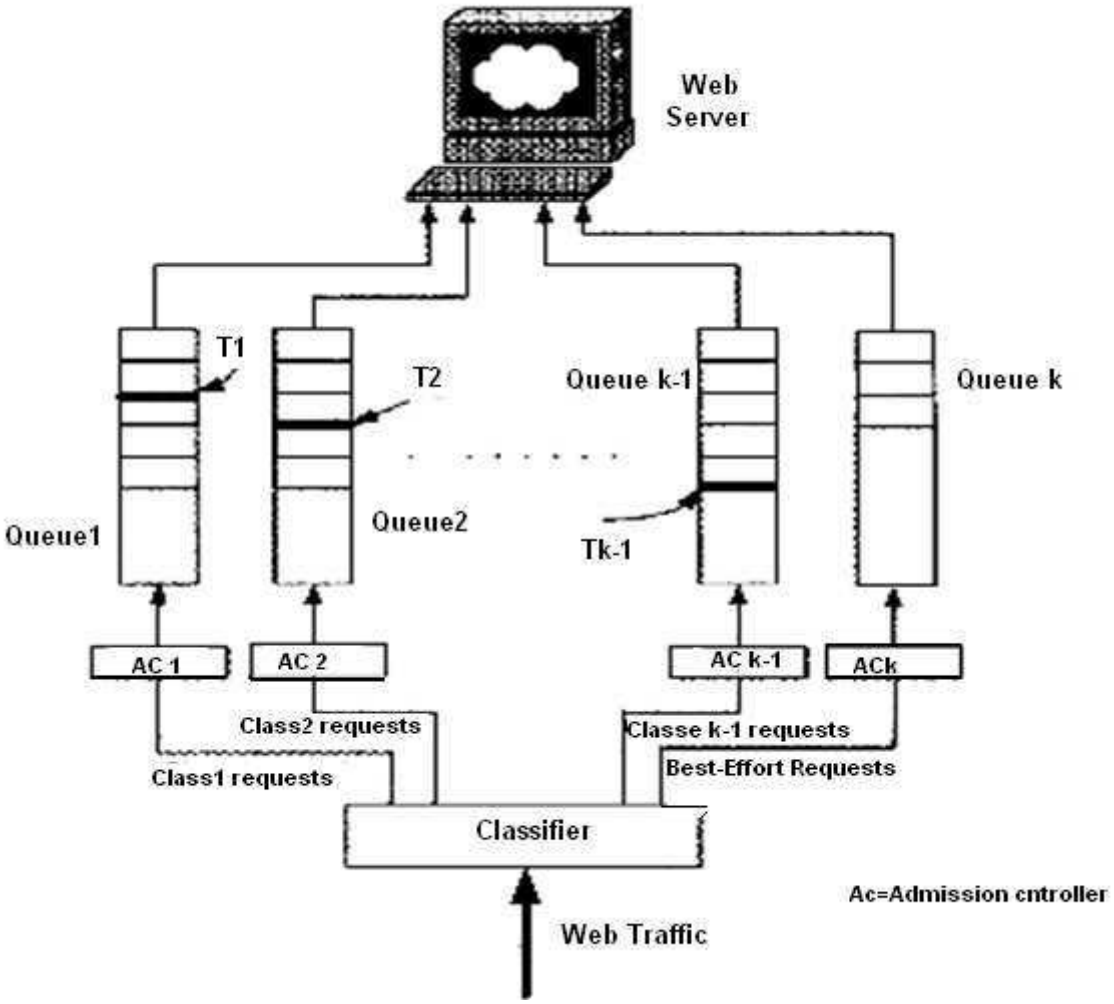


Fig. III-13: Modèle multi QoS classes du serveur Web [4]

Il y a donc, K-1 seuils (Tl, • • •, TK-l) attribués à chaque file d'attente sauf la $K^{ième}$ file d'attente qui est sans seuil. Il est à noter que les tailles des files d'attentes peuvent être illimitées. Les contrôles d'admission sont employés pour ne pas violer les taux d'arrivées définies à priori par le serveur.

Si le système est vide (le serveur à l'état de repos) et une nouvelle demande arrive, le serveur lui attribut la file d'attente correspondante et commence à la servir. Maintenant, supposons qu'à un moment le serveur est en service, sur une file d'attente i, 1<= i<= k. Le serveur traite actuellement une demande de classe i (la file d'attente i). Bien que ce service soit en cours, de nouvelles demandes arrivent dans l'une des files d'attente et par la suite l'une de la QoS files d'attente peut atteindre ou franchir le seuil.

Lorsque le serveur complète la demande actuelle, deux situations peuvent survenir :

Dans le premier cas, la $i^{ème}$ queue pourrait avoir atteint son seuil (on note que le Best effort de la file d'attente, c'est-à-dire pour i = K, ne sera pas dans une telle situation, car il n'a pas de seuil fixé). Le serveur va continuer à servir dans la file d'attente i seulement si aucune des files d'attente plus prioritaire n'atteint son seuil de points. Si une seule de ces files d'attente à priorité plus élevée a atteint son seuil, le serveur va changer à cette file d'attente. Mais, s'il y en a deux ou plusieurs de ces files d'attente, le serveur va passer à la priorité la plus élevée parmi elles.

Dans l'autre cas, la $i^{ème}$ file n'a pas atteint son seuil de point d'achèvement où le service a eu lieu dans la file d'attente du meilleur effort. Le serveur restera dans la file d'attente actuelle, sauf que si l'un des Q0S files d'attente a atteint son seuil ou que la file d'attente courante est devenue vide.

La Figure III-14 illustre l'algorithme adopté par le serveur Web pour cette politique du service:

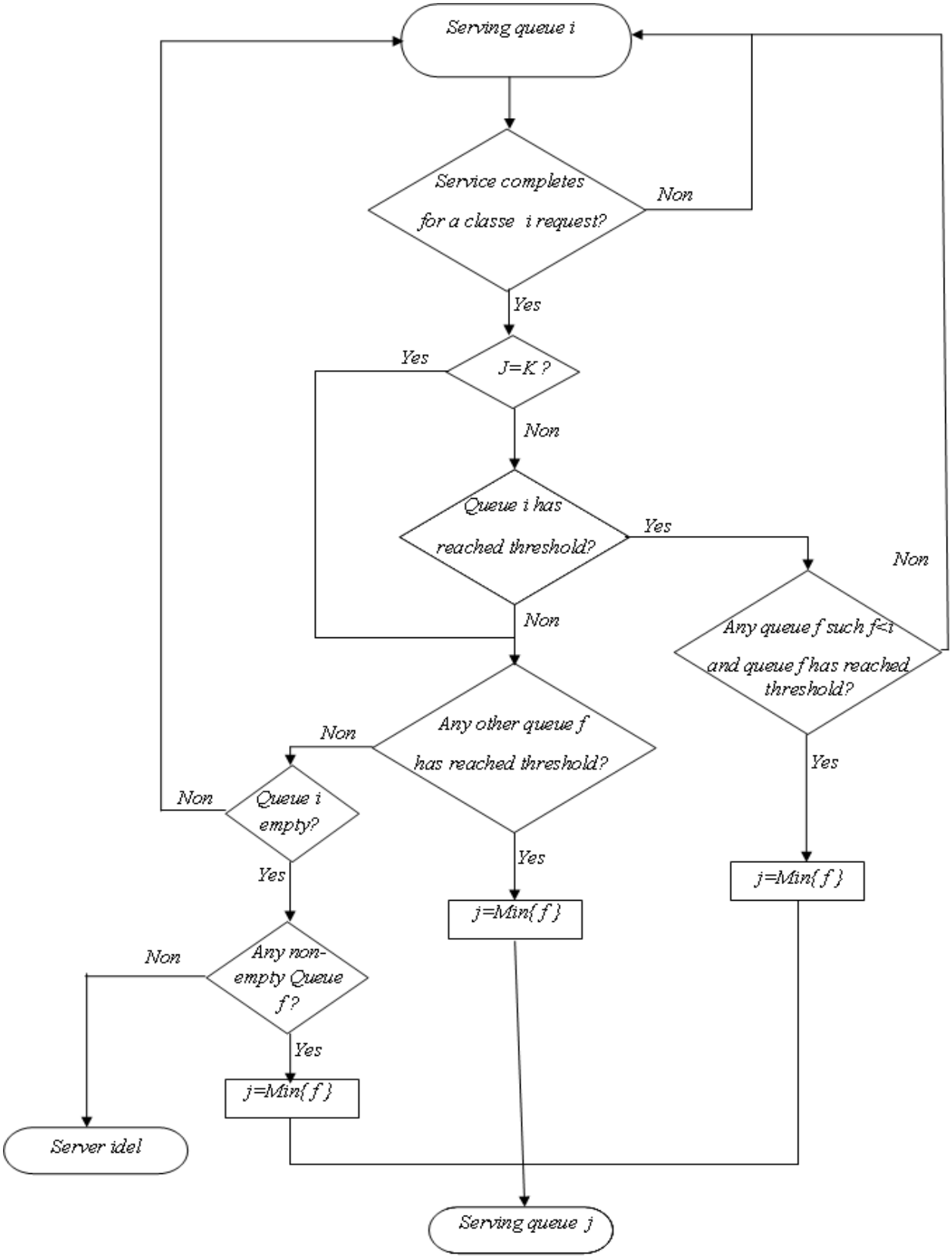


Fig. III-14: La politique d'allocation des services des files à seuils [4]

3.3 Les modèles des serveurs avec contrôle

Les approches traditionnelles de la performance ne sont pas efficaces pour une large classe de systèmes. Toutefois, la conception et la mise en œuvre des logiciels nécessitent des bases d'analyses efficientes pour assurer la performance dans les systèmes informatiques.

Récemment, la théorie du contrôle a été identifiée comme un fondement théorique de la performance et offre des solutions parfaites aux applications logicielles complexes, telles que la planification en temps réel, les serveurs Web, les gestionnaires de stockage, le contrôle de puissance de CPU, et le routage dans les réseaux informatiques. Ces solutions offrent une interface géniale entre le calcul de performance et le contrôle de système, et automatise des nombreuses parties de la rétroaction au but d'obtenir une qualité de service (QoS) satisfaisante dans ces applications.

Dans le monde des serveurs Web, le système de contrôle à commande automatique est utilisé pour aboutir d'une part à une exploitation parfaite des ressources machines, et d'autre part à une meilleure qualité de service.

3.3.1 Problématique des contrôles du QoS dans les serveurs

L'utilisation des contrôles pour l'optimisation des systèmes informatiques, particulièrement les serveurs Web, est de plus en plus adoptée.

On peut citer trois types de problèmes d'optimisation par contrôle qui sont généralement des domaines de recherches au niveau des serveurs Web **[7] :**

- Le premier problème est le DiffServ, c'est là où notre étude se concentre, en établissant des différents niveaux de services pour achever les demandes survenues au serveur Web.
- Le deuxième problème est l'amélioration de temps de réponse en prise en compte la configuration du système (nombre maximum de clients).
- Le troisième problème est le contrôle d'utilisation des ressources (Mémoire et CPU) selon une façon non excessive en raison des considérations de fiabilité (par exemple, certains logiciels deviennent fragiles à de lourdes charges), ou à cause de la conception d'un système qui met une réserve de capacité qu'il ne peut franchir.

3.3.2 Architecture générale de contrôle dans les serveurs Web

Comme on l'a indiqué antérieurement, les contrôles sont utilisés dans le but d'optimisation chez les serveurs. Dans cette section on présente une architecture générale d'un système d'optimisation et quelques exemples où le contrôle feedback est employé, en commençant par des systèmes non DiffServ. Les systèmes DiffServ sont traités à part dans la section suivante.

Le serveur Web reçoit les demandes des clients. Le serveur traite ces demandes et génère les réponses convenables, ces dernières sont par la suite envoyées aux clients.

On vise à contrôler le serveur afin que le système se comporte de la façon désirée. En effet, le système d'optimisation par contrôle du serveur est conçu par la jonction d'un optimiseur au système visé.

De ce fait, le système visé doit être contrôlé par une configuration des paramètres dont elle est dynamiquement changée par l'optimiseur suivant les mesures survenues au moyen de la chaine de rétroaction.

L'architecture générale d'un système de contrôle d'optimisation est indiquée dans la Figure **III-15**.

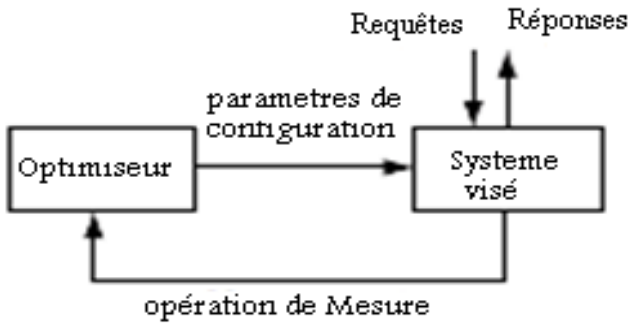


Fig. III-15 : Architecture générale d'un optimiseur à rétroaction [15]

3.3.3 Le contrôle Feedback des systèmes non DiffServ :

Un exemple typique de contrôle de systèmes traditionnels, où on ne fait pas la différenciation des services, est proposé par Hellerstein et all **[7], [15].**

Dans un serveur HTTP Apache, deux paramètres ont de l'effet sur la configuration du serveur et par conséquent sur sa performance.
Le premier est le MaxClients qui représente le nombre de processus à traiter par le serveur, si le MaxClients est assez grand il entraine une violation en besoin des ressources particulièrement en CPU et en mémoire.

Le deuxième paramètre est le KeepAlive, si le KeepAlive est court, le besoin est intensif en ressources CPU. Dans le cas contraire, le CPU est dans un état de sous exploitation, donc la performance est minimale à cause d'attentes inutiles des processus de serveur. Donc on a besoin d'un contrôle MIMO pour le serveur avec les entrées contrôlées MaxClients et KeepAlive comme l'indique la Figure III-16**.**

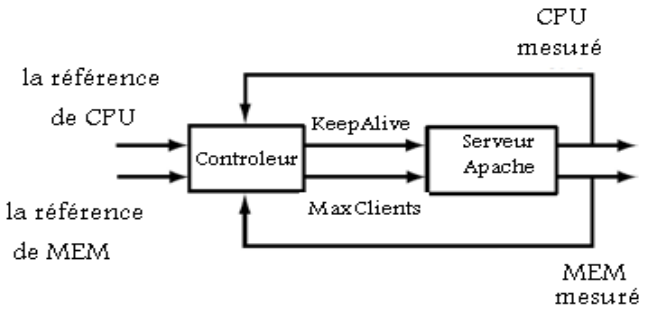


Fig. III-16 : Architecture d'un contrôle dans un serveur Apache **[7]**

3.3.4 Le contrôle dans les systèmes DiffServ

Les systèmes DiffServ ne sont posés que dans les systèmes à queue**s** qui sont largement utilisés dans les applications informatiques. Ces modèles fournissent un bon contexte d'étude des problèmes de contrôle pour les serveurs. Une file d'attente élémentaire met les requêtes des clients survenues dans une file d'attente ou buffer où elles sont sélectionnées. L'optimisation de la qualité de service dans les serveurs Web à système DiffServ, conduit à améliorer le temps de réponse des services privilégiés, ce qui est un élément essentiel pour mesurer la performance.

En général, le contrôle par le feedback du système à queue définit le temps de réponse comme le paramètre de référence, et pareillement lui-même est considéré comme le paramètre de sortie mesuré en l'absence de transducteur (organe chargé d'estimer la valeur du paramètre de sortie à la lumière d'autres paramètres, dans le cas où ce paramètre est inaccessible) comme montré dans la Figure III-17.(a). Une autre alternative, où le transducteur estime le temps de réponse en connaissant le nombre des requêtes en attente comme dans la Figure III-17.(b) **[7]**.

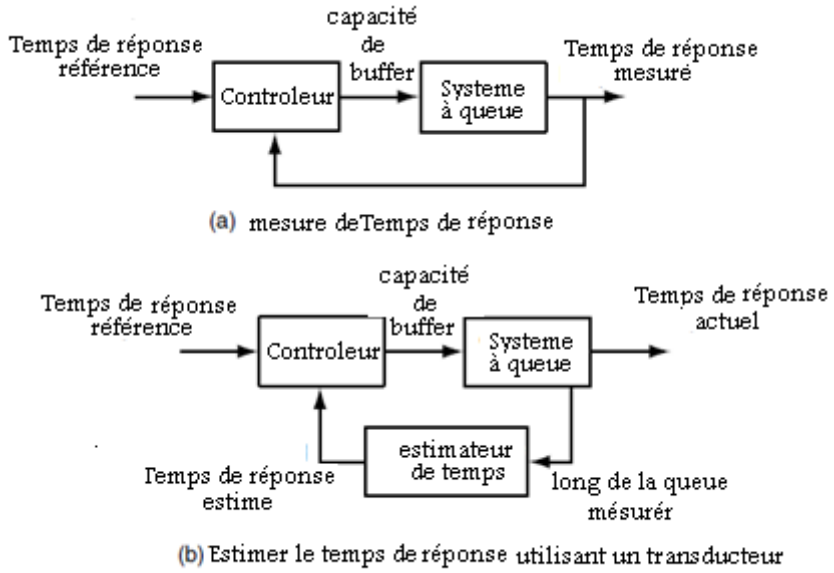


Fig. III-17 : Système de contrôle pour les systèmes à queue [7]

Les taches de contrôle pour l'optimisation du QoS dans les systèmes DiffServ s'effectuent de deux manières, soit on réagit sur le nombre de clients Maxclients, appellée contrôle d'admission, ou bien sur le choix des stratégies d'allocation du serveur aux différentes classes du service selon les formalités de performance de ces classes.

3.3.4.1 Le contrôle d'admission

Étant donné qu'en pratique les serveurs n'ont pas de files d'attente à espace illimité, un compromis est souvent inévitable pour éviter le débordement des files d'attente du serveur à l'extrême condition de surcharge.

Au but d'aboutir à des temps meilleurs de connexions pour certaines classes de services, on doit définir certains nivaux de priorité entre les requêtes en délimitant le nombre maximum de requêtes survenues par type. Chen et Mohapatra [**1**] ont proposé un modèle d'un serveur

Internet et ont tenté d'analyser la faisabilité du système DiffServ. L'objectif du modèle proposé est d'étudier les questions liées aux qualités des services à la lumière de la différenciation des services comme l'indique la Figure III-18 :

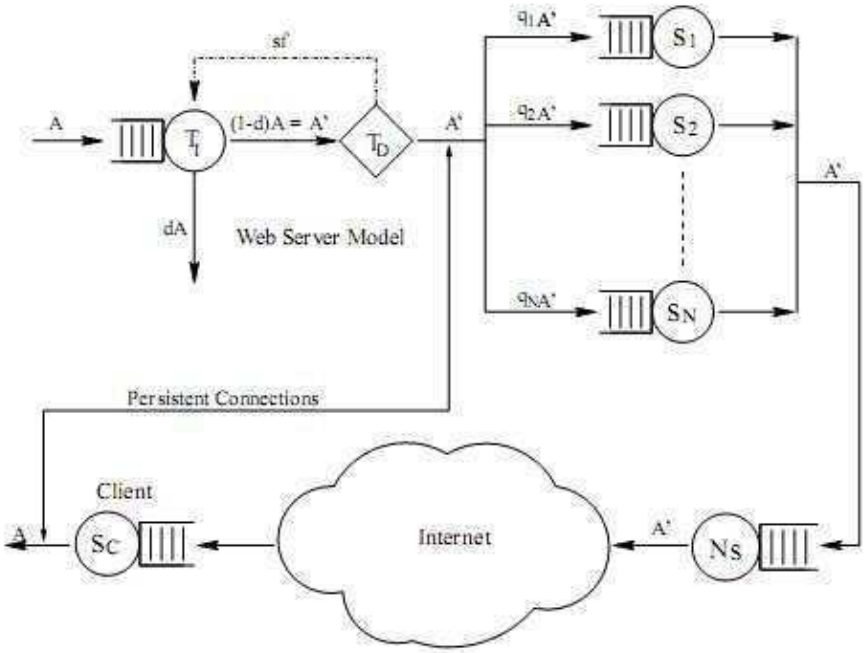


Fig.* III-18*: Un modèle DiffServ d'un serveur Web [1]

Ce modèle se compose de quatre composantes logiques principales :

- ✓ L'initiateur des taches TI : La mission de l'initiateur TI est d'ouvrir les connexions du système. Les demandes sont en attente de l'acceptation du TI à la base des paramètres du système. dA représente la probabilité des demandes rejetées si la file d'attente est pleine.

- ✓ Le distributeur des taches TD : où le répartiteur assigne à chaque demande acceptée des tâches avec un certain niveau de priorité. Le contrôle Feedback Sf du distributeur fournit à l'initiateur TI une base de paramètres afin de développer un système de contrôle pour éviter la surcharge de la file d'attente. Par exemple, dans un système à deux classes de services, il peut définir deux seuils, l'un pour les requêtes prioritaires et l'autre pour les autres. Les valeurs des seuils peuvent être ajustées dynamiquement selon l'environnement et les variétés des requêtes arrivées.

- ✓ Les Serveurs Si : Chaque tache de la classe i doit être traitée par le processus Si en fonction de leurs priorités. Ce groupe de serveur Si est un concept abstrait, dans le sens qu'il peut être un sous processus ou un thread dans un serveur multithread, un processeur d'un serveur multiprocesseur, ou un hôte dans un cluster de serveurs.

- ✓ Le canal NS : le canal de communication où les réponses sont envoyées aux clients. La capacité du canal de communication NS est déterminée par la bande passante du point d'accès du réseau des serveurs.

Le système de contrôle de rétroaction dans ce modèle intervient dans le distributeur de taches afin de limiter le nombre des requêtes dans la file d'attente selon sa priorité. Un autre moyen plus souple de contrôle de flux est de limiter le nombre des processus générés au profit des requêtes survenues au serveur.

3.3.4.2 Le contrôle selon les formalités de performance des services

Dans la différenciation des services DiffServ suivant les formalités de performance des services, les contrôleurs déterminent la politique d'allocation du serveur adopté en tenant compte des priorités des services, afin de garantir des QoSs à des délais de connexion préférentiels.

Les accords sur les niveaux des services SLA (Service Level Agreements) sont des contrats entre un fournisseur et ses clients. Ces accords constituent un ou plusieurs objectifs de niveau de service (SLO) (Service Level Objectives). Un exemple de SLO est:" le temps de connexion d'un client en Or ne devrait pas être supérieur à 2 secondes ".

Il y a trois parties pour un SLO:

- la métrique (par exemple, temps de connexion),
- la borne (par exemple, 2 sec),
- un opérateur de relation (par exemple : moins à).

Les fournisseurs de services veulent disposer de ressources suffisantes pour répondre à leurs SLO. Mais ils ne veulent pas avoir plus de ressources que nécessaires car ce fait impose des coûts inutiles. En conséquence, la réalisation d'un SLO devient souvent un problème de

régulation. La métrique de SLO est la sortie mesurée, et la borne de SLO est l'entrée de référence.

Le choix du contrôle dépend généralement de l'objectif de l'application. En effet, le système même peut disposer de plusieurs contrôleurs avec différents SLOs.

Dans les systèmes DiffServ, deux stratégies sont principalement le focus des études de contrôle. Le DiffServ de contrôle absolu, dans lequel chaque classe de requêtes a un seuil limité qui ne doit nullement être franchi. L'autre est le DiffServ du contrôle relatif, dans lequel les classes ayant les plus hautes priorités de QoS seront mieux (au moins pas pire) servies que les classes possédant des priorités moindres **[17]**.

3.3.4.2.1 Le contrôle absolu de performance

Soit une performance souhaitée W_i attribuée à chaque classe de service i (ici la performance peut être le délai de connexion, qui est le cas dans l'étude du QoS des serveurs Web), et la performance mesurée de la classe de services est C_i. Si le serveur n'est pas en état de surcharge, on exige que toutes les classes reçoivent des délais satisfaisants, donc à tout moment l'expression $C_i < W_i$ doit être correcte. Malheureusement ce n'est pas le cas dans les conditions contraires, le modèle DiffServ du contrôle absolu viole les SLOs des classes moins prioritaires au profit des services Internet plus prioritaires.

3.3.4.2.2 Le contrôle relatif de performance

Le modèle DiffServ de contrôle relatif attribue proportionnellement les performances aux différentes catégories des demandes. Cette politique de différenciation spécifie que les performances mesurées de différentes classes devraient être égales aux performances désirées.

Supposons que la performance mesurée de chaque classe de service i soit Ci. La manière d'attribution est basée sur la définition des poids θi pour chaque classe de service i $(0 \leq i \leq n)$, la classe ayant le plus grand poids c'est la plus prioritaire.

Dans une formulation théorique de contrôle, on définit la performance relative résultant par mesure de la classe i par rapport aux autres classes par:

$$R_i = \frac{C_i}{C_1 + C_2 + C_3 \ldots .. C_N} \quad \ldots\ldots\ldots \textbf{\textit{E(III.17)}}$$

D'autre part, la performance relative désirée $R_{i\ desired}$ et qui représente le point de référence pour la même classe *i* est:

$$R_{i_{desired}} = \frac{\theta_i}{\theta_1 + \theta_2 + \theta_3 \ldots ..\theta_N} \quad \ldots\ldots\ldots \textbf{\textit{E(III.18)}}$$

Nous pouvons établir le contrôle des performances relatives en comparant les deux formules **[6]**. Par conséquent, la valeur $(R_{i\ desired} - Ri)$ désigne l'erreur e_i de performance relative dans la classe *i.* Notons que l'erreur de performance relative globale du système est toujours égale à zéro.

$$\begin{aligned} &\sum_{1 \le i \le n} e_i = \sum_{1 \le i \le n} (R_{i_{desired}} - R_i) \\ &= \frac{\sum_{1 \le i \le n} \theta_i}{\theta_1 + \theta_2 + \theta_3 \ldots ..\theta_N} - \frac{\sum_{1 \le i \le n} C_i}{C_1 + C_2 + C_3 \ldots C_N} \\ &= 1 - 1 = 0 \end{aligned}$$

Donc, afin de fournir une rétroaction de performance relative à chaque classe, le contrôle s'appuie sur les sorties des différentes classes. Ce fait produit une relation non linéaire entre les classes **[6]**.

Dans le but d'éviter le non linéarité de système, une autre manière plus commode du contrôle de performance relatif, est d'achever l'égalité du rapport entre toutes deux classes adjacentes

$$\frac{W_i}{W_j} = \frac{C_i}{C_j} \quad \ldots\ldots\ldots \textbf{\textit{E(III.19)}}$$

Où W_i, C_i expriment la performance désirée (représentée en générale par le poids) et la performance résultante mesurée de la classe *i* respectivement.

Par conséquence, On obtient *n-1* boucles de rétroaction de contrôle. L'erreur e_i de performance relative dans la classe *i* est **:**

$$\mathrm{e_i} = \frac{W_i}{W_j} - \frac{C_i}{C_j} \quad \ldots\ldots\ldots \textbf{\textit{E(III.20)}}$$

Notons que la performance d'une classe et son délai de connexion T_i sont inversement proportionnelles. Pour une classe prioritaire qui possède une performance de 2 et une autre classe qui possède une performance de 1, il est nécessaire que le délai de connexions de la classe prioritaire soit la moitié de celui de l'autre classe quelque soient les circonstances. Donc on peut écrire l'égalité de contrôle de délais de connexion entre deux classes :

$$\frac{T_i}{T_j} = \frac{\theta_j}{\theta_i} \quad \ldots\ldots\ldots \textbf{\textit{E(III.21)}}$$

Abdelzaher et all **[32]** décrivent un modèle de système DiffServ suivant le contrôle des formalités de performances. Le serveur Web reçoit une demande de connexion du client suivant le protocole TCP par un paquet (SYN) via la file d'attente du serveur bien connue par le port TCP. Le serveur traitera la demande et génère la réponse, qui est par la suite envoyée au client.

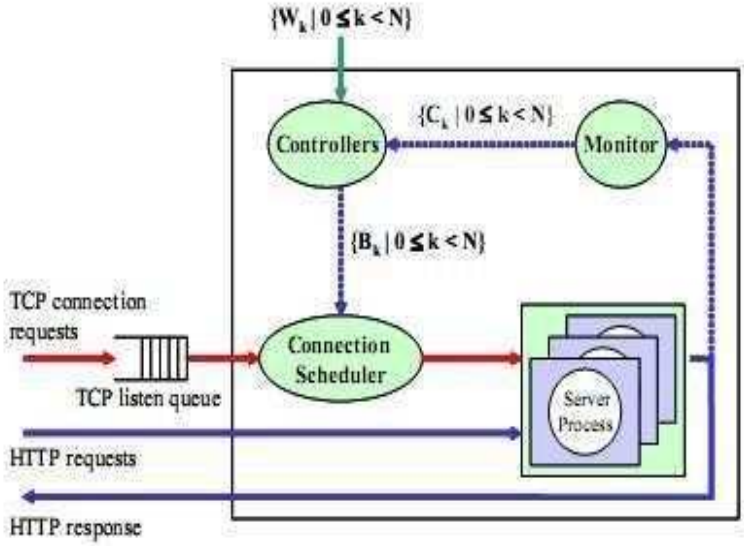


Fig. III-19: Modèle de contrôle feedback des Stratégies d'allocations [32]

Le modèle est décrit dans La Figure III-19. Ce modèle se compose de quatre composantes logiques:

L'Allocateur de connexions : le tour de l'allocateur de connexions est de classifier les requêtes HTTP survenues via la connexion TCP aux différentes classes selon la stratégie adoptée, et alloue les processus du serveur aux classes des services suivant la configuration parvenant au contrôleur.

Les processus du serveur : les processus du serveur lisent les descripteurs de la connexion et génèrent les réponses HTTP propices. Une fois qu'un processus de serveur ferme la connexion, il notifie l'allocateur qu'il est disponible pour de nouvelles connexions.

Le moniteur : à chaque échantillon de temps *m*, le moniteur relève la moyenne des délais de connexion $C_k(m)$ de chaque classe k $(0<k<n)$.

Le contrôleur : à chaque échantillon de temps *m*, le contrôleur compare l'égard des moyennes des délais de connexions calculées C_k pour les classes k (0<k<n) avec les délais désirés $W_k(0<k<n)$. L'égard utilisé par le contrôleur adapte une nouvelle configuration (appelée budget) des paramètres B_k pour chaque classe *k.* Les paramètres de budget B_k peuvent être des processus ou des unités de temps attribués à chaque classe pour le prochain échantillon de temps *m*. Ce changement de budget agit sur le système en vue d'avoir une meilleure QoS.

En revanche, dans le modèle de DiffServ de contrôle de performance relative, il est possible d'achever le contrôle à l'égard de DiffServ de contrôle de performance absolue, car dans les conditions de surcharge du système, il est impossible de satisfaire les délais de toutes les classes. Et par conséquent les classes à faible priorité peuvent subir de violation par les classes de haute priorité **[32]**.

Le système DiffServ avec le contrôle rétroaction respectant la relativité des temps de connexions désirés apporte une bonne QoS pour les flux prioritaires. Mais il se peut que les temps de connexions s'allongent à des délais inadmissibles surtout dans les cas de surcharges du flux.

D'autres recherches, semblables à l'étude de Xiabo Zhou et al **[17]**, apportent des solutions à la DiffServ relative, en constituant un module adaptatif de contrôles pour rattraper les fractions des ressources dispersées, figurées en threads inactifs associés à une classe de services. Ces fractions des ressources doivent être relouées à une autre classe pour couvrir son besoin.

Chenyang Lu et all **[11]** proposent un système hybride, qui n'est qu'une extension de système DiffServ proportionnel. Ce système applique le contrôle DiffServ relatif. Cependant, dans l'état de surcharges, le système se change au régime DiffServ absolu pour remédier à l'incommodité de contrôle DiffServ relatif.

3.4 Conclusion

Dans ce chapitre nous avons présenté tout d'abord les modèles de systèmes de files d'attente classiques, pour lesquels nous présentons des évaluations analytiques concernant les délais de connexions et la charge du système.

D'autre part, on a vu comment exploiter l'avantage de la technique de la régulation automatique, pour l'amélioration de la qualité de service des serveurs Web.

Le contrôle dans les modèles DiffServ est établi par deux façons. La première se fait à l'aide de contrôle d'admission, cette technique classique est employée pour empêcher les serveurs de s'écrouler quand la charge devient trop importante **[33]**. La deuxième, c'est l'allocation du serveur aux différentes classes du service selon les formalités des performances de ces classes.

Dans cette dernière stratégie, on a exposé deux modèles, le modèle de contrôle des délais relatifs et le modèle de contrôle des délais absolus.

On peut constater que le modèle de contrôle des délais relatifs est meilleur que le modèle de contrôle des délais absolus, puisque ce dernier viole les SLOs des classes moins prioritaires, Cependant le contrôle des délais absolus est plus apprécié, notamment pour les applications en continu et à contraintes de temps comme l'audio / vidéo [**17**].

CHAPITRE IV
ETUDE COMPARATIVE ET PROPOSITION DE SYSTEME

4.1 Introduction

Les systèmes DiffServ introduisent la notion de priorité, qui permet de différencier les services entre les différents flux en fonction de leur importance. En effet ces systèmes garantissent une QoS plus appréciable pour les flux les plus importants.

Comme vu au chapitre précédent, les différents types d'allocation des serveurs Web, sont le type de file d'attente classique ou le type de file avec rétroaction. Le régime d'allocation que nous jugeons préférable pour améliorer la QoS, est exposé ci-après.

L'amélioration du QoS des serveurs Web dans un système DiffServ conduit à améliorer les temps de connexion aux services offerts. Ces derniers sont liés aux taux d'arrivées des demandes, ainsi qu'aux taux de services attribués pour chaque classe de services. Le délai de connexion est proportionnel au taux de services, et il est inversement proportionnel au taux d'arrivée. Donc pour améliorer les délais de connexion, soit on minimise le taux d'arrivée, soit on augmente le taux de service par allocation de plus de ressources serveur.

Le premier choix consiste à limiter le nombre de clients en concurrence sur le serveur, en utilisant des files d'attente à tailles limitées pour rejeter les demandes dans le cas des surcharges. Il est clair que ce choix a des conséquences directes sur sa performance, la disponibilité de son service et sa qualité de service (QoS) **[2]**. Cette solution peut dégrader la QoS de serveur Web, à cause de la possibilité d'inhiber inutilement des demandes en cas de surcharge d'une seule file d'attente, malgré que l'ensemble du système soit dans une situation considérée comme de détente.

En conséquence, c'est le deuxième choix qui peut accroitre la QoS, bien sûr en offrant assez de ressources à nos besoins aux flux prioritaires, et ce malgré son effet gênant, qui se fait au détriment des flux moins prioritaires.

4.2 La QoS et le serveur Web GPS

Pour améliorer la QoS d'un serveur Web, on doit réduire les délais de connexion, particulièrement les flux prioritaires, à travers l'allocation de plus de quantité de services pour ceux-ci.

Le système à files d'attente prioritaires attribut des niveaux de priorité. Cette attribution est statique. En effet, les flux de priorités plus basses peuvent subir des délais assez importants en attendant la fin de service des flux prioritaires, et qui peut engendre le problème de la famine. Ce phénomène conduit au rejet de paquets des flux moins prioritaires quand les files d'attente correspondantes deviennent saturées ou le temps TimeOut prédéfini de serveur Web est atteint. Donc la priorité stricte ne permet pas de fournir une qualité de service flexible aux besoins des applications moins prioritaires.

Pour remédier à ces problèmes, des algorithmes de partage de ressources ont été proposés et ont suscité beaucoup d'intérêt dans les dernières années visant à gérer plus efficacement les ressources entre les différents flux. L'objectif de ces algorithmes consiste à assurer un accès aux ressources d'une manière équitable et empêcher qu'un flux puisse consommer plus de ressources que le taux qui lui est alloué.

En effet les systèmes à files d'attentes pondérée GPS semblent plus commodes puisque elles sont équitablement distribuées. La Figure IV-1 ci-dessous montre le fonctionnement du serveur Web GPS suivant les poids attribués aux classes de flux de requêtes :

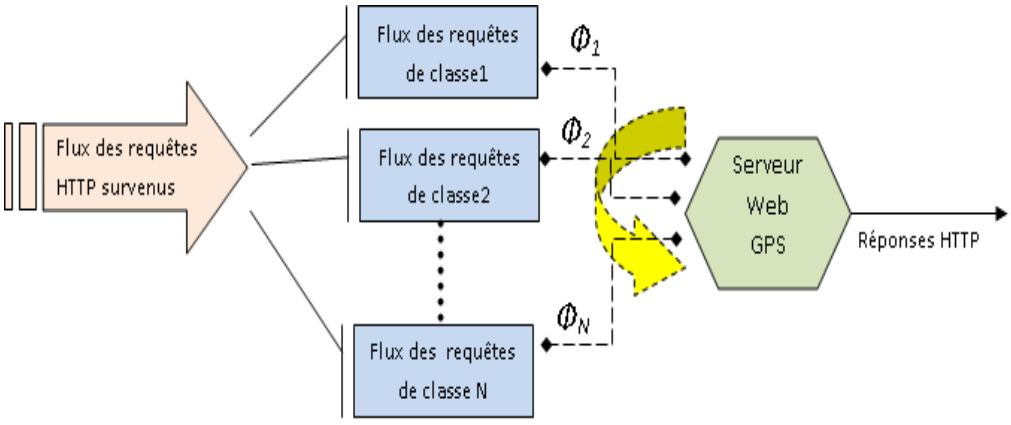


Fig. IV-1 : Processus d'allocation suivant la pondération

Afin d'améliorer le temps de connexion d'un flux dans le système GPS, on doit augmenter son taux de service à travers une amplification du coefficient de pondération attribué à ce flux, ce facteur notifié par le taux de service (représenté en poids sur la totalité du service du serveur Web) indiquera par la suite le paramètre du QoS.

Les coefficients de pondération statiques ou prédéfinis associés aux files d'attentes GPS déterminent le taux de service alloué à chaque flux. En contrepartie les taux d'arrivées des demandes survenues ne sont pas proportionnels à ces poids. Subséquemment ceci conduit sans doute à une déficience de la QoS surtout pour les files prioritaires surchargées.

En effet, on fait appel à d'autres types de DiffServ où les coefficients de pondération varient selon l'environnement du serveur Web, et qui sont basés sur la rétroaction pour achever les valeurs des consignes désirées, et par la suite, on détermine les taux de services convenables.

Dans les deux paragraphes ci-après, nous déterminons les taux de service associés aux classes pour deux approches, l'approche des délais relatifs et une approche proposée c'est le WFQ , alors que le paragraphe suivant, montre une comparaison entre ceux-ci et le Best effort.

4.3- Analyse du système DiffServ de délais relatifs

Le système DiffServ des délais relatifs n'est qu'une sorte de système GPS qui cherche à donner une meilleure QoS pour les flux prioritaires.

Comme on l'a décrit dans le chapitre précédent, le système DiffServ des délais relatifs respecte la relativité des temps de connexion désirés, par exemple pour que le temps de connexion de classe 1 soit la moitié de classe 2 on a : $2T_1=T_2$

Plus formellement, les temps de connexion des classes sont calculés suivant leurs poids de QoS, la classe ayant le plus grand poids est la plus prioritaire. D'après l'équation E(III.21), la proportionnalité des délais de connexion s'écrit:

$$\frac{T_i}{T_{i+1}} = \frac{\theta_{i+1}}{\theta_i} \quad \text{.....} \textbf{\textit{E(IV.1)}}$$

Où Ti et θ_i représentent le temps de connexion et le facteur de QoS de la classe de service i respectivement.

D'autre part, chaque classe dans le système DiffServ est perçue comme isolée avec un système FIFO M/M/1. Le temps moyen de séjour dans le système M/M/1: $T = \frac{1}{\mu - \lambda}$.

D'après E(IV.1):

$$\frac{\mu_{i+1} - \lambda_{i+1}}{\mu_i - \lambda_i} = \frac{\theta_{i+1}}{\theta_i}$$

$$\mu_i - \lambda_i = \frac{\theta_i}{\theta_{i+1}}(\mu_{i+1} - \lambda_{i+1}) \quad \text{.....} \textbf{\textit{E(IV.2)}}$$

Où μ_i et λ_i représentent le taux de service et le taux d'arrivée des demandes de la classe *i* respectivement. Posons la différence $\Delta_i = \mu_i - \lambda_i$, d'après E(IV.2):

$$\left\{ \begin{array}{l} \Delta_1 = \frac{\theta_1}{\theta_2}\Delta_2 \\ : \\ : \\ \Delta_i = \frac{\theta_i}{\theta_{i+1}}\Delta_{i+1} \\ : \\ \Delta_{n-1} = \frac{\theta_{n-1}}{\theta_n}\Delta_n \\ \Delta_n = \Delta_n \end{array} \right\} \Rightarrow \quad \Delta_i = \frac{\theta_i}{\theta_n}\Delta_n \quad \text{.....} \textbf{\textit{E(IV.3)}}$$

D'après E(IV.3):

$$\sum_{i=1}^{n} \Delta_i = \sum_{i=1}^{n} (\mu_i - \lambda_i) = \sum_{i=1}^{n} \frac{\theta_i}{\theta_n}\Delta_n$$

$$\sum_{i=1}^{n} \mu_i - \sum_{i=1}^{n} \lambda_i = \sum_{i=1}^{n} \frac{\theta_i}{\theta_n}\Delta_n$$

$$\mu - \lambda = \sum_{i=1}^{n} \frac{\theta_i}{\theta_n}\Delta_n$$

Donc : $$\Delta_n = \frac{\theta_n}{\sum_{i=1}^{n} \theta_i}(\mu - \lambda) \quad \text{.....} \textbf{\textit{E(IV.4)}}$$

D'après E(IV.3), et E(IV.4) : $$\Delta i = \frac{\theta_i}{\theta_n} \times \frac{\theta_n}{\sum_{i=1}^{n} \theta_i}(\mu - \lambda) = \frac{\theta_i}{\sum_{i=1}^{n} \theta_i}(\mu - \lambda)$$

D'où: $$\mu_i = \lambda_i + \frac{\theta_i}{\sum_{i=1}^{n} \theta_i}(\mu - \lambda) \quad \ldots\ldots \ldots\ldots\ldots\ldots \textbf{\textit{E(IV.5)}}$$

Ainsi le temps moyen de séjour d'une classe de service dans le système M/M/1:

$$T_i = \frac{1}{\mu_i - \lambda_i} = \frac{1}{\lambda_i + \frac{\theta_i}{\sum_{i=1}^{n} \theta_i}(\mu - \lambda) - \lambda_i}$$

D'où : $$T_i = \frac{\sum_{i=1}^{n} \theta_i}{\theta_i (\mu - \lambda)} \quad \ldots\ldots\ldots\ldots\ldots\ldots \textbf{\textit{E(IV.6)}}$$

On peut conclure que dans le système DiffServ des délais relatifs, le temps de connexion d'un flux, est lié à son poids, au taux global des requêtes survenues au système et il est indépendant à son propre taux d'arrivée.

La Figure IV-2 présente les graphes du temps de connexion de 3 classes dans un système DiffServ des délais relatifs, dont chaque classe de service ayant une *double* performance de la classe suivante, nous remarquons que les temps obtenus respectent cette relativité.

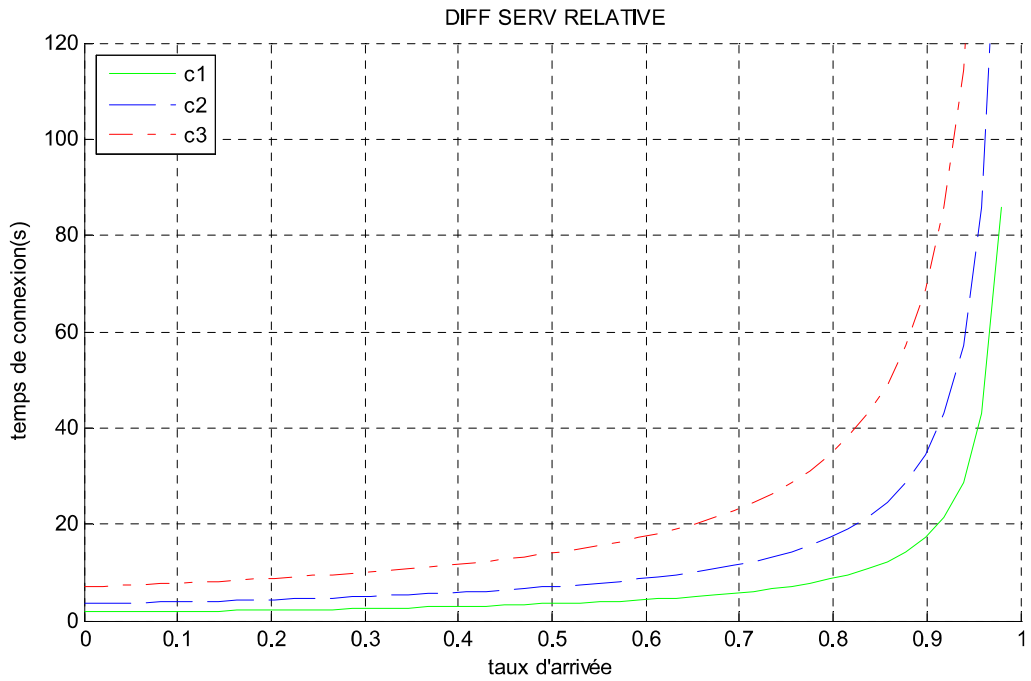


Fig. IV-2: graphes du temps de connexion dans un système DiffServ relative

En revanche, dans le système DiffServ des délais relatifs, on recense les problèmes suivants :

1. si l'un des flux cesse de procréer des requêtes à cause de défaillance du réseau ou d'indisponibilité de service, cela entraîne une incommodité de la proportionnalité des classes.
2. il se peut que les temps de connexions puissent s'allonger à des délais inadmissibles surtout dans les cas de surcharges du flux.
3. la proportionnalité peut assigner plus de ressources éventuellement aux services prioritaires dont on a besoin.

Certaines solutions sont proposées pour ces inconvénients:

- Pour le premier, on peut travailler qu'avec les classes actives et en n'affectant plus de ressources aux autres.
- Pour le deuxième, Chenyang Lu et all **[11]** établissent le contrôle des délais absolus.
- Enfin, L'étude de ***Xiabo Zhou*** et all [*17*], propose une solution au dernier problème du DiffServ des délais relatifs, en constituant un module adaptatif de contrôles pour rattraper les fractions des ressources dispersées.

De notre part, on a préféré de trouver un autre modèle où il n y 'a pas de perte de ressources, c'est le modèle WFQ.

4.4 Analyse du système DiffServ WFQ

Un autre modèle du système à files d'attentes est le WFQ qui ne se penche plus exclusivement sur les coefficients de pondération comme le GPS, mais qui varie selon les conditions du système. Ce genre de système exposé dans le chapitre 3 soutient les flux qui subissent un taux d'arrivée élevé, et amoindrit la mise en considération des poids primitifs désignés. Donc les poids doivent être recalculés pour établir le processus d'allocation selon le fameux système WFQ.

Puisque le contrôle WFQ garantit les taux du service aux différents flux proportionnellement à leurs taux d'arrivée et modérément à l'égard des poids φi des flux i, $i=1..n$, ces poids ne fournissent pas le taux véritablement exploité par le serveur Web. Donc il est nécessaire de

calculer de nouveau, le poids $\varphi'i$ qui sera attribué au flux *i* pour représenter le taux de service r_i réellement exploité.

Revenons au système WFQ vu dans le chapitre 3, la charge moyenne des flux dans une file WFQ est influencée par les poids φ_i attribué aux flux *i*, et par les flux présents dans les files WFQ, la charge moyenne de flux *i* est donnée par la formule mathématique E(III.19):

$$Q_i \approx \varphi_i . \frac{\rho_i}{1-\rho_i} + (1-\varphi_i)\left(\frac{\rho}{1-\rho} - \frac{\sum_{j=1..N, i\neq i} \rho_j}{1-\sum_{j=1..N, i\neq i} \rho_j}\right)$$

Où : $\frac{\rho_i}{1-\rho_i}$: est le taux d'occupation probable par le trafic venant des files *i.*

$\frac{\rho}{1-\rho}$: est le taux d'occupation global par le trafic venant des files WFQ

$\frac{\sum_{j=1..N, i\neq i} \rho_j}{1-\sum_{j=1..N, i\neq i} \rho_j}$: est le taux d'occupation par le trafic venant des files WFQ autres que *i.*

Ainsi, le système peut être perçu comme des flux isolés, chacun est similaire à une file M/M/1. Dans ce cas là, la charge moyenne de la file pour chaque flux *i* est donnée par E(III.6):

$$Q = \frac{\rho'^2}{1-\rho'}',$$

où ρ'_i est l'intensité de trafic du flux *i.*

Pour trouver la valeur de ρ'_i, on doit résoudre l'équation de second degré:

$$\rho_i'^2 + Q_i \rho_i' - Q_i = 0$$

On obtient alors: $\rho_i' = \frac{-Q_i + \sqrt{Q_i^2 + 4Q_i}}{2}$ ……***E(IV.7)***

En outre on peut exprimer le taux du service réel r_i attribué par le serveur au flux *i,* en utilisant l'équation E(III.4): $r_i = \frac{\lambda_i}{\rho_i'}$

Donc, on peut attribuer un poids φ'_i pour écrire le taux du service réel r_i en fonction du taux global de service μ du serveur Web: $r_i = \frac{\varphi'_i}{\sum_j \varphi'_j}.\mu$.

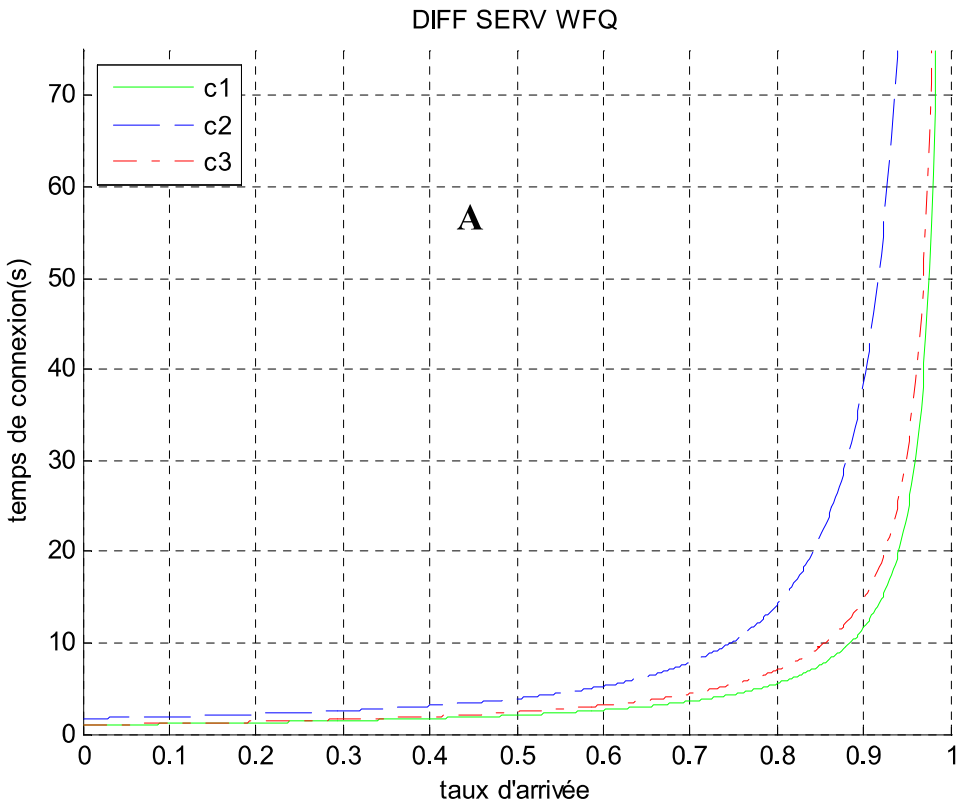
Ainsi le temps moyen de séjour d'une classe de service vaut d'après E(III.7):

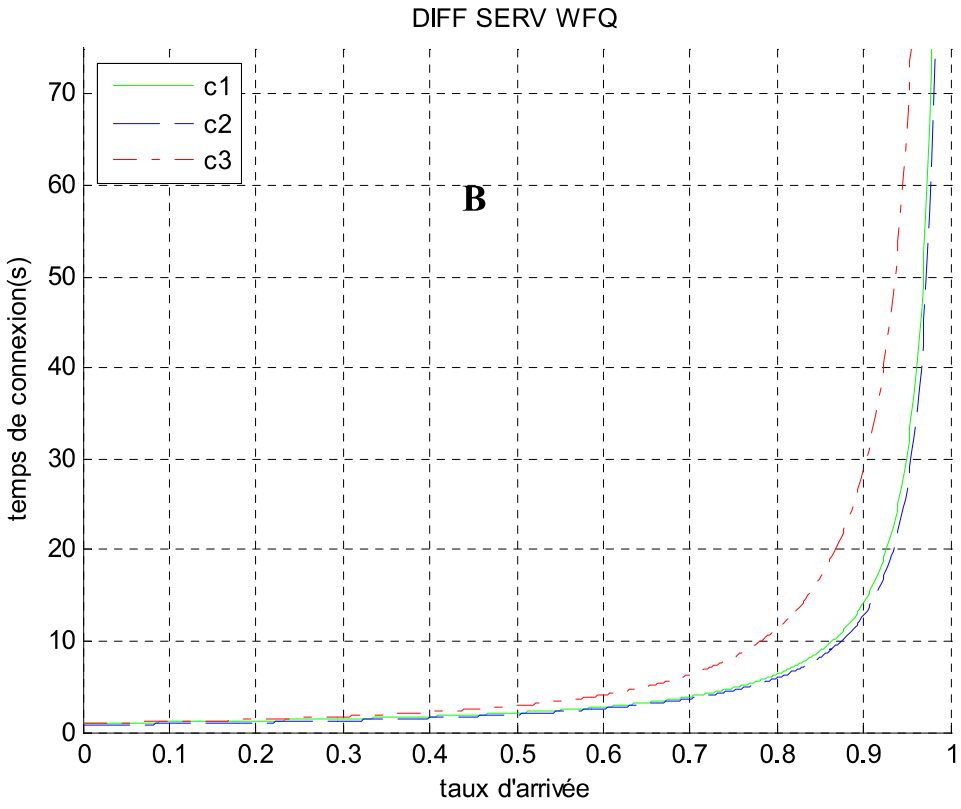
$$T_i = \frac{1}{r_i - \lambda_i} \quad \text{.....} \textbf{\textit{E(IV.8)}}$$

En revanche que l'approche WFQ est plutôt convenable si les classes prioritaires subissent des taux d'arrivées élevés par apport aux autres (voir la Figure IV-3). Les graphes dans la Figure IV-3 présentent les temps de connexions de trois investigations A, B et C de trois classes dans un système DiffServ WFQ dont chaque classe de service possède un poids et un taux d'arrivée qui est fonction du taux d'arrivée global (pour la représentation graphique). Par exemple l'intensité de trafic de la classe 2 dans la figure C est égale au poids 0.35 × taux global du trafic des demandes survenues au serveur Web. Le Tableau IV-1 ci-dessous montre les valeurs utilisées dans les trois investigations.

Investigation	**Classe**	**Poids**	**Poids du taux d'arrivée d'une classe en fonction du taux d'arrivée global**
A	1	0.50	0.40
	2	0.30	0.10
	3	0.20	0.50
B	1	0.50	0.30
	2	0.30	0.50
	3	0.20	0.20
C	1	0.50	0.10
	2	0.30	0.45
	3	0.20	0.45

Tab. IV-1 : Paramètres des investigations WFQ


DIFF SERV WFQ
c1
c2
c3
A
temps de connexion(s)
taux d'arrivée
0
10
20
30
40
50
60
70
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


DIFF SERV WFQ
c1
c2
c3
B
temps de connexion(s)
taux d'arrivée
0
10
20
30
40
50
60
70
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

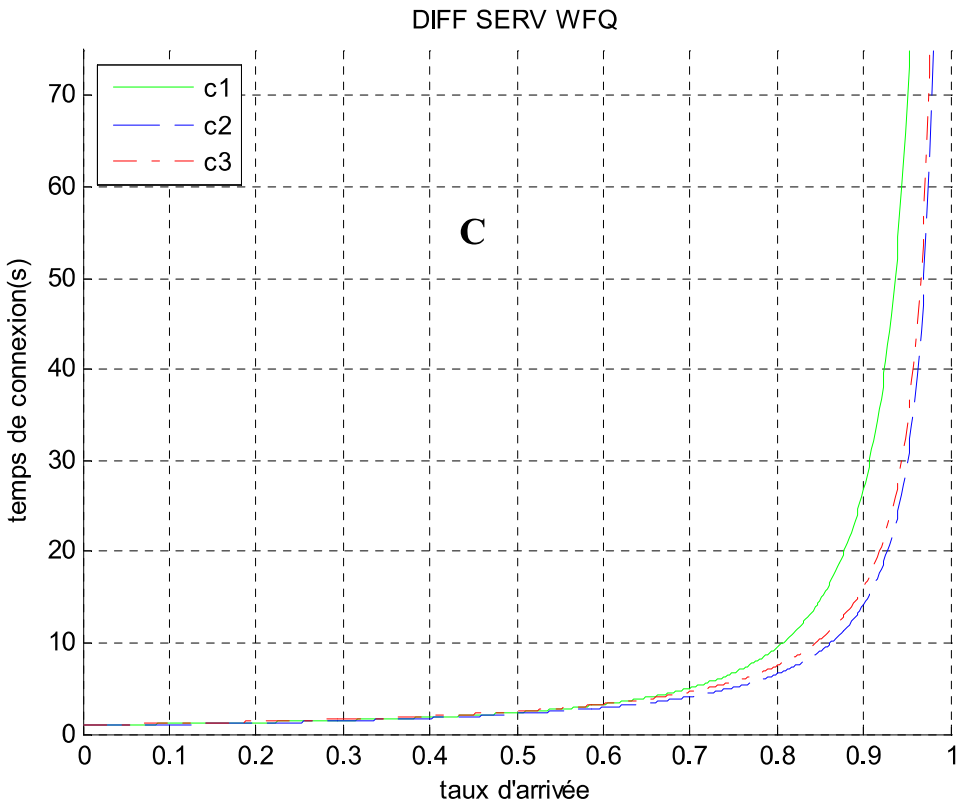


Fig. IV-3: Graphes du temps de connexion A, B et C respectivement dans un système DiffServ WFQ

4.5 Comparaison des modèles délais relatifs, WFQ & le Best effort

La Figure IV-4 présente une comparaison entre les graphes du temps de connexion des types DiffServ des délais relatifs, WFQ vis-à-vis du Best effort:

La comparaison est établie avec la performance de classe1 est égale au *double* de la performance de classe 2, et idem pour classe 2 et classe 3 dans le DiffServ des délais relatifs. Et de la même manière, les poids des classes WFQ sont établis (4/7, 2/7, 1/7). D'autre part les requêtes survenues aux différentes classes de services avec les taux (0.4, 0.3, 0.3) respectivement

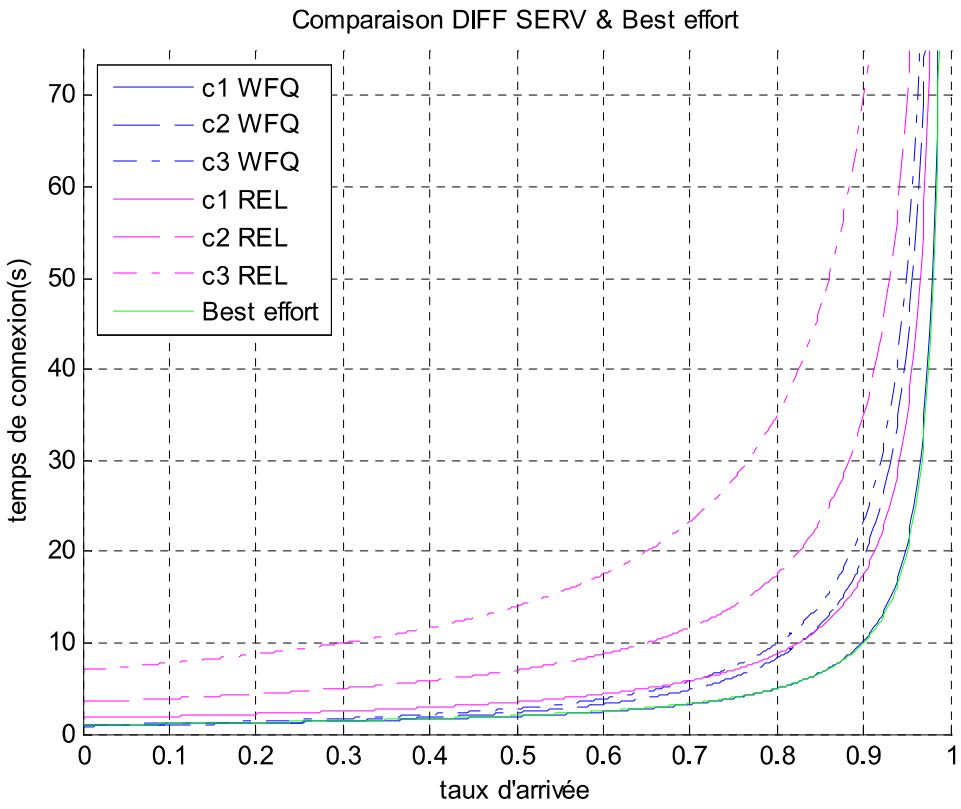


Fig. IV-4: Comparaison entre les graphes du temps de connexion DiffServ et le Best effort:

On peut voir dans la Figure IV-4, malgré que le DiffServ WFQ soit meilleur que le DiffServ des délais relatifs, mais il n'est pas aussi bon que le Best effort.

D'autre part, nous avons constaté que puisque les formules de DiffServ WFQ sont approximatives, elles peuvent utiliser plus de ressources du serveur Web qu'on en dispose (dépassement pouvant aller à 120%) comme le montre la Figure IV-5.

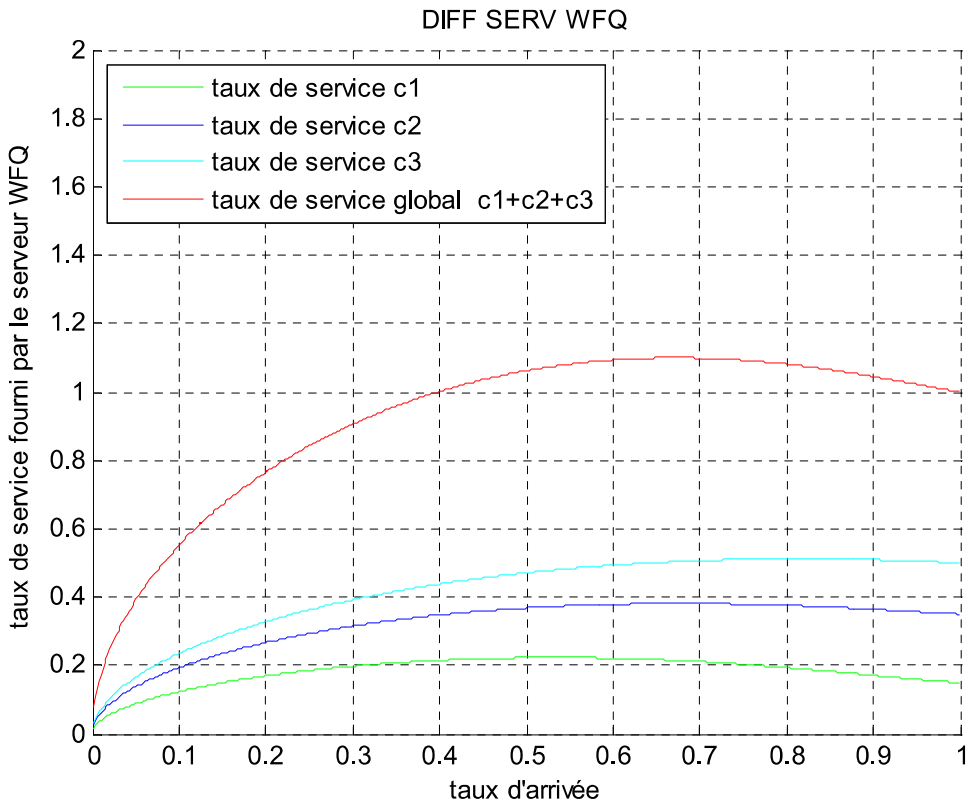


Fig. IV-5: Le dépassement des ressources disponibles dans le DiffServ WFQ

Nous constatons que le système DiffServ GPS avec ses différents types ne donne plus de QoS meilleure que le Best effort.

Ce résultat met la lumière de l'impropriété des systèmes DiffServ des délais relatifs où le DiffServ WFQ qui sont basés sur la désagrégation de taux de service (exprimé en nombre de processus ou de thread), puisqu'ils ne donnent pas de QoS meilleure que le Best Effort.

Suite à cette conclusion, on doit se pencher sur une autre politique de DiffServ qui atteint l'objectif de l'amélioration du QoS du serveur Web. Après dans la section qui suit la justification analytique, nous décrivons le système adopté.

4.5.1 Justification analytique

Pour une meilleure performance de service pour les classes prioritaires, nous devons avoir des temps de connexion plus petits que les temps obtenus dans un système Best Effort. Soit la classe 1 une classe plus prioritaire, et soient T et T_1 les temps de connexion en Best effort, et de la classe 1 en appliquant une désagrégation du taux de service. Supposons que: $T_1 < T$

Chaque classe dans un système DiffServ est perçue comme isolée avec un système M/M/1.

D'après (III.7) : le temps moyen de séjour dans le système Best Effort : $T = \frac{1}{\mu - \lambda}$

Dans un système DiffServ le trafic de la charge survenu au système est la somme des trafics de toutes les classes: $\lambda = \sum_{i=1}^{N} \lambda_i$. D'autre part le taux de service global du serveur Web est la somme des taux de services de ces classes : $\mu = \sum_{i=1}^{N} \mu_i$.

Nous rappelons que le temps moyen de séjour dans le système Best effort (FIFO) M/M/1:

$$T = \frac{1}{\mu - \lambda}$$

$$\mu - \lambda = \frac{1}{T}$$

$$\sum_{i=1}^{N} \mu_i - \sum_{i=1}^{N} \lambda_i = \frac{1}{T}$$

$$(\mu_1 - \lambda_1) + (\mu_2 - \lambda_2) + ...(\mu_N - \lambda_N) = \frac{1}{T}$$

Pour un régime stationnaire pour toutes les classes, cela signifie que toutes les demandes des classes sont servies, on doit avoir $\mu_i > \lambda_i$.

Ce qui implique que : $\mu_i - \lambda_i > 0$ est correct.

Choisissons la classe 1 puisqu'elle est la plus prioritaire, on a :

Donc: $(\mu_1 - \lambda_1) < \frac{1}{T}$ $\Longrightarrow$ $\frac{1}{T_1} < \frac{1}{T}$

$\Longrightarrow$ $T_1 > T$ Ce qui contredit la supposition.

4.6 Description du système adopté

Le but de notre approche est de tirer profit davantage de régulation afin de réaliser un module d'ajustement dynamique qui commande les taux de services des classes des services du serveur Web. En effet, cette régulation doit être relative aux besoins des classes et doit permettre de favoriser des flux, en évitant le maximum de détriments des autres qui est le cas délicat de la surcharge.

Du fait que le Best effort conduit à des temps de connexion intolérables pour les services prioritaires dans le cas de surcharge, il faut établir un autre type de DiffServ où le temps de connexion est meilleur que le Best effort dans le cas de surcharge.

Cette approche de différenciation de service notifie par le DiffServ des délais absolus, s'adapte de façon similaire au système à files prioritaires classique, tout en offrant autant plus de chances aux classes moins prioritaires que la propriété exacte.

Cette approche exige des contraintes d'objectifs aux niveaux des services appelant des SLOs qui sont prédéfinis pour chaque flux, et qui devront êtres respectés. Ainsi notre but par la suite sera d'atteindre les consignes qui ne sont là que les délais de connexion indiqués par les SLOs puisqu'elles expriment bien la QoS.

Etant donné que dans un système DiffServ les flux n'ont pas la même priorité, dans le DiffServ des délais absolus, la distinction de priorités entre les flux n'est qu'un contrat à respecter. Ce contrat garantit la QoS par allocation des quantités de service suffisantes pour obtenir un temps de connexion tolérable.

Pour chaque flux, on doit disposer d'une contrainte SLO exprimée dans une inéquation de la forme : $t_i<tcxagr_i$, ce qui signifie que le temps de connexion de flux i est moindre que le temps maximum de connexion agréé. Pour une valeur plus grande pour ce temps de connexion, la requête prend le risque d'être rejetée. Généralement ce risque vise les classes les moins prioritaires, tandis que les contraintes des autres flux sont menacées par la violation de ses contraintes.

Notre système propose d'appliquer une politique hybride de processus d'allocation du serveur Web. L'administration de politique adoptée est établie sur deux axes, et selon l'état de charge

du système, c.-à-d. le facteur d'utilisation globale du serveur $\rho = \frac{\sum \lambda_i}{\mu}$ où λ_i, μ représentent respectivement le taux d'arrivée du flux *i* et le taux de service global du serveur Web, avec $\sum_{i=1....N} \lambda_i < \mu$ pour un système à régime stationnaire.

Sur le premier axe où le facteur d'utilisation globale du serveur ρ n'est pas grand, on applique le Best effort et on mesure les délais de connexion affectés à chaque classe de requêtes. Notre objectif dans ce cas là, est le contrôle des contraintes SLOs: $t_i < tcxagr_{i.,}$ et pour que les demandes ne soient pas rejetées par le serveur Web, il faut que $tcxagr_i < timeout$.

Sur le deuxième axe où le facteur d'utilisation globale du serveur ρ est assez important, l'établissement du système précédent peut défaillir la QoS en violant les contraintes SLOs, notamment si la charge sur les flux moins prioritaires est importante. D'où le fait indispensable de remplacer le processus antérieur par un autre dont la QoS soit raffinée. De là on incite le processus d'ajustement des coefficients par le contrôle DiffServ des délais absolus de connexions.

Dans la section suivante, nous examinons le contrôle Feedforward de l'approche d'allocation DiffServ des délais absolus, basé sur les théories des files, en calculant les valeurs des taux de services assignés à chaque classe pour atteindre les temps consignes notifiés par $tcxagr_i$.

La section juste après, explique le principe de contrôle Feedback présenté par un PID qui assure la stabilité de système subi d'une variation sur les taux des requêtes survenues ou une perturbation de service causée par d'autres applications résidantes dans le serveur Web.

La dernière section est consacrée au modèle de contrôle DiffServ des délais absolus afin d'avoir un système stable et puissant, puisque il est adaptatif au changement d'objectif et aux perturbations probables sur le stade du serveur Web, et par conséquent allonge sa qualité de service.

4.6.1 La perception Feedforward du paramètre de QoS

Le contrôle Feedforward est basé sur les théories des file d'attente, et essaye de percevoir une valeur référence pour le paramètre d'entrée contrôlé afin d'achever la valeur désirée du paramètre en sortie.

Nous avons indiqué que le système peut être perçu comme des flux isolés, chacun est similaire à une file M/M/1. Dans ce cas là, la charge moyenne d'une file d'attente est:

$$Q = \frac{\rho^2}{1-\rho}$$

Où ρ est l'intensité du trafic.

Et le temps moyen d'attente dans une file d'attente est donné par la formule de Little: $W = \frac{Q}{\lambda}$

D'où : $W = \frac{\rho^2}{(1-\rho)\lambda} = \frac{\lambda^2}{(1-\frac{\lambda}{\mu})\lambda\mu^2} = \frac{\lambda}{\mu^2 - \lambda\mu}$

En partant que le temps de connexion agrée maximum *W = tcxagr*, et afin de trouver la valeur du taux de service éventuel r_i exigée par le SLO du flux *i* on doit donc, résoudre l'équation à seconde dégrée:

$$\mu_i^2 - \lambda_i . r_i \mu_i - \frac{\lambda_i}{\text{tcxagr}_i} = 0$$

Ainsi, on obtient: $\mu_i = \frac{\lambda_i + \sqrt{\lambda_i^2 + 4\frac{\lambda_i}{\text{tcxagr}_i}}}{2}$ …… …… ***E (IV.9)***

Cependant, dans le cas de la surcharge on aura forcement: $\sum_{i=1....N} \mu_i > \mu$, où μ représente le taux de service global du serveur Web .

En effet, le contrôle des délais absolus s'effectue au détriment des flux moins prioritaires et au profit des autres. Ainsi, le taux du service r_i correspondant à la consigne $tcxagr_i$ établie par le serveur pour le flux *i* est :

$$r_i = Min(\mu_i, Max(0, \mu - \sum_{j=1....i-1} \mu_j)) \text{ …… } \textbf{E (IV.10)}$$

Donc le poids en Feedforward de la classe *i,* s'écrit en fonction du taux de service global du serveur Web μ : $\theta_i = \frac{r_i}{\mu}$

Notons que nous irons substituer le temps d'attente par le temps de connexion pour nous assurer que les demandes ne soient pas rejetées par le serveur Web en tenant en compte que: $tcxagr_i < timeout$

4.6.2 Principe de contrôle Feedback des délais absolus de connexions

Afin d'obtenir un système puissant et stable, il est nécessaire d'intégrer un régulateur sur le modèle DiffServ, c.à.d. employer des régulateurs comme P, PI, PID prenant en charge l'ajustement dynamique des poids des files d'attente pour aboutir un niveau adéquat de qualité de service.

La valeur consigne désigne généralement un niveau de paramètre de QoS qu'on doit asservir comme la charge dans la file d'attente ou le délai de connexion. L'écart entre la valeur consigne Y_c d'un tel paramètre à réguler et sa valeur mesurée Y_m sera alors transmise vers le régulateur. Ainsi, une commande *u* relative à chaque erreur recensée d'une classe, est générée par ce dernier.

La Figure IV-6 ci-dessous explique le déroulement de l'approche feedback adoptée pour l'ajustement dynamique:

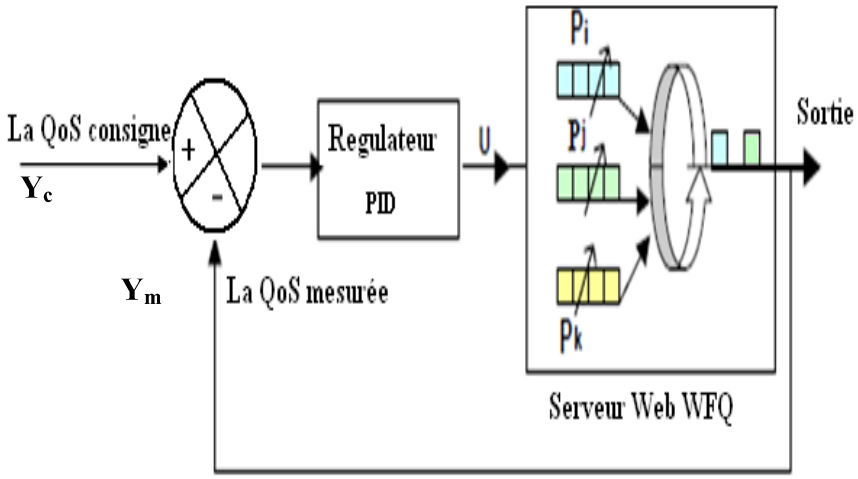


Fig. IV-6: Principe d'ajustement dynamique

Dans notre système, Les classes seront servies suivant leurs poids affectés θ_i. Constamment à chaque fraction de temps, et pour chaque classe, on mesurera les délais de connexion. Le correcteur génère une commande u relative à chaque erreur calculée (entre le délai de connexion mesuré du flux i et $tcxagr_i$). Cette commande ajustera le coefficient de pondération θ_i de chaque file et déterminera le taux de service éventuellement assigné par le serveur Web.

Le contrôle des délais absolus appose un ajustement dynamique aux taux de services pour atteindre les valeurs consignes des délais de connexions $tcxagr_i$ désirés. Ces taux de service sont ajustés suivant les exigences des SLOs qui représentent les taux de services éventuels. La totalité de ces taux de service éventuels doit dépasser le taux de service global puisque on est dans la surcharge. Bien que dans un contrôle des délais absolus, les flux moins prioritaires subissent un préjudice sur ces SLOs, cela peut s'allonger jusqu'à la famine stricte, et ceci au profit des flux plus prioritaires dont ces SLOs ne doivent jamais violer.

Établissons que la QoS consigne du contrôle rétroaction est désignée par: $Y_c = \{tcxagr_i, 1 <= i < N\}$, où $tcxagr_i$ représente le délai de connexion consigne du flux i. En contrepartie la QoS mesurée est révélée par: $Y_m = \{t_i, 1 <= i < N\}$, où t_i représente le délai de connexion réel écoulé pour que la requête du flux i soit servie par le serveur Web. Ainsi l'erreur de la chaîne rétroaction est la dissimilitude entre ces deux valeurs $\varepsilon = \{ei = tcxagr - t_i, 1 <= i < N\}$

Le principe de la régulation se résume, ainsi, par les étapes suivantes :

1. Spécification des paramètres prédéfinis : les délais de connexion consignes sont ceux définis par les SLOs associés à chaque classe: $Y_c = \{tcxagr_i, 1 <= i < N\}$, et qu'on ne doit pas franchir.
2. Mesure de la métrique : relevé les délais de connexions moyennes réelles établies par le serveur $Y_m = \{t_i, 1 <= i < N\}$
3. Relevé de l'erreur : en calculant l'écart entre la valeur mesurée Y_m et la consigne Y_c.

4. Application de la commande de régulation : pour ajuster les poids des files afin d'atteindre la valeur consigne selon le régulateur PID
5. Lancer l'algorithme d'allocation du serveur Web suivant les nouveaux poids recensés par la régulation proprement dite.
6. Attente d'un cycle de temps *m* (*wait(**m**)*).
7. Aller à 2, et refaire à nouveau le processus de commande.

Notons que le passage du mode Best effort vers le mode contrôle des délais absolus est lié aux conditions du système, et notamment la surcharge de celui-ci où le franchissement des délais de la connexion consignes ($t_i > tcxagr_i$).

4.6.3 Le contrôleur des délais absolus de connexion

Le recouvrement en FeedForward joue un rôle important dans les systèmes à commande, puisqu'à travers lui sera établi la correction du paramètre de contrôle exprimé en taux de services pour les différentes classes. Malgré cela, les perturbations éventuelles survenues sur l'entourage de serveur Web influent certainement sur ce paramètre de contrôle, et par conséquent entraînent une déficience sur notre qualité de service.

Pour établir un contrôle plus robuste de système DiffServ, nous proposons de concevoir un contrôleur Feedback adapté qui sera affecté au processus d'allocation Feedforward.

Cependant notre système de contrôle Feedback rectifie l'erreur recensée entre le délai de connexion éventuel (consigne) et le délai de connexion réel écoulé chez le serveur. A cet effet, le régulateur assigne de nouveaux poids exprimant les taux de service qui seront attribués aux différents flux.

La Figure IV-7 illustre le contrôle de la rétroaction dans le modèle de différenciation des services dont chaque classe de requêtes est alimentée par une boucle de contrôle distinct.

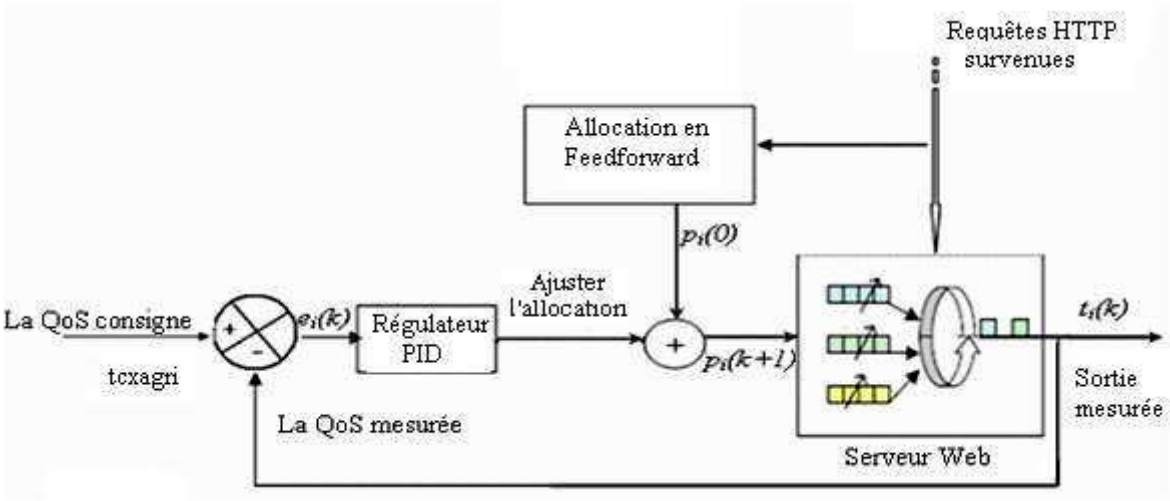


***Fig. IV-7 : Processus de contrôle* des délais absolus**

Le régulateur Proportionnel Intégral Dérivatif (PID) est l'une des techniques de conception les plus classiques de contrôle largement utilisées dans les systèmes de contrôle.

Le contrôleur PID est utilisé pour ajuster le taux de service d'une classe qui sera exprimé en nombre de processus (thread) alloués à une classe de requêtes. Le processus d'allocation s'effectue suivant l'erreur recensée entre le temps de réponse consigne et le temps de réponse moyen mesuré d'une classe de service.

Pour un système DiffServ des délais absolus à n classes de service, on a besoin de n processus de contrôle absolus (CA_i, $1 \leq i \leq N$). A chaque cycle de temps k, chacun des processus de contrôle CA_i calcule le taux de service éventuel exprimé en nombre des processus Pi(k) alloués à la classe de service i.

Notons que dans une condition de surcharge la somme des nombres éventuels des processus à allouer aux classes dépasse le nombre total des processus de serveur M: $\sum P_i(k) > M$

En conséquence les processus de contrôle CA_i, particulièrement qui sont rattachés aux classes à faibles priorités sont inexécutables parce qu'ils subissent des violations induites par le système de contrôle des délais absolus, ce qui devrait garantir les besoins des classes de services en ordre décroissant de priorité.

4.6.3.1Construction du modèle

L'analyse du système de contrôle Feedback repose sur la dépendance entre les deux paramètres d'entrée et de sortie. Le premier est le paramètre d'entrée contrôlé qui affecte le comportement du système exprimé en taux de service. Le deuxième est le paramètre de sortie mesuré qui quantifie cet effet représenté par le délai de connexion. Afin de construire le modèle, on doit identifier la dépendance de ces deux paramètres en boucle ouverte. Cette corrélation peut être exprimée par une équation différentielle linéaire qui relie la valeur actuelle de sortie aux valeurs antérieures des deux paramètres.

Le modèle en deuxième ordre indique que la valeur prochaine à prévoir de la sortie dépend des valeurs actuelles des deux paramètres ainsi que ses deux valeurs précédentes, la relation peut s'écrire:

$$y(k+1)=a_1y(k)+ a_2y(k-1)+ b_1u(k)+ b_2u(k-1)\ldots\ldots\ldots \textbf{E (IV.11)}$$

Où $y(k)$ et $u(k)$ représentent le temps de connexion et le taux de service respectivement à l'échantillon du temps k.

Ainsi que : a_1, a_2, b_1, b_2 qui sont les coefficients de la fonction de transfert qui détermine le comportement de système étudié.

La transformée en Z du modèle étudié est : $zY(z)=a_1Y(z) +z^{-1} a_2Y(z) + b_1U(z) + z^{-1} b_2U(z)$

D'où la fonction de transfert du système est donnée par :

$$G(z) = \frac{Y(z)}{U(z)} = \frac{b_1 z + b_2}{z^2 - a_1 z - a_2} \quad \ldots\ldots\ldots\ldots\ldots\ldots \textbf{E (IV.12)}$$

4.6.3.2 Formulation du régulateur Proportionnel Intégrale Dérivatif (PID)

Le contrôleur Proportionnel, Intégral et dérivatif (PID) combine les avantages du contrôleur dérivatif et intégral avec celles du contrôle proportionnel. L'avantage du contrôleur proportionnel se trouve dans sa vitesse de réponse transitoire. Notons que le choix d'un gain de contrôle proportionnel (K_P) important conduit généralement à une réponse plus rapide de contrôle, mais il accroît de plus en plus l'instabilité du système. D'autre part, le contrôleur dérivatif permet d'améliorer la stabilité du système. Cependant l'avantage du contrôleur intégral est d'éliminer l'erreur du régime statique.

Formellement, le fonctionnement du régulateur PID décrit le paramètre de contrôle $u(k)$ comme la somme des deux termes suivants:

$$u(k) = u_P(k) + u_I(k) + u_D(k)$$

Où :

$$u_P(k) = K_P e(k)$$

$$u_I(k) = u_I(k-1) + K_I e(k)$$

$$u_D(k) = K_D(e(k) - e(k-1)) = K_D \Delta e(k)$$

Donc: $u(k) = u_I(k-1) + K_I e(k) + K_p e(k) + K_D \Delta e(k)$

Pour éliminer le terme $u_I(k)$ algébriquement, il est plus simple d'écrire l'équation du contrôle $u(k)$:

$$\begin{aligned} u(k) - u(k-1) &= u_P(k) - u_P(k-1) + u_I(k) - u_I(k-1) + u_D(k) - u_D(k-1) \\ &= K_P e(k) - K_P e(k-1) + K_I e(k) + K_D \Delta e(k) - K_D \Delta e(k-1) \end{aligned}$$

Par conséquent:

$$u(k) = u(k-1) + K_P e(k) - K_P e(k-1) + K_I e(k) + K_D \Delta e(k) - K_D \Delta e(k-1) \ldots \textbf{\textit{E (IV.13)}}$$

Conformément à l'ensemble d'équations suivantes :

$$\begin{cases} u(k) = u(k-1) + K_P e(k) - K_P e(k-1) + K_I e(k) + K_D \Delta e(k) - K_D \Delta e(k-1) \\ u(k-1) = u(k-2) + K_P e(k-1) - K_P e(k-2) + K_I e(k-1) + K_D \Delta e(k-1) - K_D \Delta e(k-2) \\ u(k-2) = u(k-3) + K_P e(k-2) - K_P e(k-3) + K_I e(k-2) + K_D \Delta e(k-2) - K_D \Delta e(k-3) \\ . \\ . \\ . \\ u(2) = u(1) + K_P e(2) - K_P e(1) + K_I e(2) + K_D \Delta e(2) - K_D \Delta e(1) \\ u(1) = u(0) + K_P e(1) - K_P e(0) + K_I e(1) + K_D \Delta e(1) - K_D \Delta e(0) \\ u(0) = u(0) \end{cases}$$

Avec $e(0) = \Delta e(0) = 0$

Après, la somme de ces équations nous obtenons :

$$u(k) = u(0) + K_P e(k) + K_I \sum_{j=1}^{k} e(k) + K_D \Delta e(k) \ldots\ldots\ldots\ldots\ldots \textbf{E(IV.14)}$$

Subséquemment, on peut établir la relation qui détermine le nombre de processus alloués pour chaque classe *i* par le contrôleur Proportionnel, Intégral et Dérivatif (PID) dans le cycle de temps *k* par la formule :

$$p_i(k) = p_i(0) + K_p e_i(k) + K_i \sum_{j=1}^{j=k} e_i(j) + K_D \Delta e_i(k) \ldots\ldots\ldots\ldots\ldots \textbf{E (IV.15)}$$

$P_i(0 :)$ désigne le nombre initial de processus alloués au classe de flux *i* et qui est déterminé par: $p_i(0) = \theta_i \times M$

Où θ_i *:* représente le poids du taux de service calculé en FeedForward de la classe *i.*

M: est le nombre total des processus ou des threads offerts par le serveur Web.

Les deux autres termes ajoutés à $P_i(0)$ dans l'équation ci-dessus dénotent le composant proportionnel et intégral respectivement, avec $e_i(k)$ *:* qui est la dissimilitude entre le temps de connexion cible $tcxagr_i$ et le temps t_i de connexion mesuré pour la classe de service *i* dans le cycle de temps *k*.

4.6.3.3 Analyse des propriétés du système

La fonction de transfert du régulateur PID peut être trouvée en prenant la transformée en Z de l'équation ***E (IV.13)*** avec zéro comme conditions initiales:

$$u(k) = u(k-1) + K_P e(k) - K_P e(k-1) + K_I e(k) + K_D \Delta e(k) - K_D \Delta e(k-1)$$

$$u(k) = u(k-1) + K_P e(k) - K_P e(k-1) + K_I e(k) + K_D e(k) - K_D e(k-1) - K_D e(k-1) + K_D e(k-2)$$

La transformation en Z de cette équation est:

$$U(z) = z^{-1}U(z) + (K_P + K_I + K_D)E(z) - (K_P + 2K_D)z^{-1}E(z) + K_D z^{-2}E(z)$$

$$(1 - z^{-1})U(z) = (K_P + K_I + K_D - (K_P + 2K_D)z^{-1} + K_D z^{-2})E(z)$$

De ce fait Le gain de contrôle K(z) est :

$$\mathrm{K(z)} = \frac{U(z)}{E(z)} = \frac{(K_P + K_I + K_D - (K_P + 2K_D)z^{-1} + K_D z^{-2})}{(1 - z^{-1})}$$

$$= \frac{(K_P + K_I + K_D)z - (K_P + 2K_D) + K_D z^{-1}}{z - 1}$$

$$= K_P + K_I \frac{z}{z-1} + K_D \frac{z - 2 + z^{-1}}{z - 1}$$

$$= K_P + K_I \frac{z}{z-1} + K_D \frac{z^2 - 2z + 1}{z(z-1)}$$

$$= K_P + K_I \frac{z}{z-1} + K_D \frac{z-1}{z}$$

Donc :

$$\mathrm{K(z)} = K_P + K_I \frac{z}{z-1} + K_D \frac{z-1}{z} \quad \text{……. } \textbf{\textit{E (IV.16)}}$$

Ainsi comme le montre la Figure IV-8, l'entrée de commande est la somme des termes proportionnel, intégral et dérivatif.

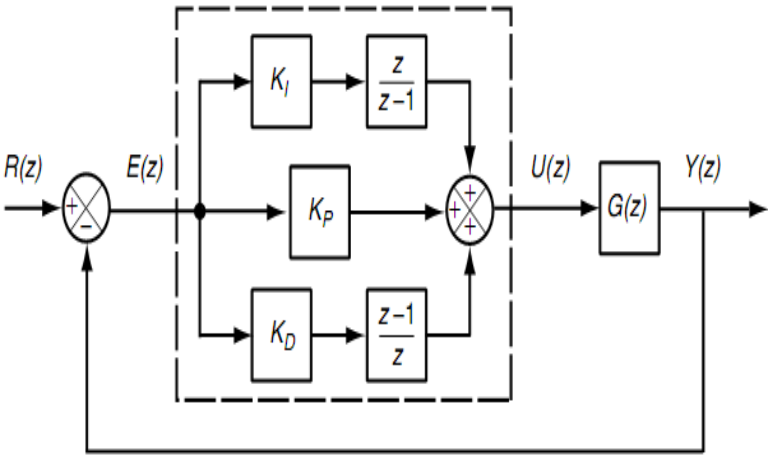


Fig. IV-8 : La boucle du contrôle Feedback PID

Dans l'étude d'un tel système de contrôle Feedback, quatre propriétés doivent être considérées. Ces propriétés expriment les performances du système à régulation automatique.

- *Stabilité du système :* le système doit être stable à la présence des perturbations survenues. Cette propriété est réalisée en veillant à ce que tous les pôles de système G(z) se situent dans le cercle unitaire.

- *Précision* : le système est précis si la valeur mesurée converge vers la valeur consigne. En partant que le contrôle PID comprend un terme du contrôle intégral, nous prévoyons que l'erreur statique doit être nulle. Cela peut être confirmé par la constatation de la fonction de transfert en boucle fermée du système de la Figure IV-8 dont la fonction de transfert de procédé G (z) se produit. La fonction de transfert $F_R(z)$ en boucle fermée est calculée à partir du gain produit du rapport Y par R déduite comme suit :

$$Y(z) = G(z).K(z).E(z)$$

$$Y(z) = G(z).K(z).(R(z) - Y(z))$$

$$Y(z)(1 + G(z).K(z)) = G(z).K(z).R(z)$$

$$Y(z)(1 + G(z).K(z)) = G(z).K(z).R(z)$$

$$F_R(z) = \frac{Y(z)}{R(z)} = \frac{G(z).K(z)}{1 + G(z).K(z)}$$

$$F_R(z) = \frac{G(z)(K_P + K_I \frac{z}{z-1} + K_D \frac{z-1}{z})}{1 + G(z).(K_P + K_I \frac{z}{z-1} + K_D \frac{z-1}{z})}$$

$$F_R(z) = \frac{G(z)((K_P + K_I)z - K_P + K_D \frac{(z-1)^2}{z})}{z - 1 + G(z).((K_P + K_I)z - K_P + K_D \frac{(z-1)^2}{z})} \quad \text{..... } \textbf{E(IV.17)}$$

Posons *: z=1* (t tend vers l'infini)

$$F_R(1) = \frac{G(1).((K_P + K_I) - K_P + 0)}{0 + G(1).((K_P + K_I) - K_P + 0)}$$

$$F_R(1) = 1$$

Ainsi l'erreur statique e_{ss} à partir du $F_R(1)$ quelque soit la valeur de consigne r_{ss} :

$$e_{ss} = r_{ss}[1 - F_R(1)] = 0$$

- *Convergence (settling time) :* présente le temps écoulé, de préférence minimum, pour que le système converge à la valeur cible.

- *Dépassement (overshoot) :* il est recommandé pour certains systèmes d'achever ces objectifs sans un dépassement ennuyé.

Les deux dernières propriétés de contrôle peuvent être atteintes par le bon choix des paramètres K_P, K_D et K_I du régulateur PID.

4.7 Conclusion

Nous avons présenté une évaluation analytique de système DiffServ des délais relatifs, et nous avons proposé le système WFQ pour remédier au problème de perte de ressource (treads inactif) dans le système DiffServ des délais relatifs.

La comparaison montre l'impropriété des systèmes DiffServ des délais relatifs ou tout autre système basé sur la désagrégation de taux de service (exprimé en nombre de processus ou de thread), puisqu'ils ne donnent pas de QoS meilleure que le Best Effort, et qui est une réfutation des études d'optimisation de QoS des serveurs Web adoptés dans **[11]**, **[17]**, et **[32]**.

Pour l'amélioration de QoS du serveur Web, nous avons adopté une politique hybride de processus d'allocation du serveur Web. Cette politique adoptée est établie sur deux axes suivant la charge du système.

Sur le premier axe où le facteur d'utilisation globale du serveur n'est pas grand, on applique le Best effort.

Sur le deuxième axe où le facteur d'utilisation globale du serveur est assez important, l'établissement du Best effort peut défaillir la QoS. D'où le fait indispensable de remplacer le processus antérieur par un processus de contrôle de système DiffServ des délais absolus.

CHAPITRE V

SIMULATION ET INTERPRETATION DES RESULTATS

5.1 Introduction

Nous avons travaillé avec MATLAB qui est à la fois un langage de programmation et un environnement de développement. Pour le procédé du système proposé, on s'est basé sur les deux modules Simulink et Simevent intégrés à MATLAB, qui comportent tous les outils et ToolBox nécessaires à notre simulation.

Simulink est une plate-forme de simulation multi-domaines de modélisation de systèmes dynamiques. Il fournit un environnement graphique et un ensemble de bibliothèques contenant des blocs de modélisation qui permettent le design précis, la simulation, et le contrôle, ainsi qu'un accès immédiat aux nombreux outils de développements algorithmiques, de visualisation et d'analyse de données de MATLAB. Simevent étendu de Simulink, comporte les outils de simulation à événements discrets pour les transactions entre les composants d'un système. Nous pouvons utiliser ce module pour analyser les caractéristiques de performance comme les délais de connexion de bout en bout, le débit et la perte de paquets. Des bibliothèques et des blocs prédéfinis, tels que les files d'attente, les serveurs, commutateurs, générateur d'événement pour représenter les composants de l'architecture d'un tel système.

Les architectures des modèles adoptées pour la simulation sont présentées dans l'appendice A à la fin de ce mémoire.

En premier lieu, on doit déterminer les paramètres des régulations, ensuite des simulations de : « DiffServ des délais relatifs vis-à-vis de Best effort », « Contrôle des délais absolus de connexion», « Système adopté », et une interprétation des résultats. La simulation de contrôle des délais absolus de connexion est établie sur deux classes et trois classes de services. Finalement, on examine le système de contrôle hybride proposé.

5.2 Perception des Paramètres de contrôle PID

Le système DiffServ des délais absolus est en deuxième ordre, d'après l'équation E (IV.12), la fonction de transfert du système est donnée par : $G(z) = \frac{Y(z)}{U(z)} = \frac{b_1 z + b_2}{z^2 - a_1 z - a_2}$

Pour l'identification du système, nous nous basons sur les résultats mentionnés par C. Lu et all **[32]**, où ils proposent plusieurs fonctions de transfert pour ce modèle. L'une de ces fonctions de transfert est en deuxième ordre, et apporte un système stable avec une excellente précision (à l'ordre de R^2=92%) : $G(z) = \frac{0.36z + 0.15}{z^2 - 0.14z + 0.05}$

Pour calculer les coefficients de contrôle PID, On a suivi la méthode de Ziegler & Nichols qui consiste à mettre en oscillations entretenues la boucle de régulation. A partir du gain G_{rc} qui a permis d'obtenir cette oscillation, et de la période T de cette oscillation, il est possible de choisir les paramètres de réglage du régulateur. La Figure V-1 montre le système en oscillations si on établit G_{rc}=17/3, et la période de cette oscillation est T=10s.

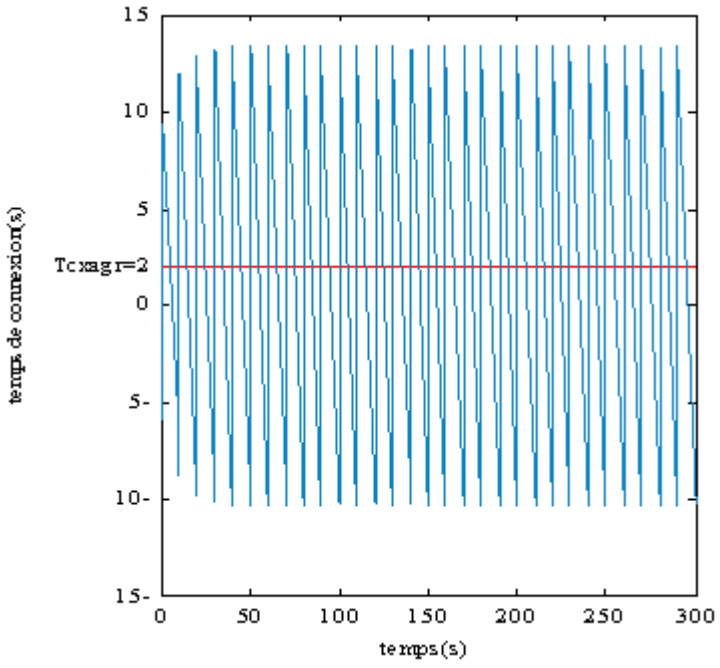


Fig. V-1: Choix de paramètres par la méthode de Ziegler & Nichols

D'où : les coefficients de contrôleur PID sont :

$$K_p = \frac{G_{rc}}{1.7} = 3.3\ ,\ K_i = \frac{1}{T_i} = \frac{1}{\frac{2.T_{osc}}{G_{rc}}} = 0.7\ ,\ K_d = T_d = \frac{T_{osc}.G_{rc}}{13.3} = 4.3\ .$$

La Figure V-2, représente un contrôle des délais absolus de connexion à deux classes, dont on perçoit clairement l'effet de régulation du contrôleur PID. Les valeurs délais désirées de connexions choisies sont 2 secondes et 5 secondes, pour la première classe et la deuxième classe respectivement. En outre, la Figure V-3 représente l'effet de régulation du contrôleur PID en présence des perturbations sur l'entourage du système.

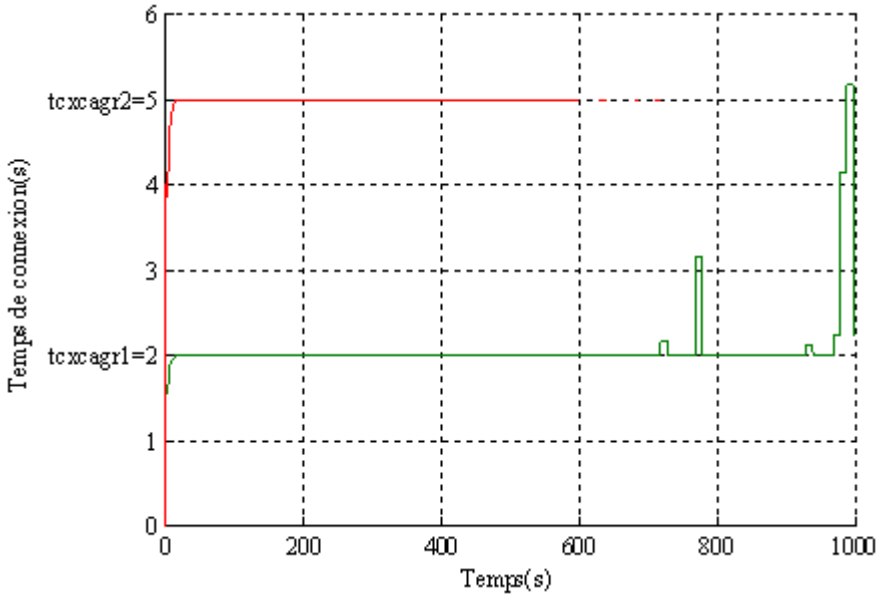


Fig. V-2: L'effet de la régulation du contrôleur PID

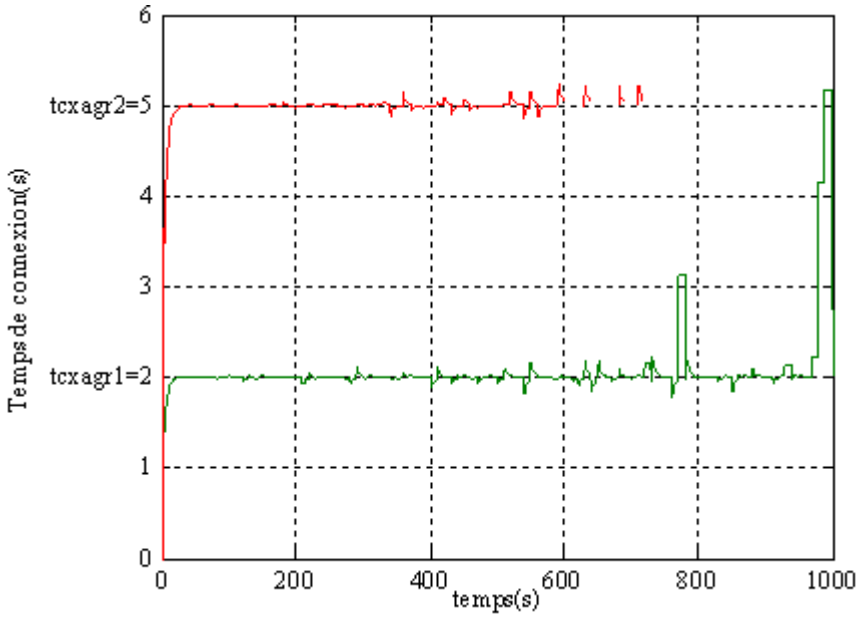


Fig. V-3: L'effet des perturbations sur la régulation du contrôleur PID

5.3 Le modèle DiffServ des délais relatifs vis-à-vis de Best effort

Nous montrons dans la Figure V-4 ci-après la validité des résultats obtenus de l'évaluation analytique par le module Simevent de MATLAB. L'architecture du modèle est exposée à l'appendice A. La génération des entités, qui représentent les requêtes survenues au système, suit un processus de Poisson avec une moyenne λ, tandis que le temps de service est un processus exponentiel avec une moyenne μ. Les taux de service par classe sont calculés par la formule E(IV.5). Les paramètres des tests sont indiqués dans le Tableau V-1 ci dessous:

Test		Taux de génération des entités	poids	les temps de connexions théoriques
Test1	Classe 1	0.25	5	2.4
	Classe 2	0.25	1	12
	Best effort	0.5	/	2
Test2	Classe 1	0.25	3	5.33
	Classe 2	0.5	1	16
	Best effort	0.75	/	4
Test3	Classe 1	0.5	3	5.33
	Classe 2	0.25	1	16
	Best effort	0.75	/	4
Test4	Classe 1	0.05	2	1.76
	Classe 2	0.10	1	3.53
	Best effort	0.15	/	1.18

Tab. V-1 : Paramètres de la simulation de DiffServ des délais relatifs et de Best effort

Les graphes de la simulation montrent que les temps de connexion correspondent aux temps de connexions obtenues théoriquement, ainsi que la cause de favorisation du modèle Best effort par apport au modèle DiffSev des délais relatifs. En outre, nous remarquons que les graphes des Test2 et Test3 sont presque les mêmes, ce qui affirme que le temps de connexion relatif d'une classe est indépendant de son taux d'arrivée.

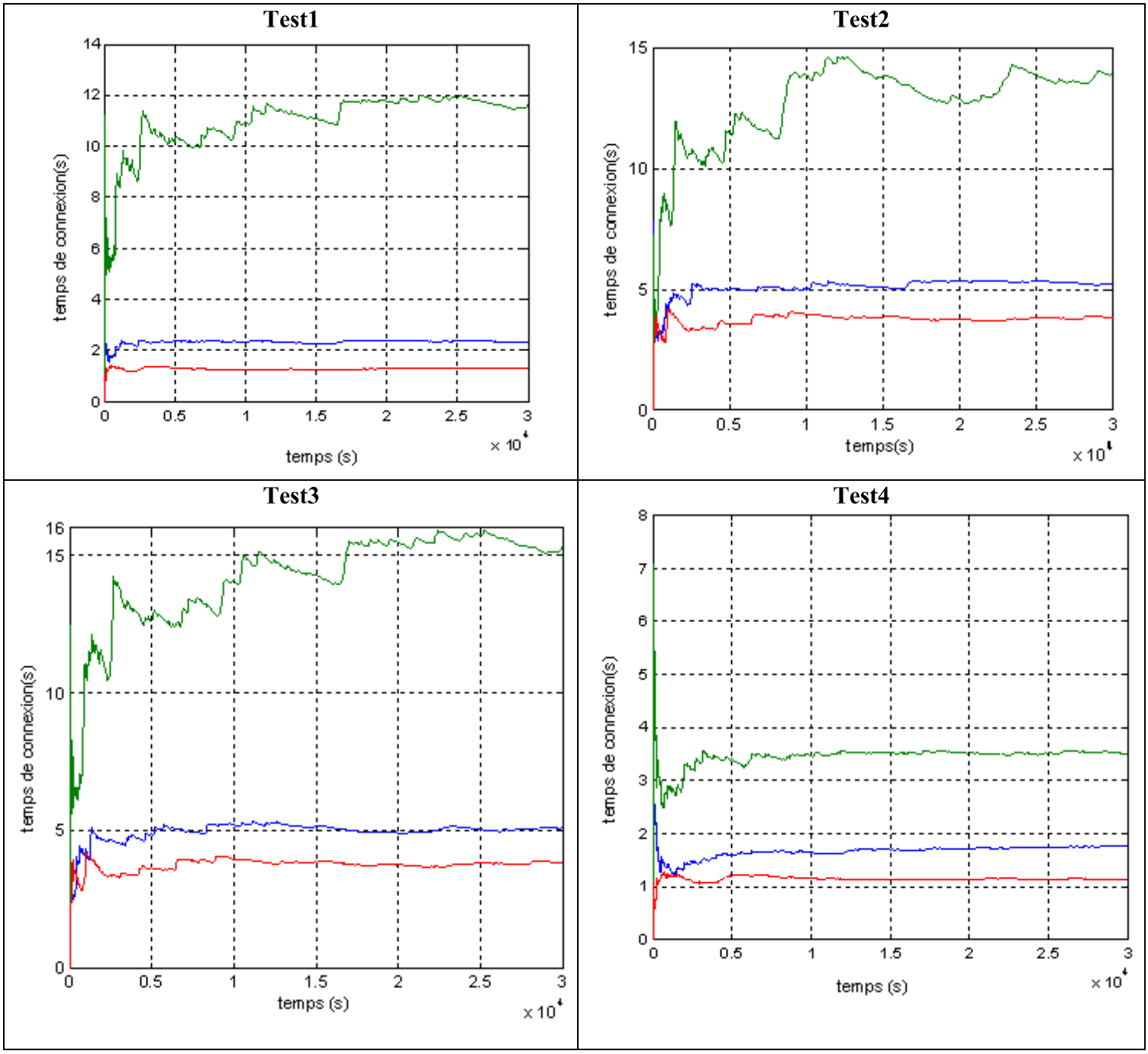


Fig. V-4: **Le modèle DiffServ des délais relatifs vis à vis du Best effort**

5.4 Simulation du système proposé et interprétation des résultats

Pour l'interprétation des résultats de la simulation pour le système de contrôle des délais de connexion, nous avons établi nos graphes en fonction du taux de charge du système, qui représente le taux des requêtes survenues au système.

5.4.1 Le contrôle des délais absolus de connexion à deux classes de service

On a établi la simulation pour deux classes, et on a fixé les valeurs maximales des délais de connexions, pour la première classe à 2 secondes, et à 5 secondes pour la deuxième. D'autre part on a établi la valeur de TimeOut du serveur Web à 7 secondes.
Les valeurs des taux des requêtes survenues sont prises aléatoirement en fonction du taux de charge du système.

La Figure V-5 représente les graphes du choix des taux des requêtes survenues aux deux classes à l'aide de la fonction aléatoire Random, le taux global varie de 0 à 1 suivant le taux de charge du système.

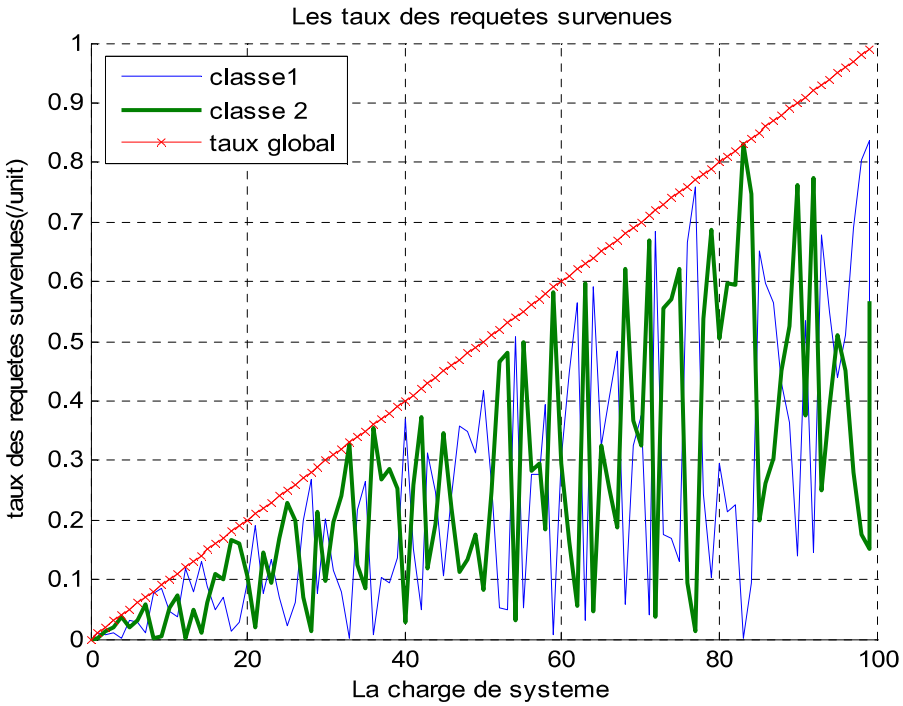


Fig. V-5 : Le choix des taux de requêtes survenues aux deux classes

La Figure V-6 présente les graphes du taux des services éventuels (nécessaire pour les temps de connexions qui sont comme prévu 2 secondes et 5 secondes pour les deux classes respectivement), en fonction de la charge du système, et en tenant compte des valeurs des taux des requêtes survenues pour chaque classe (Figure V-5).

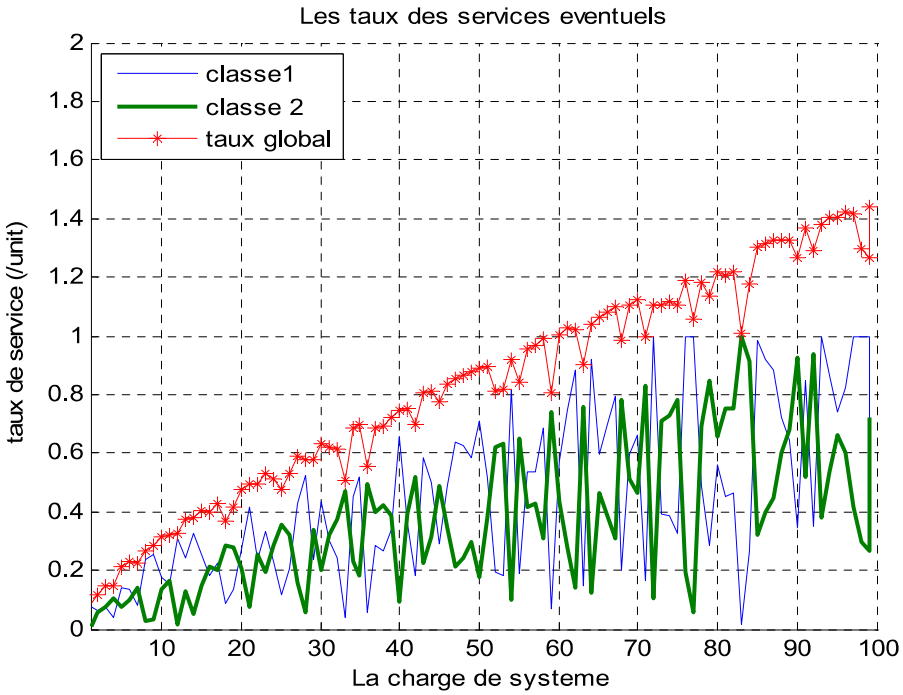


Fig. V-6 : Les taux des services éventuels du contrôle des délais absolus à deux classes

Nous pouvons constater sur la Figure V-6, que quand le système n'est pas en surcharge à l'ordre moins du seuil 60%, la somme des taux des sévices éventuels (indiqué par le taux global sur la figure) ne dépasse pas la capacité unitaire 1 du serveur Web. Par conséquent le contrôle doit être maintenu aux délais désirés, contrairement à cela dans le cas où la charge du système est supérieure à ce seuil.

La Figure V-7 décrit les graphes du temps de connexion généré par le contrôle des délais absolus des classes en fonction du taux de charge du système et sur la base de la Figure V-6.

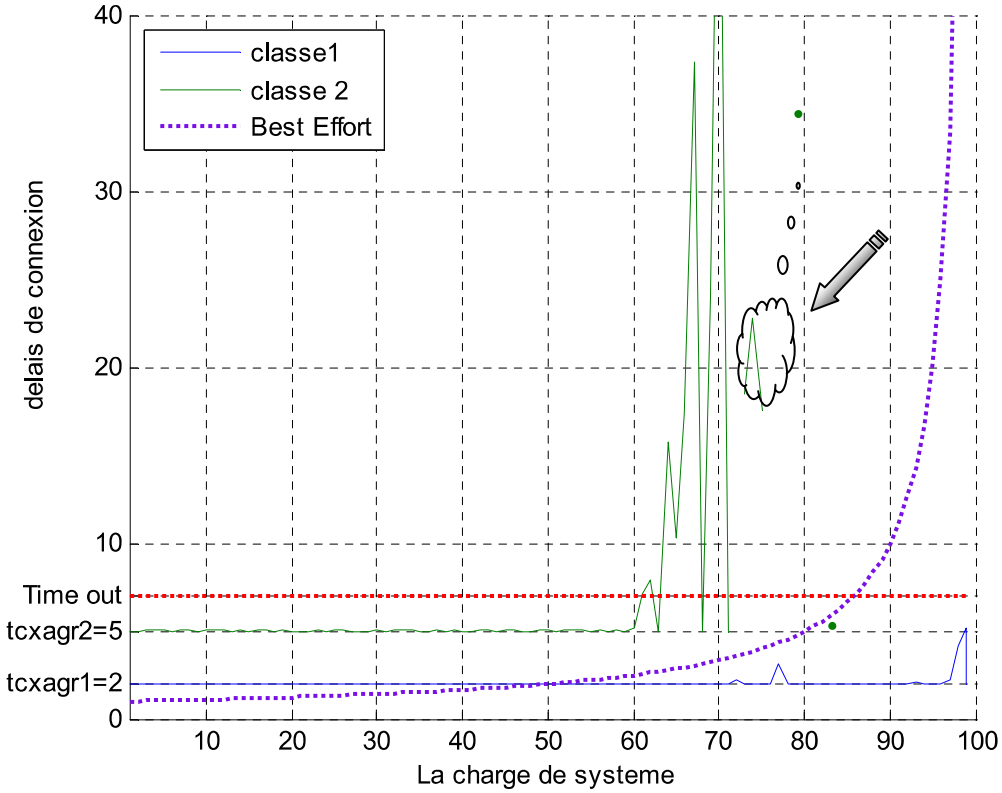


Fig. VI-7 : le contrôle des délais absolus à deux classes

En premier lieu, quand le système n'est pas en forte surcharge (moins de 60%), on remarque que les temps de connexions prédéfinis (2 secondes et 5 secondes) des deux classes sont respectés. Après, les requêtes de la classe 2 sont retardées et risquent d'être rejetées (selon la valeur de TimeOut du serveur Web, 7 seconde dans notre exemple). Quand le système est en surcharge (plus de 70%), le contrôle des délais absolus de connexions offre plus de performance aux classes prioritaires (classe 1), au le détriment de la classe 2.

La zone marquante signalée par une flèche à la Figure V-7, signifie qu'on peut servir des requêtes de classe 2, sans bien sûr tenir compte du temps Timeout prédéfini pour le serveur Web.

Nous remarquons que si la charge de système est supérieure à 85%, le contrôle des délais absolus offre plus de performance à la classe 1. D'autre part Le Best Effort arrive à inhiber toutes les requêtes survenues aux systèmes puisque le TimeOut du serveur web est de 7 secondes.

5.4.2 Le contrôle des délais absolus à trois classes de service

La simulation ici est pour trois classes, et on a fixé les valeurs maximales des délais de connexions, pour la première classe à 2 secondes, et à 5 secondes pour la deuxième et 8 seconde pour une autre classe. La valeur de TimeOut du Serveur Web est établie à 10 secondes.

De même que la Figure V-5, La Figure V-8 représente les graphes du choix des taux de requêtes survenues aux trois classes à l'aide de la fonction aléatoire Random.

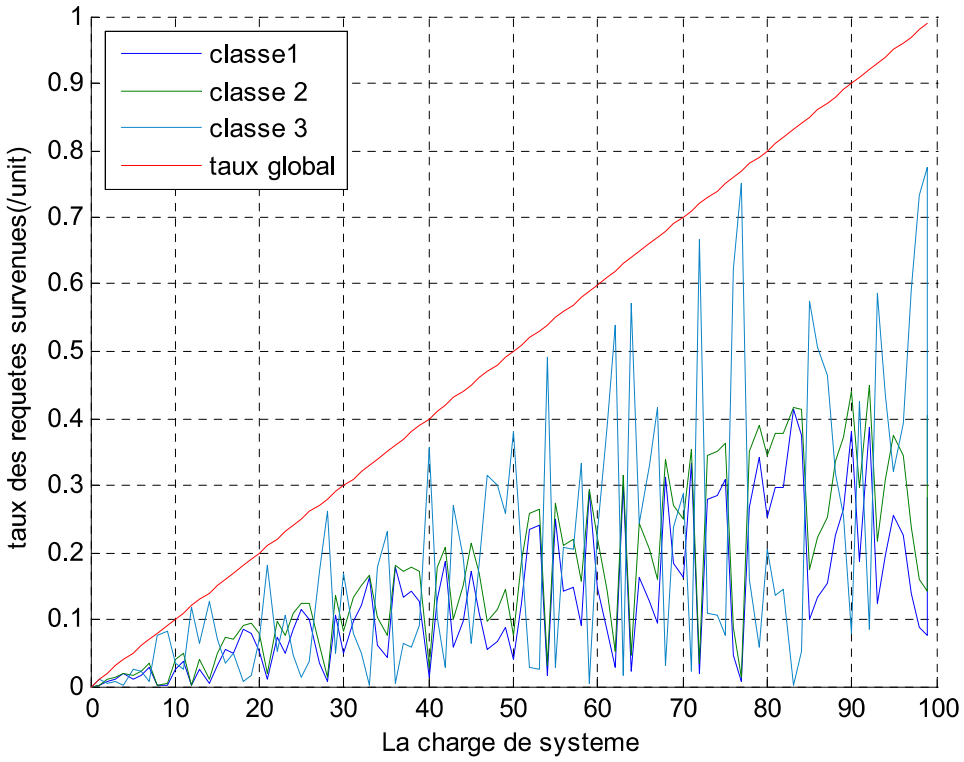


Fig. V-8 : Le choix des taux de requêtes survenues aux trois classes

Egalement, nous constatons ici, sur la Figure V-9 ci-après, qui représente les graphes de taux des services éventuels, quand le système n'est pas en surcharge à l'ordre moins du seuil 60%, la somme des taux des services éventuels ne dépasse pas la capacité unitaire 1 du serveur Web. Par conséquent le contrôle doit être maintenu aux délais voulus.

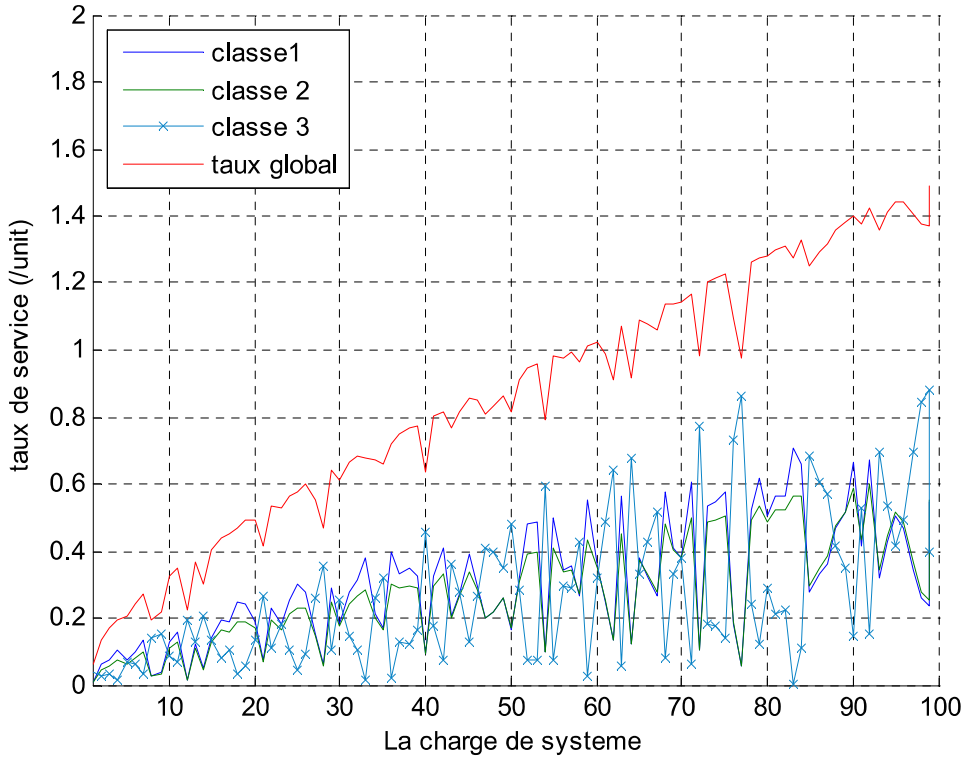


Fig. V-9 Les taux des services éventuels du contrôle des délais absolus à trois classes

La Figure V-10 décrit les graphes du temps de connexion généré par le contrôle des délais absolus des trois classes en fonction du taux de charge du système et sur la base de la Figure V-9.

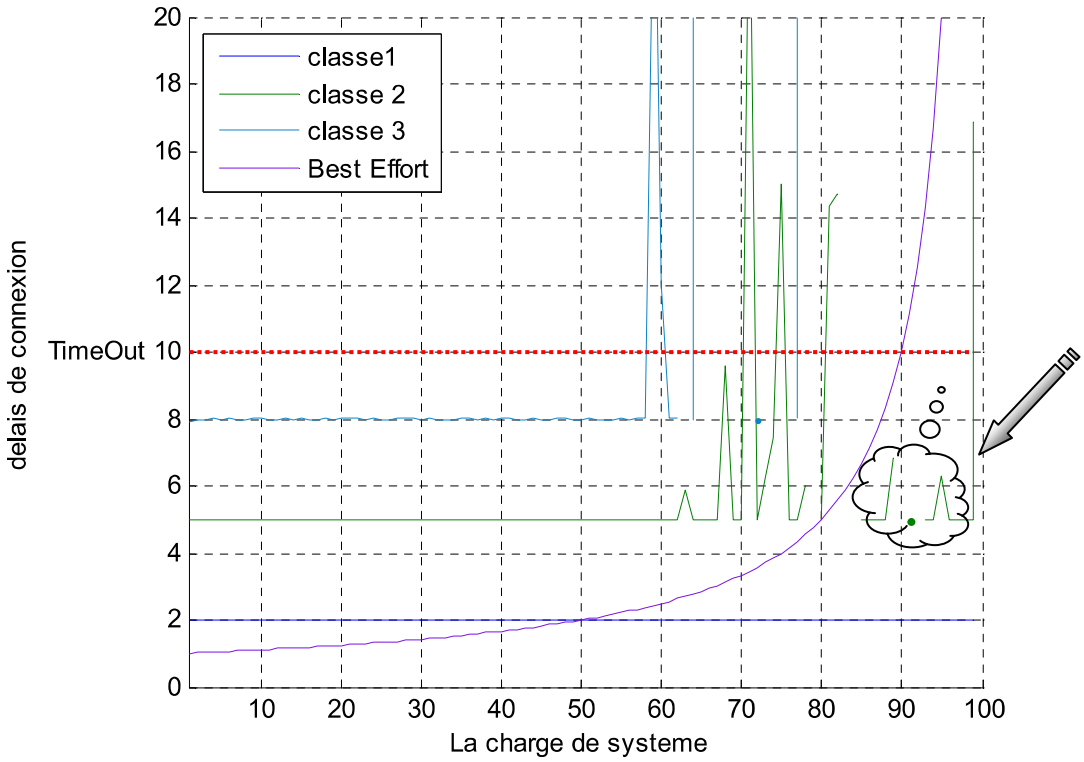


Fig. V-10 : Le contrôle des délais absolus à trois classes

Nous constatons sur la Figure V-10, que lorsque le système n'est pas en forte surcharge (moins de 60%), les temps de connexions prédéfinis (2 secondes, 5secondes, 8 secondes) des trois classes sont respectés. Après, les requêtes de la classe 2 sont retardées et risquent d'être rejetées (TimeOut du serveur web est 10 secondes dans cet exemple).

Quand le système est en surcharge (plus de 70%) le contrôle des délais absolus offre plus de performance aux classes prioritaires (classe 1) malgré la famine de la classe 1 et au détriment de la classe 2.

Nous remarquons que si la charge de système est supérieure à 90%, le contrôle des délais absolus offre plus de performance aux classes prioritaires, classe 1 et parfois classe 2 (la zone marquante signalée par une flèche) selon le besoin. D'autre part Le Best effort est déconseillé puisque le serveur Web rejettera toutes les requêtes survenues aux systèmes (TimeOut du serveur web est 10 secondes).

5.4.3 Le Système de contrôle hybride

Dans les conditions où le système n'est pas en surcharge, on peut constater que le Best Effort apporte plus de performance. Donc on n'a plus besoin de mettre le contrôle des délais absolus puisque le système apporte des délais admissibles pour toutes les classes sans restrictions.

Cependant le contrôle des délais absolus est indispensable si le système est en surcharge, notamment quand les taux des requêtes survenues des classes prioritaires est faible, où on peut les satisfaire carrément dans ces délais désirés. Et de temps en temps, il est possible de desservir d'autres classes.

Pour avoir une meilleure performance du serveur Web, nous avons indiqué que notre système adopté travaille sur deux axes selon les conditions de système. Donc notre système de contrôle hybride est un modèle qui combine le Best effort et le contrôle des délais absolus.

On débute par le Best effort puis, et dés que la charge de système dépasse un seuil, on applique le contrôle des délais absolus. Le problème qui se pose est de bien définir le moment où réside le point de changement pour que le système change d'une politique vers l'autre et vis versa. Ce point de changement est lié à la tolérance des délais de connexion désirés des classes.

La Figure V-11 décrit les graphes du temps de connexion généré par le système de contrôle hybride adopté qui combine le contrôle des délais absolus et le Best effort sur la base du taux de charge du système pour les deux classes. Le point de changement est fixé à l'ordre de 75%

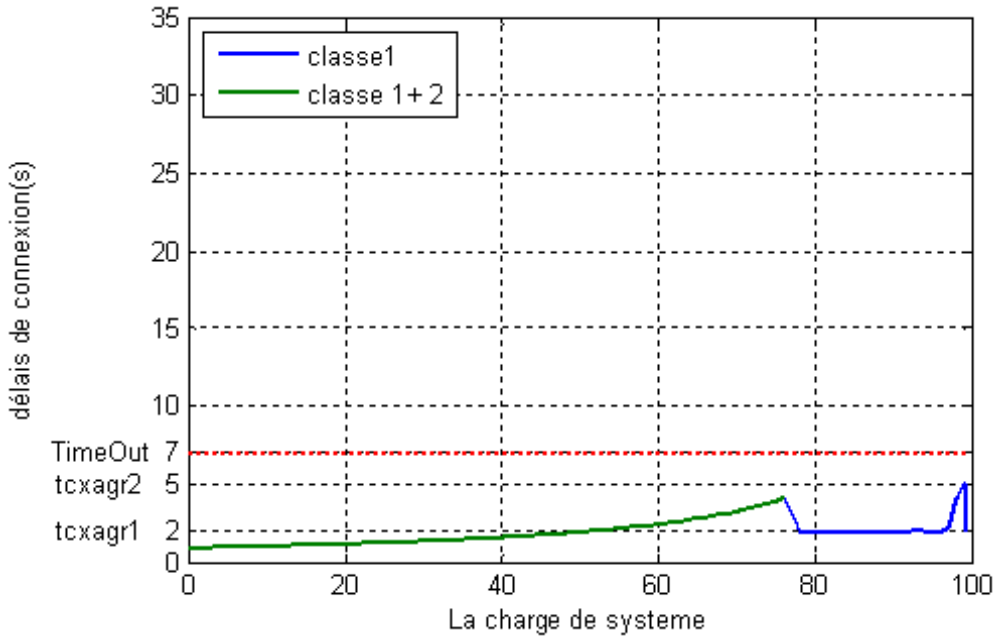


Fig. V-11 : Activités des deux classes imposées par le système de contrôle hybride

Comme le montre la Figure V-11 (déduite de la Figure V-7), où la classe 1 est satisfaite avec des délais meilleurs que son délai désiré (2 seconde) jusqu'à la surcharge du système qui est de l'ordre de 50%. Après le délai de connexion de la classe commence à défaillir jusqu'à 4 secondes, puis se maintenu à son délai de connexion voulus. La défaillance de délai de connexion de la classe 1 est liée au choix de point de changement qui est de l'ordre de 75%.

Pour la classe 2, elle est toujours satisfaite avec des délais meilleurs que son délai désiré jusqu'à ce que la charge du système arrive au seuil de 75%, la classe entre dans une famine.

La Figure V-12 (déduite de la Figure V-10) décrit les graphes des délais de connexion générés par le système de contrôle hybride pour trois classes.

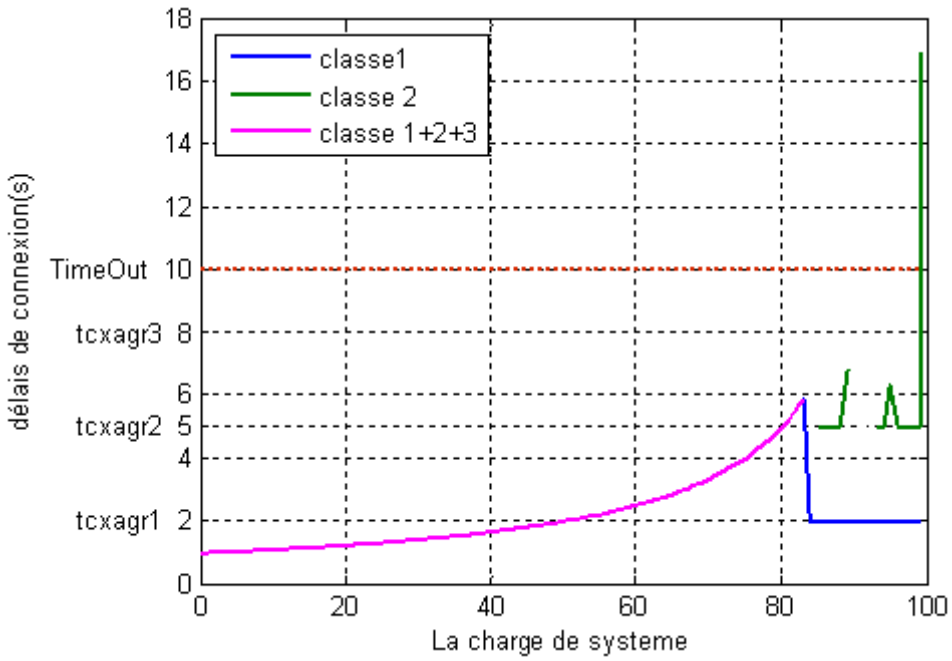


Fig. V-12 : Activités des trois classes imposées par le système de contrôle hybride

La Figure V-12 montre le point de changement qui est défini à l'ordre de 85%. La classe 1 est satisfaite avec des délais meilleurs que son délai désiré (2 seconde) jusqu'à la surcharge du système qui est de l'ordre de 50%. Après, son délai de connexion commence à défaillir jusqu'à 6 secondes, puis elle se maintient à son délai de connexion voulu.

Pour la classe 2, elle est toujours satisfaite avec des délais meilleurs que son délai désiré jusqu'à ce que la charge de système arrive au point de changement. Là, la classe entre dans une incertitude de satisfaction causée par les flux de requêtes survenues, la classe peut être servie si le taux de flux de la classe 3 et assez grand.

Pour la classe 1, elle est toujours satisfaite avec des délais meilleurs que son délai désiré jusqu'à ce que la charge du système arrive au point de changement où la classe entre dans une famine.

CONCLUSION

Dans ce travail nous avons récapitulé et étudié les techniques qui existent pour l'allocation de serveur Web en utilisant les politiques classiques ou avec les contrôles automatiques.

Notre but principal a été d'exploiter l'avantage de la technique de la régulation automatique. Nous avons donc proposé une solution d'amélioration de la qualité de service basé sur l'utilisation des lois de contrôle automatique dans le cas d'un serveur Web.

Du fait que le Best effort conduit à des délais de connexion intolérables pour les services prioritaires dans le cas de surcharge, il a fallu établir une autre politique d'allocation de services où le temps de connexion doit être meilleur qu'avec l'utilisation du Best effort.

Notre évaluation analytique des modèles existants nous a permis de réaliser une comparaison entre ces modèles. Cette dernière permet de justifier le choix du contrôle des délais absolus comme la technique la plus commode lors des cas de surcharge.

Nous avons aussi, mis en évidence le mauvais comportement des systèmes de type 'DiffServ des délais relatifs', ou toute autre sorte de systèmes basés sur la désagrégation des taux de service (exprimée en nombre de processus ou de thread), puisqu'il ne donne pas une meilleure QdS par apport au Best Effort.

Ainsi notre technique adoptée est d'employer le Best Effort pendant la non surcharge du système, et dans les conditions contraires, on bascule vers le contrôle absolu des délais de connexions. Dans ce dernier modèle, nous avons perçu l'effet de la régulation par rétroaction sur les délais de connexion des classes afin d'achever les délais désirés en ordre de priorité.

L'environnement MATLAB utilisé pour notre simulation, a fourni des phases de développement à l'implémentation d'un système de contrôle, au but d'améliorer la QoS des serveurs Web.

Comme perspective, on pourrait appliquer la politique DiffServ des délais relatifs mais sans la désagrégation du serveur Web, en ajoutant un module d'allocation du serveur basé uniquement sur la relativité de délais de connexion de classes des services. Dans ce cas là, le serveur Web établit un contrôle sur le choix des requêtes d'une classe à traiter à la base de la relativité de délais moyennes de connexion des classes.

En outre, nous pouvons proposer une autre technique qui est celle des files d'attente à seuils, décrite dans le troisième chapitre. Ce système de différenciation des services définit pour chaque classe de service un point de seuil sur sa file d'attente et permet de décrire sa priorité. L'ajustement dynamique de ces points seuils, c.à.d. en introduisant des seuils dynamiques variés selon les diverses sollicitations des services chez un serveur Web, et peut être aussi dans un cadre d'étude pour le contrôle réactif dans le but d'améliorer la QdS du serveur Web.

Enfin, une application de méta-heuristiques pour l'optimisation des gains du régulateur PID, pourrait améliorer les performances de la régulation

REFERENCES

1. Xiangping Chen & Prasant Mohapatra, « Performance Evaluation of Service Differentiating Internet Servers ». IEEE Transactions on Computers,Vol.51, No.11 , Nov 2002, pp.1368-1375

2. Luc Malrait, Sara Bouchenak, Nicolas Marchand "Modélisation et contrôle d'un serveur" RenPar'19 / SympA'13 / CFSE'7, Toulouse, septembre 2009.

3. North American Systems International Inc. The True Cost of Downtime. http://www.nasi.com/downtime cost.php. , 2008.

4. Mohammad Mamunur Rashid « Provisioning and Controling Differentieted Quality of Service in Web Servers: An Analytical Framework ». Master of science. Department of Electrical &Computer Engineering university of Manitoba Canada 2004

5. S.Blake, D.Black,M.Carlson,E. Davies,Z. Wang and W. Weiss « An Architecture for Differentiated Service » . RFC 2475 . December 1998

6. T. F. Abdelzaher, J. A. Stankovic, C. Lu, R. Zhang, and Y. Lu « Feedback Performance Control in Software Services », IEEE Control Systems, 23(3), June 2003.

7. Joseph L. Hellerstein,Yixin Diao, Sujay Parekh,Dawn M. Tilbury « Feedback Control of Computing Systems » Copyright © 2004 Published by John Wiley & Sons, Inc., Hoboken, New Jersey.

8. http://fr.wikipedia.org/wiki/Internet

9. T. Berners-Lee, R. Fielding, H. Frystyk « Hypertext Transfer Protocol -- HTTP/1.0 » May 1996.

10. Fontaine Rafamantanantsoa, Patrice Laurençot, Alex Aussem « Analyse, Modélisation et Contrôle en temps réel des performances d'un serveur web » Rapport de recherche LIMOS/RR-05-06 université Blaise Pascal, BP 10125, 63173 Aubière, France.

11. Chenyang Lu « Feedback Control Real-Time Scheduling » Doctor Philosophy these in Computer Science. university of Virginia, 2001

12. Vasile-Marian Scutirici « Utilisation efficace des serveurs web en tant que serveurs vidéo pour des applications vidéo à la demande » thèse doctorat, université Lumiere Lyon2, décembre 2001.

13. R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, « Hypertext Transfer Protocol -- HTTP/1.1» IETF RFC 2616, June 1999. http://www.ietf.org/rfc/rfc2068.txt

14. Carr RW. « Virtual Memory Management ». UMI Research Press, Ann Arbor, MI, 1984.

15. Xue Liu, Lui Sha, Yixin Diao, Steven Froehlich, J.L.Hellerstein, and Sujay Parekh « Online Response Time Optimization of Apache Web Server» IWQoS, June 2-4, 2003, LNCS 2707, pp. 46–478, Springer-Verlag Berlin Heidelberg.

16. Fateh Guenab « Contribution aux systèmes tolérants aux défauts : Synthèse d'une méthode de reconfiguration et/ou de restructuration intégrant la fiabilité des composants », thèse Doctorat de l'université Henri Poincaré, Nancy 1, 2007.

17. Xiaobo Zhou, Yu Cai , Edward Chow « An integrated approach with feedback control for robust Web server QoS design ». Computer Communications 29 (2006) 358–369.

18. Antoine Mahul « Apprentissage de la Qualité de Service dans les réseaux multiservices ». Thèse Doctorat, université Blaise Pascal - Clermont-Ferrand II- 2005-

19. Titica Mariana « Commande des Procédés (Automatique, Contrôle, Régulation) » cours d'école doctorale Energétique université de Nantes, 2005.

20. Joseph J. Distefano, Allen R. Stubberud, Ivan J. Williams « Systèmes asservis Cours et problèmes » 2ième édition, Série Schaum, 1994

21. « La régulation industrielle Les notions de base ». FICHE N°25 '' Mesure '' : revue technique de l'instrumentation et des automatismes industriels, Article publié dans le numéro 608. juin 1989.

22. Mohamed Karim Fellah « Cours d'asservissement linéaires continus » Université Sidi Bel-Abbès, 2003

23. Patrick prouvost, « Automatique, contrôle et régulation » Edition Dunod. ©2004

24. Bénédicte Vatinlen « Optimisation du routage dans les réseaux de télécommunications avec prise en compte de la qualité de service ». Thèse Doctorat, université Paris VI, 2004.

25. Alan Reugg « Processus stochastiques » Presses polytechniques romandes. Lausanne, Suisse.©1989

26. Hakim Badis « Linux Introduction à la qualité de service sous Linux ». Support de cours 2007.professeur à université de Paris-Est.

27. E. Gelenbe and G. Pujolle « Introduction aux réseaux de files d'attente » édition Eyrolles, 1980

28. D. Gauchard « Simulation hybride des réseaux IP-DiffServ-MPLS multiservices Sur un environnement d'exécution distribué. ». Thèse de doctorat de l'université Paul Sabatier 2003.

29. B. Baynat. « Théorie des files d'attente ». Editeur(s) : Hermès – Lavoisier 1ère édition, 2000.

30. Babylon8 : ver 8 Dictionnaire en ligne (glossaire, traducteur…..)

31. C.Bockstal, J.M.Garcia, and O.Brun. « Approximation du régime stationnaire d'un systeme WFQ ». Algotel(6ème Rencontres Francophones sur les aspects Algorithmiques des Télécommunication), 2004. LAAS-CNRS 31077 Toulouse, France.

32. C. Lu, T.F. Abdelzaher, J.A. Stankovic, and S.H. Son, « A Feedback Control Architecture and Design Methodology for Service Delay Guarantees in Web Servers », IEEE Transaction parallel Distributed System, Vol 7, No 9 SEPT 2006.

33. J. Hyman, A. A. Lazar, and G. Pacifici. « Joint Scheduling and Admission Control for ATS-based Swit-ching Nodes ». In ACM SIGCOMM), Baltimore, MA, August 1992

34. V. Misra, W. B. Gong, and D. Towsley. « Fluid-based analysis of a network of AQM routers supporting TCP flows with an application to RED ». Proceedings of ACM SIG-COMM '00, 2000.

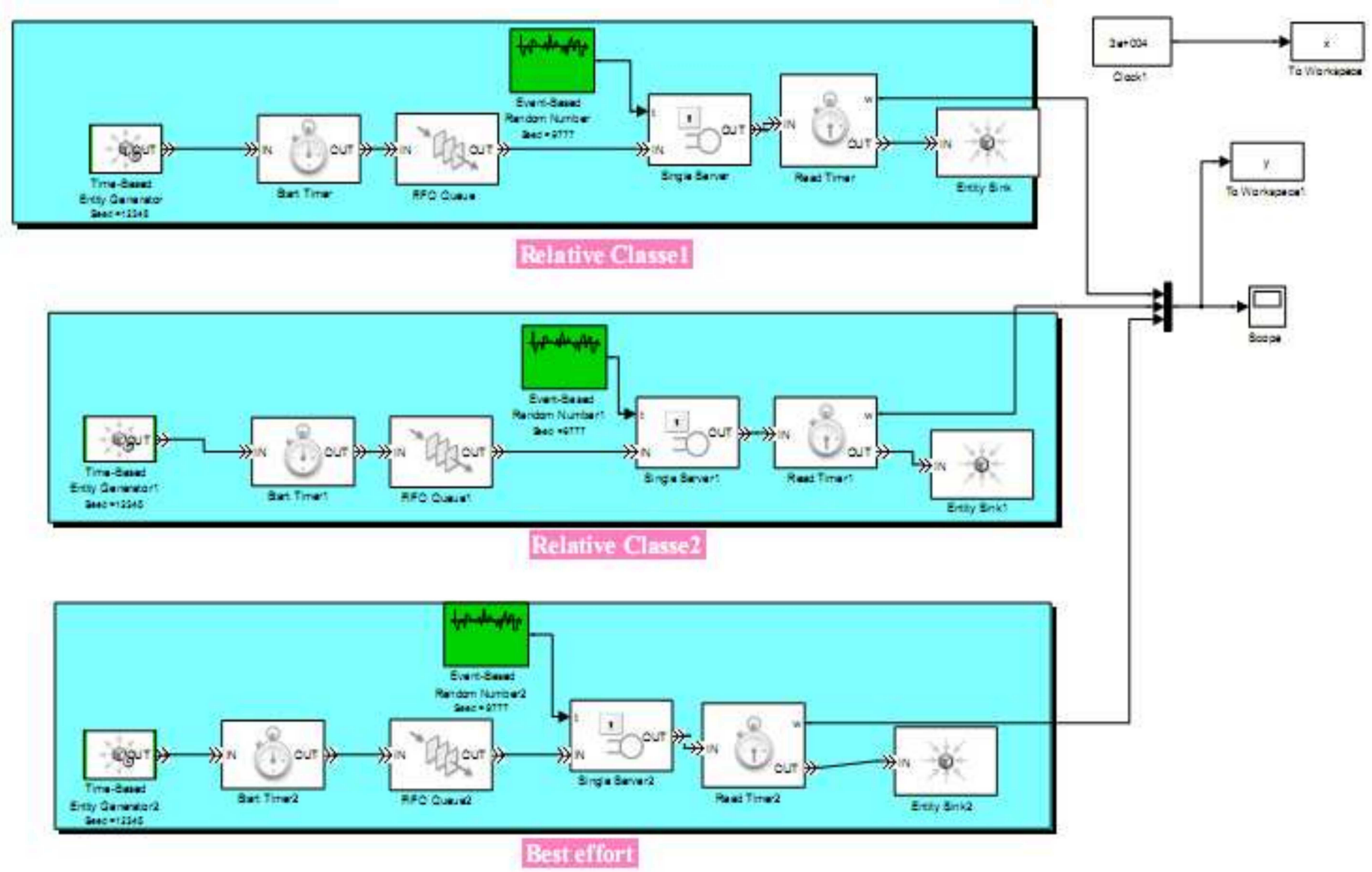
35. Y. Lu, A. Saxena, and T. F. Abdelzaher « Differentiated caching services: a control theoretic approach ». International Conference on Distributed Computing Systems, April 2001.

APPENDICE A

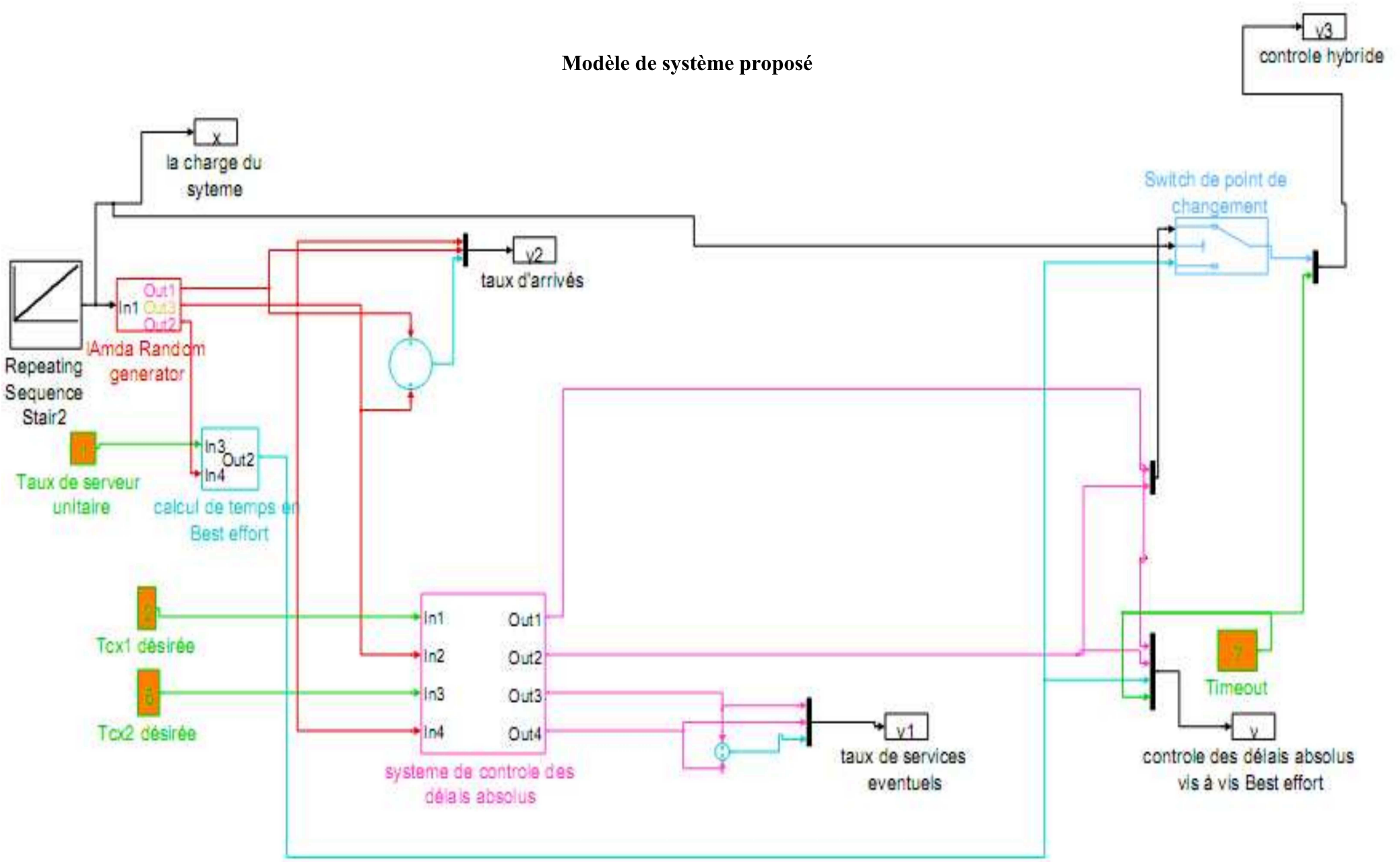
LES MODELES EMPLOYES DANS SIMULINK

POUR LE SYSTEME ADOPTE

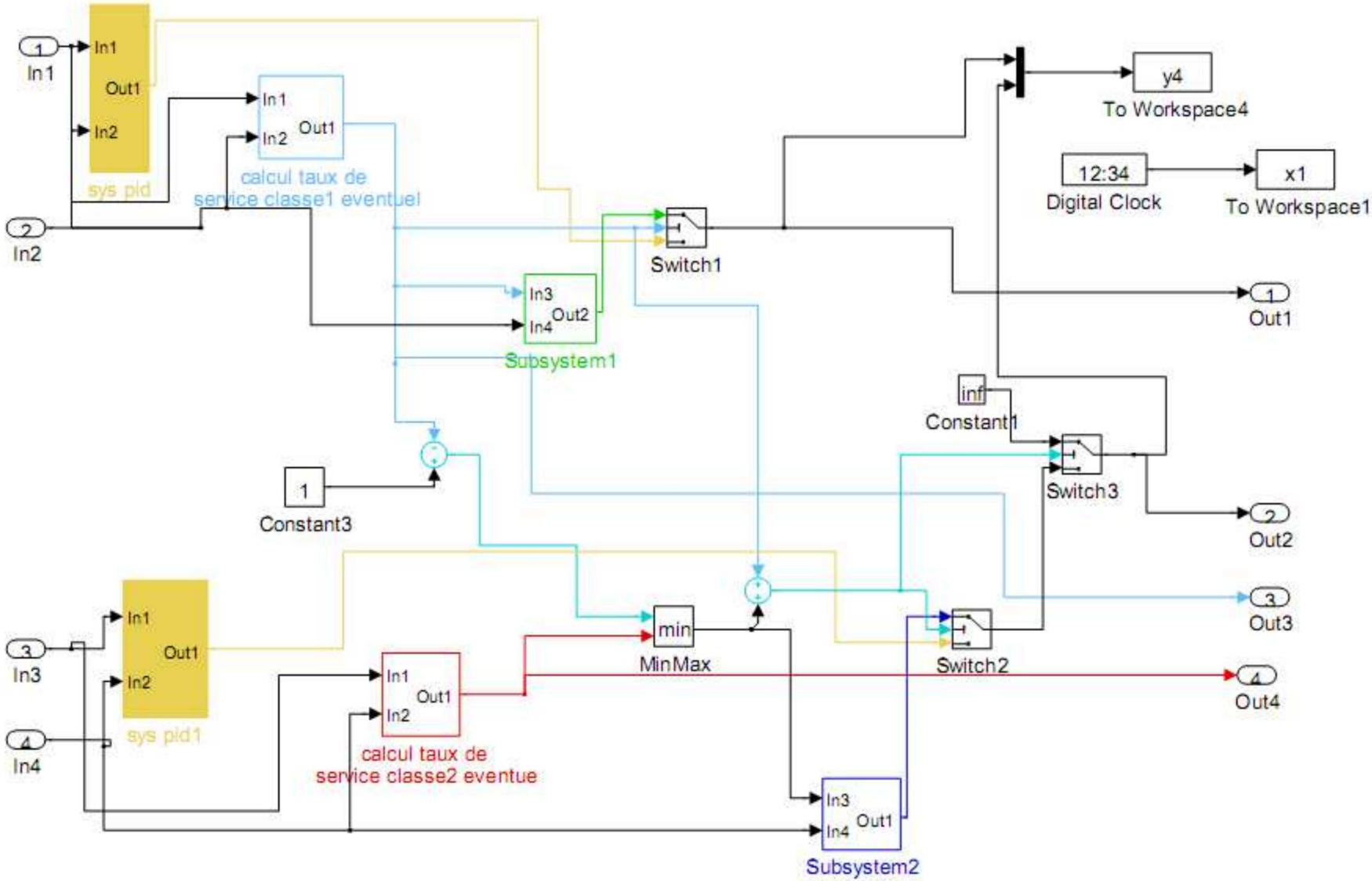
Modèle Relatif vis à vis Best effort



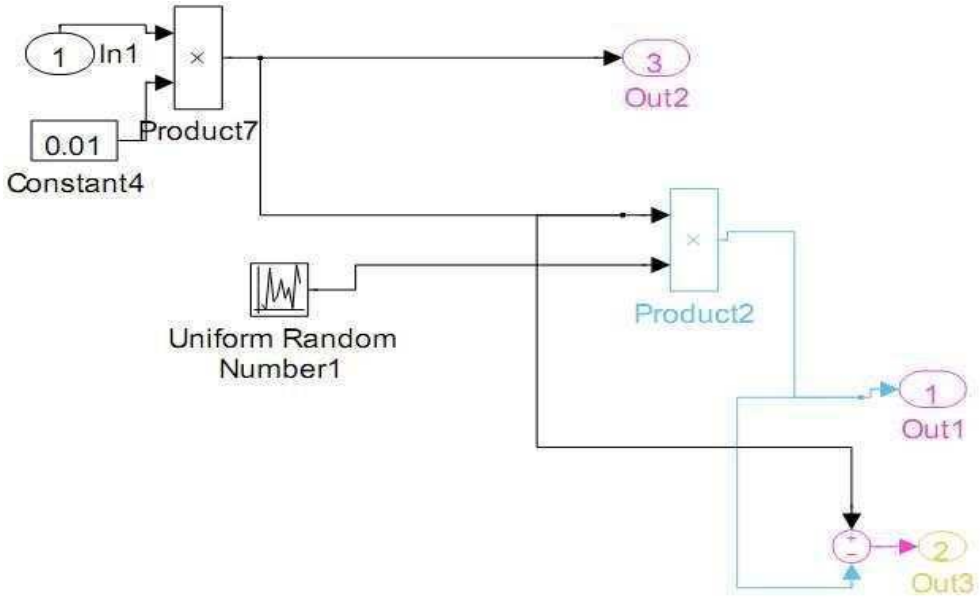
Modèle de système proposé



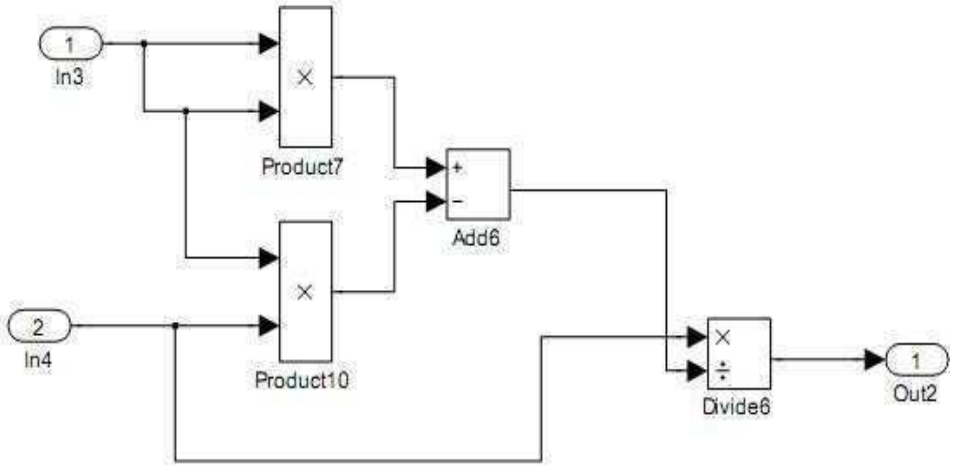
Sous-système de contrôle des délais absolus



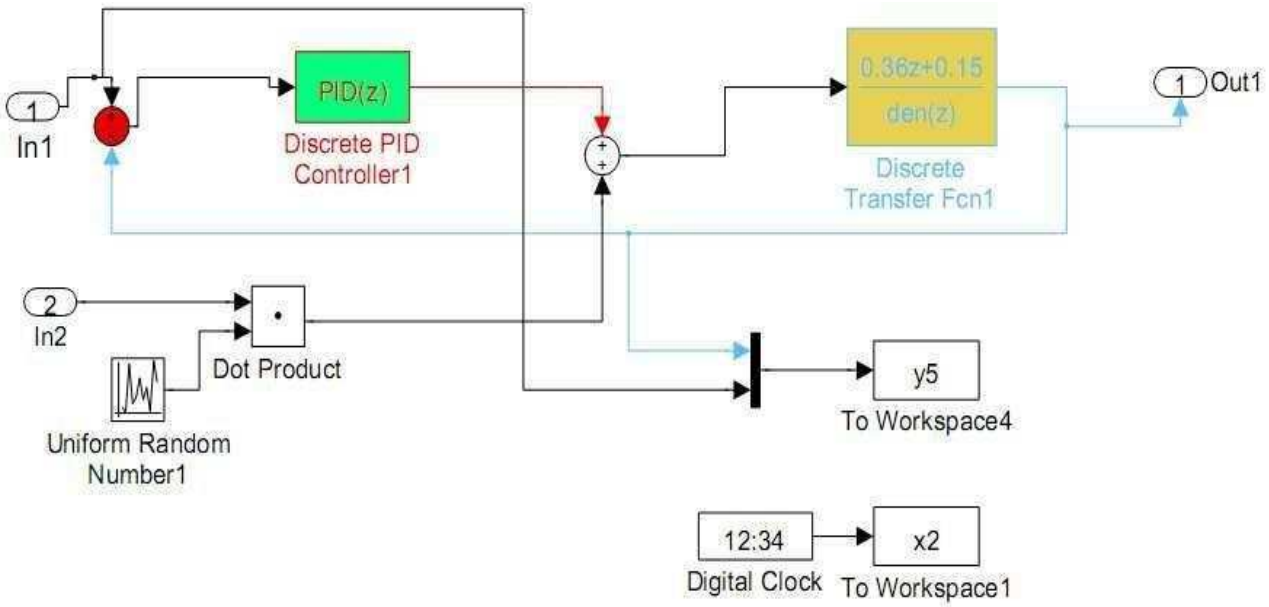
Sous-système générateur du taux d'arrivé



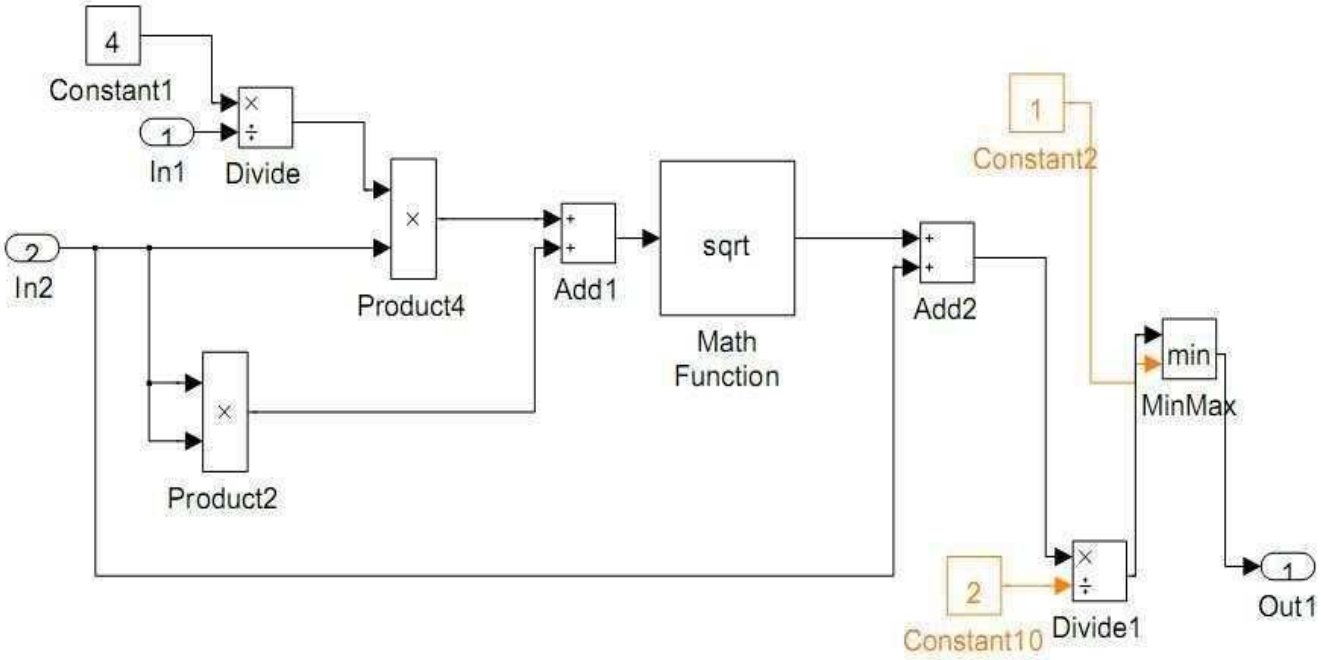
Sous-système de Calcul de temps de connexion



Sous-système de contrôle PID



Sous-système de Calcul de taux éventuel



APPENDICE B

MODELISATION DU CONTROLE ET LA TRANSFORME EN Z

A.1 Modélisation du contrôle

L'analyse du système de contrôle Feedback repose sur la dépendance entre les deux paramètres d'entée et de sortie. Le premier est le paramètre d'entrée contrôlé-le consigne- qui affecte le comportement du système. Le deuxième est le paramètre de sortie mesuré qui quantifie cet effet. Ces paramètres ne sont que des signaux se changeant à chaque instant, ils peuvent être soit à temps continus qui tiennent une valeur à chaque instant, ou à temps discrets lesquels tiennent uniquement des valeurs sur un certains instants.

La Figure A-1 montre les deux paramètres de contrôle d'entrée et de sortie ainsi que les perturbations survenues. Les états internes sont des variables qui peuvent se produire et caractérisent l'effet de la consigne sur la sortie mesurée. Notons qu'en général on s'intéresse au système de bout en bout (end to end).

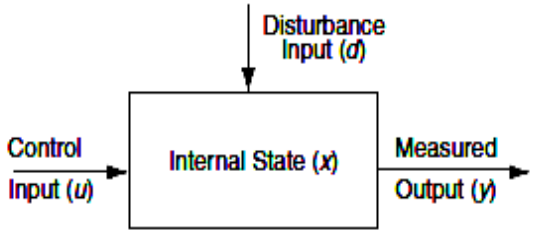


Fig. A-1 : les paramètres de contrôles [7]

Pour les systèmes informatiques, il est généralement plus commode de travailler avec les signaux à temps discrets au lieu des signaux continus. Il y a plusieurs raisons à préférer cette voie. Les outils de mesure utilisés dans les systèmes informatiques fournissent en général les valeurs à intervalles réguliers. Aussi, la fonction de mesure en continu est absolument pénible, qui nécessite souvent des instruments particuliers. Aussi, même si les réalisations en continu sont disponibles, elles sont coûteuses à obtenir. Enfin, les mesures de contrôle sont généralement

prises à des moments distincts et donc il vaut mieux travailler avec des signaux de sortie discrets, si le signal d'entrée est discret.

Les signaux discrets peuvent être construits à partir de signaux continus de plusieurs façons. L'approche simple est l'échantillonnage. Posons un signal continu X_C et un échantillon de temps fixe, noté T_S (time sample), donc le signal échantillonné est $X(k) = X_C (kT_S)$ ou X_C. Un problème ici est que pour des données très variables, l'échantillon de temps, doit être suffisamment court pour saisir le comportement dynamique du signal en temps continu par le signal en temps discret. D'autres méthodes comme la médiane statistique peuvent être utilisées.

A.2 Construction du modèle

Afin de construire le modèle, on doit identifier la dépendance des deux paramètres de la rétroaction, la sortie mesurée et l'entrée contrôlée. Cette corrélation peut être exprimée par une équation différentielle linéaire qui relie la valeur actuelle de sortie aux valeurs antérieures des deux paramètres. Cette équation est définie par la relation mathématique:

$$y(k) = a_1 y(k-1) + \cdots + a_n y(k-n) + b_1 u(k-1) + \cdots + b_m u(k-m) \quad \ldots\ldots E(A.1)$$

Dont n et m exprime respectivement le nombre des valeurs d'entrée et de sortie passées en tenant compte des coefficients a_i et b_i. En effet n et m décrivent l'ordre du modèle utilisé. Le modèle à premier ordre indique que la valeur prochaine à prévoir de la sortie à mesurer ne dépend que des valeurs actuelles des deux paramètres et de ses coefficients joignant:

$$y(k+1) = ay(k) + bu(k) \quad \ldots\ldots E(A.2)$$

Notons que la plupart des systèmes en réalité ne sont pas linéaires, par exemple dans le modèle à queue M/M/1/K, la capacité de la file d'attente a un effet non linéaire sur la charge ou le temps de réponse. Toutefois on peut formuler le modèle non linéaire par un autre modèle linéaire se basant sur des théories mathématiques, comme celle de la série de Taylor **[7]**. Plusieurs études ont été développées pour l'identification du modèle se basant sur les équations différentielles comme dans **[34]** et **[35]**. Malheureusement ces modèles nécessitent un considérable effort, et par fois il est impossible d'aboutir à la relation exacte entre ces paramètres. De ce fait il est suggéré de se relier aux modèles expérimentaux fondés sur la collecte des données du système actuel étudié. Quatre aspects ont été proposés au but de développer un tel modèle. Ces aspects sont décrits par l'organigramme dans la Figure A-2 ci-dessous :

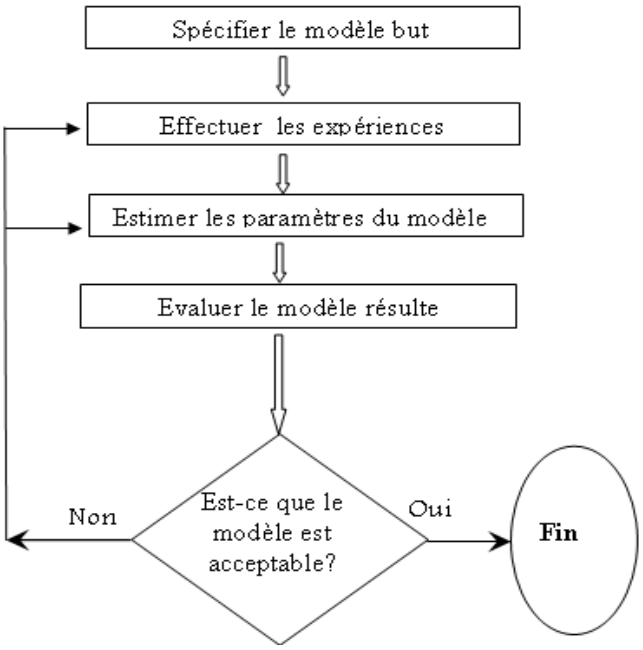


Fig. A-2 : Organigramme de développement d'un tel modèle [7]

1. la première étape est de spécifier le modèle à développer en déterminant les paramètres d'entrée et de sortie du système.

2. L'étape suivante est d'effectuer des essais expérimentaux suffisants pour estimer les paramètres de l'équation différentielle linéaire à l'ordre requis.

3. L'estimation des paramètres du modèle en se basant sur la collecte des données par les observations expérimentales représente une étape importante pour le développement du modèle. Les deux paramètres essentiels du modèle concernés sont l'entrée à contrôler et la sortie à mesurer. les valeurs à estimer ne sont que les coefficients *a* et *b* de ces paramètres dans l'équation différentielle.

   Ces observations expérimentales génèrent un ensemble de valeurs pour les deux paramètres (l'entrée à contrôler et la sortie à mesurer respectivement) et forment des tuplets $\{ \tilde{u}(k), \tilde{y}(k) \}$,$1 \leq k \leq N+1$.

   Posons que le point opérationnel à atteindre est $(\bar{u}, \bar{y})$, les égarements des paramètres recensés par l'observation sont :

$$\begin{cases} u(k) = \tilde{u}(k) - \bar{u} \\ y(k) = \tilde{y}(k) - \bar{y} \end{cases}$$

Or on prévoit la valeur de y par l'équation E(II.2): $\hat{y}(k+1) = ay(k) + bu(k)$

Donc l'erreur de la prédiction résiduelle est: $e(k+1) = y(k+1) - \hat{y}(k+1)$

On doit choisir les coefficients a et b dans le sens que la somme des moindres carrés soit optimum (minimisation des carrées des erreurs) [**19]:**

$$J(a,b) = \sum_{k=1}^{N} e^2(k+1) = \sum_{k=1}^{N} \left[y(k+1) - ay(k) - bu(k)\right]^2 \qquad \ldots\ldots\ldots \; E(A.3)$$

On peut trouver les valeurs des coefficients a et b en mettant les dérivés partiales par apport à ces variables à zéro, donc on obtient un système d'équation:

$$\begin{cases} \dfrac{\partial}{\partial a} J(a,b) = -2 \sum_{k=1}^{N} y(k) \left[y(k+1) - ay(k) - bu(k)\right] = 0 \\ \dfrac{\partial}{\partial b} J(a,b) = -2 \sum_{k=1}^{N} u(k) \left[y(k+1) - ay(k) - bu(k)\right] = 0 \end{cases}$$

Pour la résolution de ces deux dernières équations posons:

$$S_1 = \sum_{k=1}^{N} y^2(k)$$

$$S_2 = \sum_{k=1}^{N} u(k)y(k)$$

$$S_3 = \sum_{k=1}^{N} u^2(k)$$

$$S_4 = \sum_{k=1}^{N} y(k)y(k+1)$$

$$S_5 = \sum_{k=1}^{N} u(k)y(k+1)$$

Donc la solution de l'équation est:

$$a = \frac{S_3 S_4 - S_2 S_5}{S_1 S_3 - S_2^2} \qquad \ldots\ldots\ldots \; E(A.4)$$

$$b = \frac{S_1 S_5 - S_2 S_4}{S_1 S_3 - S_2^2} \qquad \ldots\ldots\ldots \; E(A.5)$$

4. L'évaluation du modèle résulte en utilisant les métriques statistiques de la précision comme le RMSE (Root Mean Square Error)

$$\text{RMSE} = \sqrt{\frac{1}{N} \sum_{k=1}^{N} \left[y(k+1) - \hat{y}(k+1)\right]^2} \qquad \ldots\ldots\ldots \; E(A.6)$$

Ou le modèle noté par R^2 : $R^2 = \frac{Var(y-\hat{y})}{Var(y)}$ ..... ***E(A.7)***

Le système est parfait si pour RMSE est proche de 0, et pour R^2 est proche de 1.

Cependant, si les données ne sont pas bien exprimées par le modèle résulte, alors en doit effectuer plus d'expériences et employer un ordre plus haut de l'équation différentielle.

A.3 La transformée en Z

La transformée en Z est l'une des techniques utilisées pour la résolution des problèmes d'ingénierie, fondée sur le remplacement des équations différentielles à des variables réelles, par certains développements dépendant de la variable complexe Z, pour fournir un moyen plus facile d'étudier, analyser puis extraire les propriétés du signal d'un système.

La transformée en Z est définie sur les fonctions de signal échantillonnées. Les valeurs de ces fonction sont déterminées à des instants réguliers 0, T, 2T…, kT par la suite $f(0)$, $f(1)$, $f(2)$…. c'est à dire $f(k)$ pour k=0, 1,2…

La transformée en Z est décrite par la formule suivante:

$$U(z) = u(0)z^0 + u(1)z^{-1} + u(2)z^{-2} + \cdots = \sum_{k=0}^{\infty} u(k)z^{-k} \quad \text{..... } \textit{E(A.8)}$$

Le k eme terme de cette série est le k eme élément de la suite multipliée par z^{-k}.

A.3.1 Exemples :

- La transformée de la fonction step $f(k)=1$ est :

$$U_{\text{step}}(z) = 1 + z^{-1} + z^{-2} + \cdots = \sum_{k=0}^{\infty} z^{-k} = \frac{1}{1-z^{-1}} = \frac{z}{z-1}$$

- La transformée de la fonction ramp $f(k)=k$ est :

$$\begin{aligned} U_{\text{ramp}}(z) &= 0 + 1z^{-1} + 2z^{-2} + 3z^{-3} + \cdots \\ &= \sum_{k=0}^{\infty} kz^{-k} \\ &= \frac{z}{(z-1)^2} \end{aligned}$$

Le tableau A-1 ci-dessous montre la transformée de quelque fonctions. Ci-après le tableau A-2 montre certaines propriétés de la transformée en Z [7]:

signal	***Domaine du temps (k≥0)***	***Transformé en Z***
impulsion	$u(0)=1$	$U(z)=1$
Étape	$u(k)=1$	$U(z)=\frac{z}{z-1}$
Rampe	$u(k)=k$	$U(z)=\frac{z}{(z-1)^2}$
Exponentiel	$u(k)=a^k$	$U(z)=\frac{z}{z-a}$
Sinus	$u(k)=\sin k\theta$	$U(z)=\frac{z\sin\theta}{z^2-(2\cos\theta)z+1}$
Cosinus	$u(k)=\cos k\theta$	$U(z)=\frac{z(z-\cos\theta)}{z^2-(2\cos\theta)z+1}$
Exp-Sin	$u(k)=a^k\sin k\theta$	$U(z)=\frac{za\sin\theta}{z^2-(2a\cos\theta)z+a^2}$
Exp-Cos	$u(k)=a^k\cos k\theta$	$U(z)=\frac{z(z-a\cos\theta)}{z^2-(2a\cos\theta)z+a^2}$

Tab. A-1 : La transformé de quelques fonctions usuelles

Propriété	***Domaine du temps (k≥0)***	***Transformé en Z***
Scalaire	$y(k)=au(k)$	$Y(z)=aU(z)$
Addition	$y(k)=u(k)+v(k)$	$Y(z)=U(z)+V(z)$
Retard unitaire	$y(k)=u(k-1)$	$Y(z)=z^{-1}U(z)$
Retard n-unités	$y(k)=u(k-n)$	$Y(z)=z^{-n}U(z)$
Changement unitaire	$y(k)=u(k+1)$	$Y(z)=zU(z)-zu(0)$
Changement n-unités	$y(k)=u(k+n)$	$Y(z)=z^nU(z)-z^nu(0)-\cdots-zu(n-1)$

Tab. A-2 : Propriétés de la transformation en Z

La transformée inverse Z^{-1} est la Transformée dont il attribue à chaque fonction *F(Z)* sa fonction originaire *f(k)* dans le domaine du temps échantillonnée.

A.4 La fonction de transfert

On définit La fonction de transfert, appelée aussi le gain de transfert qui décrit le système de contrôle et montre comment l'entrée U(Z) se transforme en sortie Y(Z), elle est définie par la relation: G(Z)=Y(Z) / U(Z).

A.4.1 Les pôles et les zéros de la transformé en Z

Soit la transformée en Z d'un signal échantillonné *F(Z)* **[20]**:

- ✓ Les valeurs de la variable *z* pour lesquelles la fonction *F(Z)* est nulle s'appellent zéros de la fonction *F(Z)*
- ✓ Les valeurs de la variable *z* pour lesquelles la fonction *F(Z)* est infinie s'appellent pôles de *F(Z)*

Autrement dit si une fonction *Y(Z)*:

$$Y(z)=\frac{N(z)}{D(z)}=\frac{\prod_{i=1}^{n}(z-z_i)}{\prod_{j=1}^{m}(z-p_j)}$$

zeros

poles

Donc les racines p_j du dénominateur *D(Z)* sont les pôles. Dans le *Z*-domaine *Y(Z)* peut s'écrire:

$$Y(z) = \frac{N(z)}{D(z)} = \frac{\prod_{i=1}^{n}(z - z_i)}{\prod_{j=1}^{m}(z - p_j)} = c_0 + \sum_{j=1}^{m}\frac{c_j}{z - p_j}$$

On obtient la fonction y(k) la transformée inverse $\mathbf{Z^{-1}}$ du *Y(Z)* dans le domaine de temps:

$$y(k) = c_0 + \sum_{j=1}^{m} c_j \times p_j^{k-1} \qquad \text{…….. } \textit{E(A.9)}$$

Les pôles sont essentiels pour déterminer les caractéristiques du système spécialement la stabilité et le temps de transition:

- y(k) converge si tous les pôles $P_j < 1$
- le pôle négatif fournit des oscillations
- les pôles à des petites magnitudes ayant des temps moins court pour converger
- le pôle à large magnitude c'est le pôle dominant.

La Figure A-3 représente des pôles avec des valeurs différentes, tandis que la Figure A-4 montre l'influence de la magnitude des pôles.

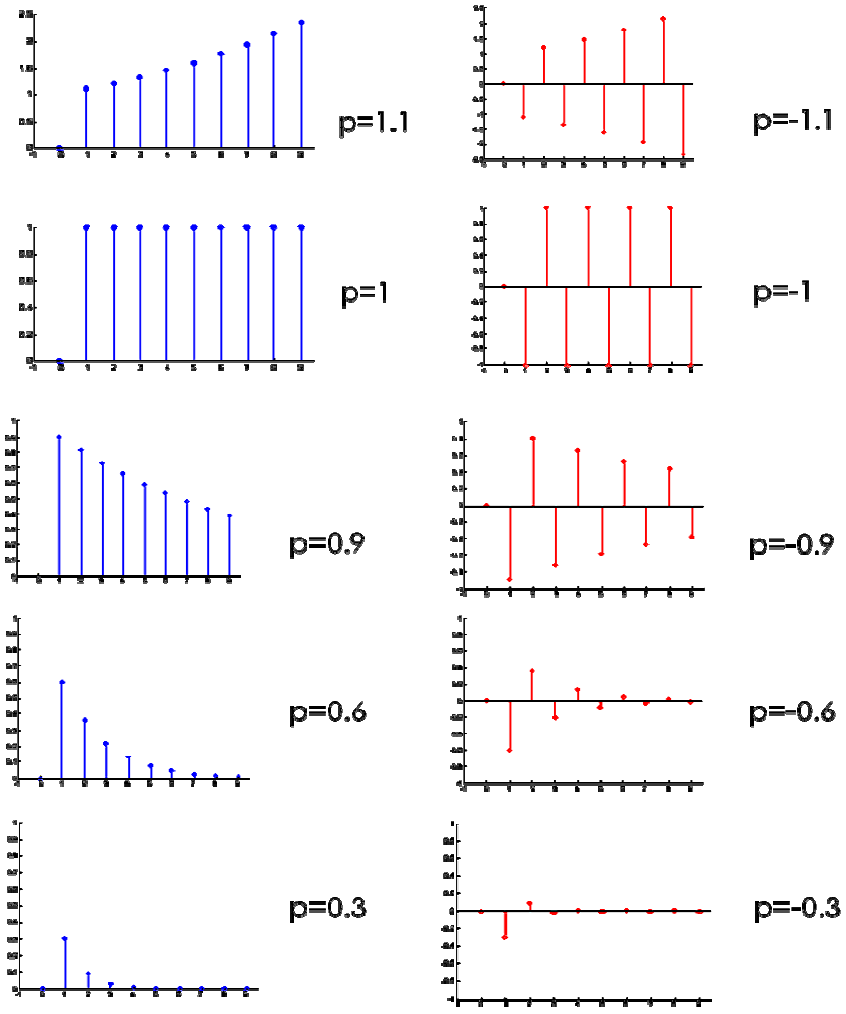


Fig. A-3 : Représentation des Différente valeurs des pôles

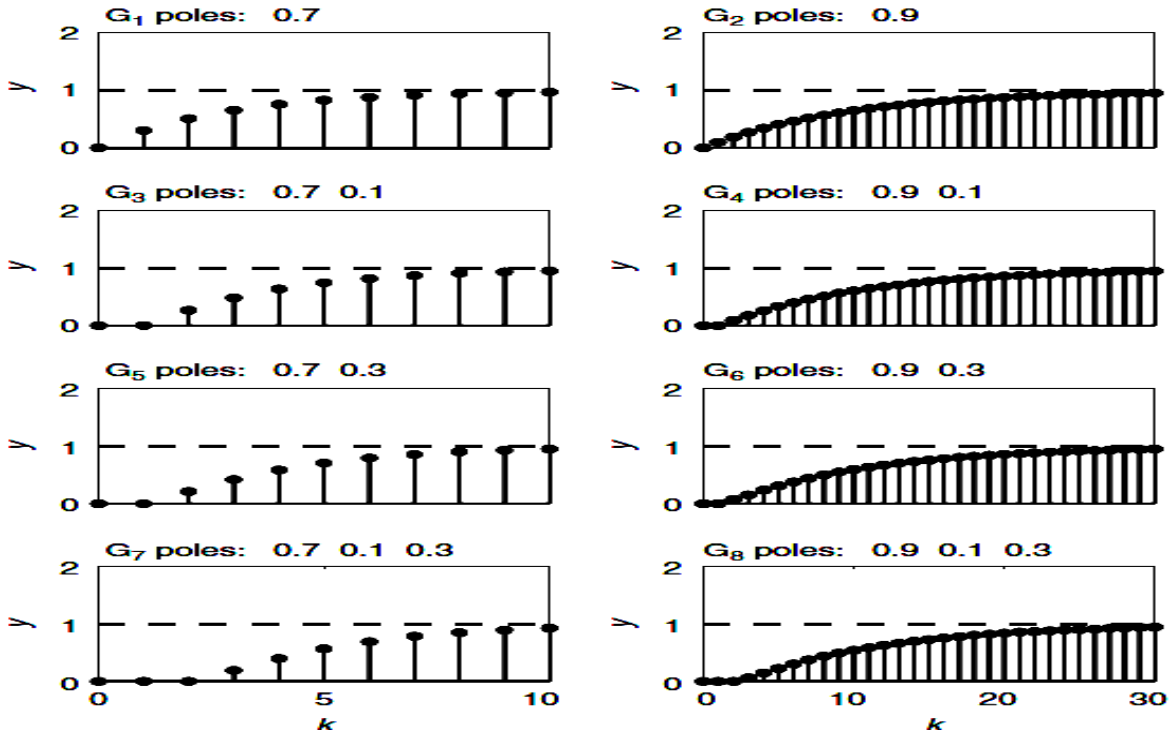


Fig. A-4 : Influence des magnitudes des pôles

A.4.2 Théorème des valeurs finies

Si tous les pôles de *(z-1) F(Z)* sont dans le cercle unitaire (moins de 1) alors:

$$\lim_{k\to\infty} f(k) = \lim_{z\to 1}(z-1)F(z) \qquad \text{…….. } E(A.10)$$

Quelle est la valeur du (Steady-State) pour laquelle la sortie vient de converger?

$$y(k) = a_1 y(k-1) + \ldots + a_n y(k-n) + b_1 u(k-1) + \ldots + b_m u(k-m)$$

La transformée $$G(z) = \frac{Y(z)}{U(z)} = \frac{b_1 z^{-1} + \ldots + b_m z^{-m}}{1 - a_1 z^{-1} - \ldots - a_n z^{-n}} \qquad \text{…….. } E(A.11)$$

Avec z=1: $$G(1) = \frac{b_1 + \ldots + b_m}{1 - a_1 - \ldots - a_n}_{ss}$$