N° d'ordre : 11/2011-M/IN

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE DES SCIENCES ET DE LA TECHNOLOGIE

HOUARI BOUMEDIENE

USTHB / ALGER

FACULTE D'ELECTRONIQUE ET D'INFORMATIQUE

MEMOIRE

Présenté pour l'obtention du diplôme de MAGISTER

en : INFORMATIQUE

Spécialité : Informatique Mobile

Par : CHARABI Leila

SUJET :

Proposition d'une Approche Numérique pour l'Évaluation des Performances des Systèmes avec Rappel et Serveurs Hétérogènes

Soutenu publiquement le 06/07/2011, devant le jury composé de :

Mr.	A. AISSANI	Prof	USTHB / FEI	Président
Mme.	N. GHARBI	MAB	USTHB / FEI	Directrice de mémoire
Mme.	M. BOUKALA	Prof	USTHB / FEI	Examinatrice
Mr.	M. BENCHAIBA	MCA	USTHB / FEI	Examinateur

Dédié à

Mes très chers parents, mes grands parents, mes soeurs KARIMA, AHLAM et SAFIA et surtout mon mari ISMAIL ... qui m'ont toujours soutenue et encouragée... À mon cher frère DJAAFAR et la petite HIND. À toi particulièrement mon adorable neveu ABD ENACER, tu es le bienvenu dans notre monde !

Remerciements

Je remercie DIEU le Tout Puissant pour Son aide.

Je tiens à exprimer ma profonde gratitude et mes remerciements les plus sincères à Madame le docteur **Nawel Gharbi Bouhal**, *ma directrice de mémoire, pour m'avoir accueillie et accepté de diriger mes recherches dans des moments difficiles ... et pour la confiance qu'elle m'a toujours témoignée. Sa compétence, ses conseils, ses encouragements, sa patience, sa compréhension et son amitié m'ont permis de bien mener ce travail.*

Je souhaite aussi exprimer ma gratitude à Mr le professeur Aissani *pour l'honneur qu'il m'a fait en acceptant de présider le jury de mon mémoire. Mes remerciements vont aussi au professeur* Ioualalen *et au docteur* M. Benchaiba *pour avoir accepté de juger mon travail.*

Mes sincères remerciements vont également à tous mes enseignants, sans exception, qui ont veillé à ma formation.

Merci à tous les membres de ma famille qui m'ont accompagnée tout au long de mes études, voir de ma vie, par leur amour inconditionnel et leur soutien absolu.

Je n'oublie pas d'adresser mes vifs remerciements à toutes mes amies, et à tous ceux qui, de près ou de loin, m'ont aidé à l'aboutissement de ce modeste travail.

Table des matières

Table des figures

Liste des tableaux

Résumé

Les systèmes avec rappel sont des systèmes dans lesquels les clients qui trouvent tous les serveurs occupés ou non disponibles, rappellent ultérieurement pour le service, à des intervalles de temps aléatoires. Cependant, la prise en considération du phénomène d'appels répétés a introduit de grandes difficultés analytiques. En fait, des résultats explicites détaillés existent pour certaines files d'attente avec rappel particulières, avec des hypothèses contraignantes sur certains paramètres, tel que la taille de la source de clients, le nombre de serveurs, l'homogénéité des serveurs, etc.

Il est important de préciser que les études des modèles avec rappel et serveurs hétérogènes restent à ce jour très rares et les résultats obtenus sont assez limités. À cet effet, le but du sujet est la proposition d'une approche numérique permettant l'analyse des performances des systèmes avec rappel à source finie de clients et deux classes de serveurs hétérogènes, en utilisant le modèle des réseaux de Petri stochastiques généralisés. Ce formalisme de haut niveau permet une description simple du comportement des systèmes complexes avec phénomène de rappel. Par ailleurs, il offre un bon moyen de génération automatique de la chaîne de Markov correspondante, pour l'analyse des performances. Cependant, la génération et la résolution de la chaîne de Markov déduite, nécessite souvent un espace mémoire très large et un temps d'exécution très long, puisque la taille de l'espace d'états croît exponentiellement en fonction de la taille de la source de clients et du nombre de serveurs. Ainsi, nous nous intéressons dans ce projet à la conception et l'implémentation d'une approche numérique permettant l'obtention automatique du générateur infinitésimal et le calcul des indices de performances à l'état stationnaire sans avoir à générer ni le graphe d'accessibilité ni la chaîne de Markov, et ceci, en considérant deux disciplines de service : la discipline du service aléatoire, et celle du service le plus rapide.

Mots clés : Systèmes avec rappel, Source finie de clients, Serveurs hétérogènes, Réseaux de Petri stochastiques généralisés, Générateur infinitésimal, Indices de performance.

Introduction Générale

Dans la théorie des files d'attente classiques [59,95], il est souvent supposé qu'un client qui ne peut pas obtenir le service immédiatement dès son arrivée, soit quitte définitivement le système, ou bien rejoint un espace d'attente pour être servi suivant une certaine discipline. Parfois, des clients impatients peuvent quitter la file d'attente, mais il est également supposé qu'ils quittent le système définitivement. Toutefois, l'hypothèse de la perte de clients qui ont choisi de quitter le système n'est qu'une approximation à une situation réelle. Habituellement, après un temps aléatoire, un tel client revient au système et tente de se faire servir de nouveau. Dans un réseau téléphonique par exemple, tout le monde sait qu'un abonné qui trouve le réseau saturé, répète l'appel jusqu'à ce que la connexion nécessaire soit établie. En conséquence, le flux des appels circulant dans un réseau téléphonique se compose de deux parties ; le flux des appels primaires, qui reflète le nombre réel des abonnés qui veulent accéder au service, et le flux des appels répétés, qui est la conséquence du manque de ressources. Ce comportement des utilisateurs est connu dans la littérature par le *phénomène de rappel* [38,7]. Ce phénomène a suscité l'intérêt de plusieurs chercheurs [23, 94, 24] et a étendu le modèle classique de file d'attente à celui des files d'attente avec rappel (FAR), et ceci dans le but d'expliquer et d'étudier le comportement stochastique des clients des systèmes téléphoniques. L'application de ce modèle a été ensuite étendue à la modélisation des réseaux LAN (Local Area Network), des réseaux informatiques, des réseaux de télécommunication, des réseaux mobiles cellulaires, des systèmes aéronautiques, ainsi que différents protocoles de communication, notamment les protocoles CSMA/CD (Carrier Sense Multiple Access with Collision Detection), les disciplines Auto-repeat, Repeat-last-number, etc. Le progrès réalisé dans ce domaine est résumé dans les articles de synthèse de Yang

et Templeton [97], Falin [36], Kulkarni [64], Artalejo [6], la monographie faite par Falin et Templeton [38], les collections de la bibliographie des travaux publiés dans ce domaine entre 1990 et 1999 [9, 10], et entre 2000 et 2009 [11] faites par Artalejo, ainsi que le livre de Artalejo et Gómez-Corral [7].

Cependant, d'un point de vue technique, la considération du deuxième flux qui est le flux des appels répétés dans l'étude des systèmes avec rappel, a engendré beaucoup de difficultés analytiques. Ce qui rend les méthodes classiques et les résultats obtenus dans la théorie des files d'attente classiques inadéquats. Ceci explique le recours des chercheurs à des techniques d'approximation et de simulation et aux algorithmes numériques pour l'élaboration des indices de performances dans le domaine des FAR. Cependant, la plupart de ces résultats restent rares et assez limités, en effet, des résultats analytiques détaillés n'existent que pour un certain nombre de files d'attente particulières, avec des hypothèses contraignantes sur certains paramètres comme le nombre et l'homogénéité des serveurs, l'homogénéité des clients, la taille de la population, et la distribution des temps d'arrivées et de rappel des clients qui est souvent supposée exponentielle.

De plus, la majorité des études sur les FAR supposent que la station de service consiste en un ou plusieurs serveurs identiques. Toutefois, dans beaucoup d'applications pratiques dans le domaine des télécommunications et des réseaux mobiles cellulaires par exemple, les serveurs utilisés peuvent avoir des caractéristiques différentes (comme le temps moyen de service), c'est ce qui définit les *FAR avec serveurs hétérogènes*. Il est clair que les modèles hétérogènes sont beaucoup plus difficiles pour l'analyse mathématique que les modèles homogènes. Ainsi, des résultats explicites ne sont disponibles que dans quelques cas particuliers.

Par ailleurs, il est souvent moins difficile dans l'étude de FAR, de considérer une source infinie de clients et un flux poissonnien des arrivées primaires. Or, dans de nombreux systèmes réels, le nombre d'utilisateurs qui accèdent au système est fini. Il est très important de prendre en compte le fait que le taux de génération de nouveaux appels primaires décroît quand le nombre de clients dans le système croît. Cela définit ce qu'on appelle les FAR *à entrée quasi-aléatoire*, ou bien les FAR *à source finie* de clients.

Les systèmes avec rappel à entrée quasi-aléatoire et serveurs hétérogènes restent encore un domaine intéressant à explorer. Nous n'avons trouvé dans la littérature que le peu d'articles d'Efrosinin [33], et Sztrik [87] où le cas des serveurs hétérogènes a été considéré en utilisant le modèle des FAR, et l'article de Gharbi [43] qui a proposé la modélisation et l'analyse des FAR multi-classes, à l'aide des réseaux de Petri stochastiques généralisés colorés (RdPSGC).

Par ailleurs, les *réseaux de Petri* (*RdP*), sont un outil graphique et mathématique de haut niveau qui permet de décrire et de modéliser le comportement dynamique des systèmes à évènements discrets, tels que les systèmes de production, les systèmes automatisés, les systèmes informatiques et les protocoles de communication, etc. Du côté graphique, on peut voir d'une manière naturelle la synchronisation, le parallélisme, les conflits, le partage de ressources, etc. Du côté mathématique, les équations d'états que l'on peut établir permettent d'évaluer les propriétés quantitatives du modèle. Dans le but d'augmenter la puissance de spécification de cet outil, en prenant en compte la notion de temps, plusieurs extensions ont été apparues, comme les réseaux de Petri temporisés, les réseaux de Petri temporels, les réseaux de Petri stochastiques, ainsi que les réseaux de Petri stochastiques généralisés (RdPSG).

Ces derniers sont caractérisés par la présence de deux types de transitions ; les transitions immédiates, et les transitions temporisées. Comme leur nom l'indique, les transitions immédiates modélisent les actions instantanées telles que la synchronisation, les opérations logiques et les évènements d'urgence qui ont un temps de franchissement nul, alors que les transitions temporisées, dont le délai de franchissement suit généralement une loi de probabilité aléatoire, décrivent les opérations nécessitant un temps aléatoire pour s'exécuter, comme la prise en charge d'une requête de client par exemple. L'analyse par RdPSG consiste à générer, à partir du graphe des marquages accessibles du RdPSG en question, une chaîne de Markov à temps continue CMTC réduite. Cependant, la génération du graphe d'accessibilité ainsi que la chaîne de Markov réduite, et la résolution de cette dernière, requièrent un espace de stockage important et un temps de calcul très long, étant donné que l'espace d'états augmente de façon exponentielle en fonction de la taille de la source des clients et du nombre de serveurs. Donc, pour un système avec rappel réel, le

modèle correspondant pourrait avoir un espace d'états énorme.

Ainsi, en utilisant le modèle RdPSG, nous proposons dans cette thèse, une approche algorithmique pour le calcul direct du générateur infinitésimal sans avoir à générer ni le graphe d'accessibilité ni la chaîne de Markov sous-jacente. En outre, nous développons les formules des principaux indices de performance, en fonction du nombre de serveurs de chaque catégorie de serveurs, la taille de la source de clients et les probabilités stationnaires, mais surtout indépendamment de l'ensemble des marquages accessibles.

Organisation du Document

De façon plus concise, ce mémoire est organisé en trois chapitres, comme suit :

- Le premier chapitre est consacré à la théorie des files d'attente avec rappel. Un rappel sur les modèles de files d'attente classiques sera d'abord présenté afin de mieux aborder le domaine de files d'attente avec rappel. Dans le cadre d'un état de l'art, nous allons présenter quelques variantes de FAR, en exposant les différentes méthodes et les principaux résultats obtenus, ainsi que les difficultés engendrés par la considération du flux de rappel. Nous allons nous focaliser sur les FAR multiserveurs, les FAR à source finie de clients et les FAR avec serveurs hétérogènes.
- Le second chapitre a pour objet les réseaux de Petri stochastiques généralisés. Nous commencerons par donner quelques rappels sur les probabilités, en particulier les processus stochastiques, ainsi que sur les chaînes de Markov à temps discret et à temps continu. Nous attaquerons dans la deuxième partie, les réseaux de Petri ordinaires ; leur définition, propriétés, évolution dynamique, ainsi que les différentes méthodes d'analyse des RdP. Enfin, la troisième partie traitera les réseaux de Petri stochastiques généralisés RdPSG, en se focalisant surtout sur leur évolution dynamique, et leur résolution numérique.
- Et dans le troisième chapitre, nous présenterons notre approche algorithmique pour la modélisation et l'évaluation des performances des systèmes avec rappel, source finie de clients et à deux classes de serveurs hétérogènes, en se basant

sur les RdPSG. Tout en considérant deux politiques de service, à savoir la politique du service *aléatoire*, et la politique du service *le plus rapide*, nous commencerons par donner le modèle RdPSG équivalent à chaque politique, ensuite nous développerons des algorithmes permettant le calcul automatique du générateur infinitésimal, en se passant du graphe des marquages accessibles, et de la chaîne de Markov réduite correspondante, avant de procéder au développement des formules de calcul des indices de performances. Sachant que nous avons implémenté cette approche, en développant un outil en *C#*, nous allons valider notre modèle dans le cas homogène, en comparant les résultats obtenus par notre approche avec ceux obtenus par le programme de Falin et Templeton dans [38]. Nous allons aussi effectuer quelques expérimentations pour mettre en évidence l'effet de quelques paramètres, comme le phénomène de rappel et le nombre de serveurs sur le temps de réponse moyen, tout en comparant les deux disciplines de service utilisées.

– Nous terminerons notre mémoire par une conclusion générale et les perspectives ouvertes à ce travail.

Chapitre 1

Files d'attente avec rappel

1.1 Introduction

« Tout résultat théorique qui ne prend pas en considération l'effet de la répétition (rappel), doit être suspect », KOSTEN [63] s'est exprimé pour montrer l'importance de la théorie des files d'attente avec rappel dans la modélisation de tout système à structure de file d'attente.

La notion de *files d'attente avec rappel* , ou bien avec *appels répétés* (*retrial queues* ou *queueing systems with repeated calls*, en anglais), a vu le jour à la fin des années quarante du siècle précédent, quand elles ont été présentées par plusieurs chercheurs comme CLOS [23], WILKINSON [94] et COHEN [24] comme une alternative dans la modélisation des systèmes de commutation téléphoniques. À cette époque, *les files d'attente classiques*, étaient le meilleur moyen utilisé dans ce domaine. D'une façon abrégée, les files d'attente standards, offrent deux scénarios pour régler le problème d'un client qui arrive et trouve tous les serveurs occupés :

- Soit, il quitte définitivement le système (le principe de systèmes d'Erlang avec perte) [18],
- Soit, dans les files d'attente avec buffer, il rejoint la file d'attente et attend la libération du serveur pour lui [59, 95].

Cependant, le comportement réel des clients peut être différent ; en recevant le signal de ligne occupée, les abonnés téléphoniques généralement répètent leurs demandes jusqu'à ce que la connexion désirée soit établie. D'un point de vue technique, le

flux d'appels circulant dans un réseau téléphonique est constitué de deux flux ; le flux des appels *primaires*, qui reflètent le nombre réel des clients désirant effectuer une communication téléphonique, et le flux des appels *répétés* (*rappels*), qui sont causés par l'échec des tentatives d'appels précédentes. Le modèle des files d'attente standards ne prend pas en considération ce phénomène, et par conséquent ne peut pas être appliqué pour résoudre un grand nombre de problèmes [38].

Ainsi, le modèle des files d'attente avec rappel (FAR) a été introduit. Ce modèle est caractérisé par le fait qu'un client qui arrive alors qu'aucun serveur n'est disponible pour le prendre en charge, quitte le système mais revient après un temps aléatoire et répète sa demande. Durant ce temps aléatoire, le client est dit *en orbite*.

Dans [24], COHEN a étudié le problème principal de la théorie du trafic téléphonique, il a trouvé que les éléments suivants étaient essentiels pour son étude : le nombre de clients qui utilisent le service téléphonique, la distribution de temps d'inter-arrivées, la distribution de la durée des appels, et le comportement des clients qui trouvent toutes les lignes occupées. Le problème principal était de trouver le nombre moyen de lignes occupées, la probabilité que toutes les lignes soient occupées, et le nombre des appels perdus.

En plus des réseaux téléphoniques classiques [52], notamment avec les facilités : *répéter-dernier-numéro* (*repeat-last-number*) et *auto-recomposition* (*auto-repeat*), l'effet de rappel dans les réseaux sans fil est généralement un problème plus délicat [91,78]. On trouve aussi cet effet dans le transfert de données via Internet [17]. Il est à noter aussi que le phénomène de rappel peut aussi être observé dans beaucoup de systèmes, comme les protocoles de communication, ainsi que dans beaucoup de files d'attente dans notre vie quotidienne quand un client part et revient plus tard dans l'espoir de tomber sur un guichet libre plus rapidement, dans l'évaluation des performances des protocoles de la couche MAC, comme CSMA/CD [22], ainsi que dans la modélisation des centres d'appels [68], etc.

Ainsi, vu son importance dans la modélisation de toutes ces applications et beaucoup d'autres, la théorie des FAR a continué à susciter l'attention de beaucoup de chercheurs. Aujourd'hui, on trouve des milliers de publications dans la littérature sur les FAR. Le livre de FALIN et TEMPLETON [38], ainsi que le livre d'ARTALEJO

et Gomez-Corral paru en 2008 [7] font partie des sources les plus intéressantes. Pour une bibliographie exhaustive et classifiée des travaux publiés dans ce domaine, le lecteur est référé à [9, 10, 11].

Nous nous proposons dans ce chapitre de présenter un état de l'art sur les files d'attente avec rappel. Afin de bien aborder cela, nous commençons d'abord par définir le fonctionnement général des files d'attente standards, avant celui des files d'attente avec rappel, puis nous introduisons la notation de Kendall. La théorie des files d'attente avec rappel étant devenue un domaine très vaste qui englobe beaucoup de variantes selon plusieurs critères, comme la taille et les caractéristiques des clients ainsi que les caractéristiques et le nombre des serveurs impliqués dans chaque modèle, nous allons nous satisfaire d'étudier quelques variantes de FAR qui ont un lien avec notre modèle, en exposant leur modèle mathématique ainsi que les principaux résultats qui existent dans la littérature. Les modèles qui seront présentés sont : les FAR multiserveurs, les FAR alimentées par une source finie de clients (homogène et hétérogène), et les FAR avec serveurs hétérogènes (fiables et non-fiables). À la fin, nous terminerons par une conclusion.

1.2 Le modèle général de files d'attente avec rappel

1.2.1 Rappel sur la structure des files d'attente classiques

Les files d'attente constituent une théorie très riche, dérivée du domaine des mathématiques appliquées et de la recherche opérationnelle, elle s'intéresse à la modélisation et l'évaluation des performances des systèmes à files d'attente, et fait appel à des méthodes statistiques et algébriques. Le mathématicien et ingénieur danois Erlang qui travaillait pour la *Copenhagen Telephone Exchange*, a publié le premier papier sur les files d'attente, il y a un siècle de cela [34]. Depuis, les files d'attente ont trouvé un domaine d'applications très large (réseaux informatiques, commerce, business, industrie, etc.) [59].

De manière générale, une file d'attente est constituée de :

- Un ou plusieurs *serveurs* (ou guichets) à qui on demande un service,
- Un ensemble de *clients* qui demandent ce service,

– et d'un lieu où les clients attendent quand aucun serveur n'est disponible, ce lieu est appelé *buffer*.

L'aléatoire réside dans la variabilité des temps de service et l'imprévisibilité des arrivées de clients. Les instants d'arrivées constituent un *processus stochastique* ; plus précisément, on considère que les intervalles de temps qui séparent les instants d'arrivée de deux clients consécutifs sont des *variables aléatoires indépendantes et identiquement distribuées* (v.a.i.i.d). De même, les durées de service allouées aux clients sont des v.a.i.i.d. L'évolution du système est définie par le choix d'une discipline de service qui précise dans quel ordre sont servis les clients (FIFO, LIFO, RANDOM, etc.).

Schématiquement, on représente une file d'attente de la manière suivante (figure 1.1) :

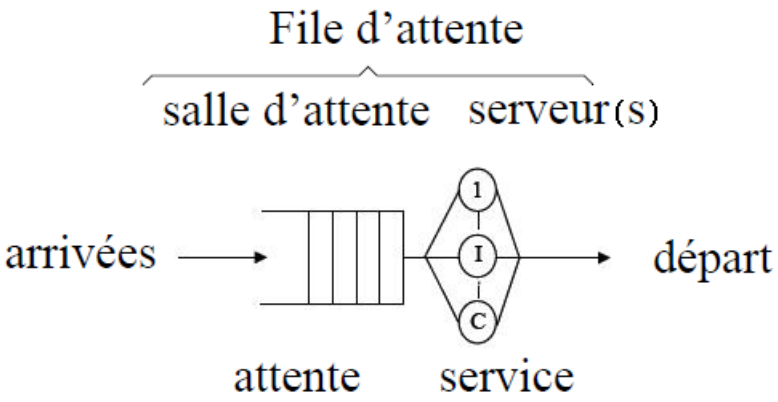

FIG. 1.1 – Représentation graphique d'un système d'attente classique.

Cependant, un système de files d'attente tel qu'il est défini ne convient pas pour modéliser un grand nombre de problèmes qui sont caractérisés par le *phénomène de rappel.* Pour expliquer ce phénomène, prenons l'exemple de systèmes téléphoniques. En effet, les clients désirant effectuer un appel, mais qui trouvent toutes les lignes occupés ne peuvent pas s'insérer dans un buffer pour attendre qu'une ligne soit libre, mais rappellent chacun indépendamment des autres à des instants aléatoires jusqu'à ce qu'ils soient satisfaits. Les *files d'attente avec rappel* (FAR) ont été introduites pour palier à ce problème [23, 94, 24].

1.2.2 Description du modèle des files d'attente avec rappel (FAR)

Les systèmes de files d'attente avec rappel, se distinguent des files d'attente standards par le fait qu'un client qui trouve tous les serveurs occupés ne rejoint pas le buffer, mais rappelle pour le service après un temps aléatoire. Ainsi, un système de files d'attente avec rappel (FAR) se compose d'une station de service avec s serveurs ($s \geq 1$) indépendants et montés en parallèle, d'une source de clients finie ou infinie, et d'un espace d'attente imaginaire qu'on appelle *orbite* ou *pool*, l'orbite peut être à son tour finie ou infinie.

Le schéma général d'une file d'attente avec rappel est donné dans la figure 1.2

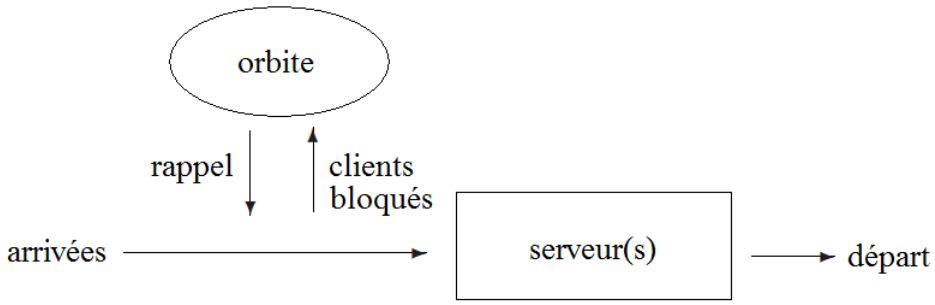

FIG. 1.2 – Représentation graphique d'un système d'attente avec rappel

Le client qui arrive à une station de service peut trouver un ou plusieurs serveurs libres et être pris en charge par un des serveurs immédiatement, il quittera le système dès la fin de son service. Sinon, si tous les serveurs sont occupés, le client quitte le système mais revient après un temps aléatoire pour rappeler pour le service, durant ce temps là, le client est dit *en orbite*. chaque client dans l'orbite forme un processus d'*arrivées secondaires* et est traité comme un client primaire. Le processus d'appels secondaires est en fait un processus aléatoire où l'intervalle de temps entre deux appels consécutifs s'appelle *temps de rappel*, ce temps est indépendant de tous les temps de rappel précédents.

Notons que le processus décrit par un tel modèle est un processus stochastique ; en effet, ni les moments d'arrivées des clients, ni les temps de rappel, ni les durées de services ne sont connus à l'avance.

1.2.3 La notation de Kendall

Afin de mieux se retrouver dans les nombreuses variantes possibles des FAR, la notation suivante, connue sous le nom *Notation de Kendall* [97] est utilisée. Cette notation permet également de décrire les files d'attente standards. Elle est de la forme suivante :

$$A/B/c/L/K$$

où :

- A désigne la loi des *inter-arrivées*,
- B désigne la loi de service. Selon la nature de la loi des inter-arrivées et la loi de service, A et B peuvent prendre les valeurs :

  M la distribution exponentielle (M pour Markov) ; dans ce cas, le processus d'arrivée est un processus de Poisson,

  E_n la distribution d'Erlang avec n phases,

  D pour la distribution déterministe ; dans ce cas, le processus d'arrivée est périodique,

  G pour une distribution générale.

  Cette liste n'est pas exhaustive, il existe bien d'autres lois.
- c, est le nombre de serveurs. c peut prendre n'importe quelle valeur entière, supérieure ou égale à 1.
- L est la capacité du système, c'est-à-dire le nombre maximum de clients qui peuvent être présents simultanément, en comptant le ou les clients en service. L peut prendre n'importe quelle valeur supérieure à c, y compris l'infini. Par défaut, c'est l'infini.
- K représente la taille de la source des clients. Par défaut c'est l'infini.

L et K admettent des valeurs par défaut, ainsi, ils peuvent être omis de la notation.

La distribution des temps de rappel est omise de la notation, car elle est supposée généralement exponentielle de taux ν. Cependant, on doit noter que plusieurs publications récentes considèrent une loi de rappel non-exponentielle [45, 56, 55], étant

donné que la distribution exponentielle pour les temps de rappel n'est pas toujours l'idéale pour modéliser les systèmes de télécommunication, par exemple [96].

Cette notation peut aussi être étendue pour exprimer l'hétérogénéité des clients ou bien des serveurs, dans ce cas, une flèche ($\rightarrow$) est marquée sur le symbole de la loi des inter-arrivées, et/ou celle des services respectivement. Par exemple, $M/\vec{M}/c$ désigne une FAR à c serveurs hétérogènes, avec temps de service markovien, source infinie de clients, et des arrivées markoviennes.

Nous allons maintenant introduire les caractéristiques les plus fréquemment utilisées dans l'analyse des files d'attente. En effet, lors de l'étude d'un système de files d'attente avec rappel (FAR), on tient compte des caractéristiques suivantes :

- La nature stochastique du processus d'arrivées des clients dans le système qui est définie par la distribution des intervalles de temps séparant deux arrivées successives ;
- La taille de la source de clients qui peut être finie ou infinie ; une source finie donne lieu à un *système fermé*, tandis qu'une source infinie correspond à un *système ouvert* ;
- La capacité de l'orbite ;
- La loi de probabilité du processus de rappel ;
- La distribution du temps aléatoire de service ;
- Le nombre et l'homogénéité des serveurs ; quand la station est formée d'un seul serveur, le système est dit *mono-serveur* ou à *serveur unique*, mais si elle comporte plusieurs serveurs, on parle de système *multi-serveurs*. Dans les modèles multi-serveurs, les serveurs peuvent avoir des caractéristiques statistiques distinguées (fiabilité, rapidité, etc.), dans ce cas là, le système est dit à *serveurs hétérogènes*.

Ainsi, en combinant ces caractéristiques, des extensions très variantes des FAR ont été distinguées dans la littérature. On trouve par exemple le modèle de FAR multiserveurs à source finie, le modèle de FAR à source hétérogène et serveurs homogènes, les FAR à serveurs hétérogènes non-fiables, etc. Chaque modèle convient pour la modélisation d'une application pratique particulière, et a ses propres méthodes de résolution. Dans les paragraphes suivants, nous allons décrire quelques uns de ces

modèles.

1.3 Les FAR multiserveurs

Le modèle de files d'attente multi-serveurs avec rappel a toujours attiré l'attention des chercheurs dans ce domaine, depuis l'apparition de ce concept, et ceci grâce à ses diverses applications dans les systèmes de commutation téléphonique. La plupart des modèles étudiés dans la littérature sont inspirés de schémas spécifiques de systèmes de commutation téléphoniques.

Les premiers articles de files d'attente avec rappel $M/M/c$ datent du milieu du dernier siècle [62, 94, 24, 85]. Les cas d'orbite finie ou infinie et la possibilité de perte de clients ont été considérés. Les principaux résultats incluent la détermination des équations à l'état stationnaire, les principales caractéristiques probabilistes du système, et des solutions analytiques dans certains cas particuliers.

Dans cette section, nous allons présenter le modèle $M/M/c$, décrire sa modélisation mathématique ainsi que le processus de résolution. Nous allons découvrir que, à cause du phénomène de rappel, l'espace d'états de la chaîne de Markov sous-jacente n'est pas homogène, ce qui crée des complications pour l'obtention des paramètres de performance du système, et ne permet d'avoir des formules explicites que pour le cas $c \leq 2$. Pour les modèles avec plus de deux serveurs, comme nous allons montrer, la distribution stationnaire est calculée à l'aide d'algorithmes numériques et de techniques d'approximations [38, 12, 97].

Nous considérons un système $M/M/c$ de file d'attente avec rappel, auquel les appels primaires des clients arrivent suivant un processus de Poisson de paramètre λ. Le service est garanti par c serveurs identiques, indépendants et parallèles, dont le temps de service suit une loi exponentielle de taux μ. Le système n'a aucune file d'attente. Les clients trouvant au moins un serveur libre à leur arrivée, occupent immédiatement une place en service et quittent le système dès que ce dernier se termine. Par contre, les clients qui trouvent tous les serveurs pris, sont obligés de rejoindre l'orbite de rappel, et de là, ils rentrent en concurrence avec les appels primaires, car ils commencent à générer des flux de rappel poissonniens de taux ν.

De plus, nous supposons que les processus des appels primaires, de service et de rappel sont mutuellement indépendants.

A tout instant t, l'état du système peut être représenté par le processus bidimentionnel suivant $X(t) = \{C(t), N(t); t \geq 0\}$, tel que $C(t)$ est le nombre de serveurs occupés et qui désigne aussi le nombre de clients en service et $N(t)$ le nombre de clients en orbite. Le processus $X(t)$ est une chaîne de Markov à temps continu homogène, irréductible, avec l'espace d'états $S = \{1, ..., c\} \times \mathbb{Z}_+$ [7].

La principale caractéristique de ce modèle est l'infinité de son espace d'états $|S| = \infty$, et l'hétérogénéité de cet espace, causée par les appels répétés.

Le générateur infinitésimal Q du processus de Markov $X(t)$, est défini par ses taux de transitions $q_{(i,j)(m,n)}$ comme suit :

1. $0 \leq i \leq c-1$

$$q_{(i,j)(m,n)} = \begin{cases} \lambda, & \text{si } (m,n) = (i+1, j), \\ i\mu, & \text{si } (m,n) = (i-1, j), \\ j\nu, & \text{si } (m,n) = (i+1, j-1), \\ -(\lambda + i\mu + j\nu), & \text{si } (m,n) = (i,j), \\ 0, & \text{sinon.} \end{cases}$$

2. $i = c$

$$q_{(i,j)(m,n)} = \begin{cases} \lambda, & \text{si } (m,n) = (c, j+1), \\ c\mu, & \text{si } (m,n) = (c-1, j), \\ -(\lambda + c\mu), & \text{si } (m,n) = (c,j), \\ 0, & \text{sinon.} \end{cases}$$

La figure 1.3 est une illustration des transitions entre les états de $X(t)$ pour le cas $c = 3$.

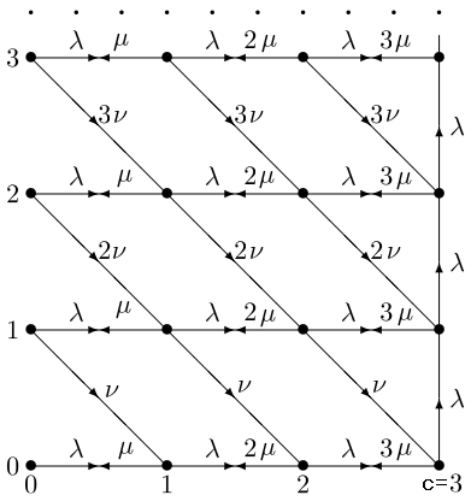

FIG. 1.3 – Transitions entre les états du processus X

Si on ordonne les états comme suit : $S = \{(0,0), ..., (c,0), (0,1), ..., (c,1), ...\}$, on pourra exprimer le générateur infinitésimal Q du processus $X(t)$ sous la forme d'un bloc de matrices comme suit [7] :

$$Q = \begin{pmatrix} A_{00} & A_{01} & 0 & \cdots & 0 & \cdots \\ A_{10} & A_{11} & A_{12} & \cdots & 0 & \cdots \\ 0 & A_{21} & A_{22} & \cdots & 0 & \cdots \\ 0 & 0 & A_{32} & \cdots & 0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

$A_{j,j+1}$, $A_{j,j-1}$et $A_{j,j}$ sont des matrices carrées d'ordre $(c+1)$

$$Aj, j+1 = diag \begin{pmatrix} 0 & 0 & \cdots & \lambda \end{pmatrix}$$

$$Aj, j-1 = \begin{pmatrix} 0 & j\nu & 0 & \cdots & 0 \\ 0 & 0 & j\nu & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & j\nu \\ 0 & \cdots & \cdots & \cdots & 0 \end{pmatrix}$$

$$A_{j,j} = \begin{pmatrix} a_{0j} & \lambda & 0 & \cdots & 0 \\ \nu & a_{1j} & \lambda & \cdots & 0 \\ 0 & 2\nu & a_{2j} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \lambda \\ 0 & 0 & 0 & c\nu & a_{cj} \end{pmatrix}$$

où : $a_{ij} = -(\lambda + i.\mu + (1 - \delta_{ic}).j.\nu)$, pour $0 \leq i \leq c$ et $j \geq 0$, où δ_{ij} connu sous *delta de Kronecker* est défini par :

$$\delta_{ij} = \begin{cases} 1, & \text{si } i = j, \\ 0, & \text{sinon.} \end{cases}$$

D'un point de vue pratique, les principaux paramètres de performance du système, en plus de la répartition probabiliste à l'état stationnaire, sont :

(i) la probabilité de blocage ($B = \lim_{t\to\infty} P[C(t) = c]$),

(ii) Le nombre moyen de clients en orbite ($N = \lim_{t\to\infty} E[N(t)]$), et

(iii) le nombre moyen de serveurs occupés ($Y = \lim_{t\to\infty} E[C(t)]$).

Lors de l'étude de tels systèmes, le premier point à vérifier est bien la récurrence positive (l'ergodicité) du processus X, cela garantit l'existence du régime permanent. En effet, pour que le processus X soit ergodique, on suppose que l'intensité du trafic $\rho = \lambda/c.\mu < 1$. Ceci peut être expliqué informellement par le fait que le nombre moyen de serveurs occupés doit être inférieur au nombre total de serveurs disponibles [24, 28]. Dans ce cas, le régime stationnaire existe, et par conséquent, les probabilités stationnaires

$$p = \{p_{ij}/p_{ij} = \lim_{t\to\infty} P[C(t) = i, N(t) = j]; (i, j) \in S\}$$

peuvent être obtenues à l'aide des équations de Kolmogorov $p.Q = 0^t$ où 0 est le vecteur colonne d'ordre infini d'éléments nuls, 0^t est sa transposée. En divisant le vecteur des probabilités limites p tel que $p = (p(0), p(1), ...,)$ où $p(j) = (p_{0j}, ..., p_{cj})$, on peut donc écrire la forme matricielle des équations de Kolmogorov suivante :

$$p(j-1)A_{j-1,j} + p(j)A_{j,j} + p(j+1)A_{j+1,j} = 0^t_{c+1}, j \geq 0$$

où $p(-1)$ et $A_{-1,0}$ sont nuls, et 0_{c+1} est le vecteur colonne d'ordre $c+1$ de zéros.

Dans le cas où $c \leq 2$, des solutions analytiques explicites peuvent être trouvées [38, 7] car les probabilités stationnaires satisfont un ensemble d'équations de type de naissance et de mort. Cependant, dans le cas où $c > 2$, à cause de l'absence d'une structure de type de naissance et de mort, aucune solution en forme analytique (exacte) n'est encore connue [7]. Certaines approches théoriques fournissent des solutions sous forme d'intégrales de contour [24], ou bien comme limite de fractions continues étendues [80]. Cependant, malgré leur intérêt théorique dans le domaine des FAR, elles ne conviennent pas à des implémentations pratiques car les probabilités limites p_{ij} ne peuvent toujours pas être exprimées sous une forme qu'on peut traiter et ne conduisent pas à un calcul récursif direct.

L'absence de formules analytiques explicites pour les probabilités stationnaires, empêche bien évidemment de dériver les principaux indices de performance du modèle de FAR multi-serveurs quand le nombre de serveurs est supérieur à deux. Ceci a incité les chercheurs à diriger leurs efforts aux *méthodes numériques* pour la résolution du système d'équations de Kolmogorov, mais comme ce système est infini, il ne peut pas être résolu directement même par un ordinateur. De plus, aucune transformation réduisant cet ensemble d'équations en une solution d'un problème fini dans le cas général n'est disponible. C'est pourquoi des méthodes de *solutions numériques approximatives* de ce système sont apparues.

Ces méthodes peuvent être classifiées en trois catégories [12] :

– *Les approximations* : Cette catégorie englobe toutes les solutions qui consistent à remplacer le modèle original par un autre modèle plus simplifié. Cette classe n'est efficace que dans certains domaines des paramètres du système, ou dans des cas particuliers extrêmes (trafic intense, un flux de rappel d'une faible intensité, etc.).

  Une approche intuitive est de traiter les appels répétés comme de nouveaux appels. Avec cette approche ([38], section 2.8), le système devient un simple modèle avec perte, dont la solution peut être obtenue en utilisant les formules

B d'Erlang avec un taux d'arrivée qui est la somme des taux d'arrivées des nouveaux clients et des clients en orbite. L'inconvénient réside dans la dégradation de la qualité de l'approximation quand le taux de rappel devient important. D'autre part, pour des taux de rappel élevés, le système peut être approché par un modèle $M/M/c$ classique avec buffer, et la solution est calculée en utilisant les formules *C d'Erlang*. Ces deux méthodes étant adaptées aux faibles et fortes intensités de trafic, respectivement, une interpolation entre les deux peut être proposée pour avoir une solution plus précise.

Dans [46] Greenberg et Wolf ont proposé une autre alternative basée sur l'hypothèse que la probabilité à l'état d'équilibre qu'un client arrivant voit le système à l'état i est la même que la probabilité limite que le système soit à l'état i (*See Time Average*). Cette approximation est supposée être plus précise quand le taux de rappel diminue, en fait, quand un client bloqué fait sa tentative de rappel plus rapidement, la probabilité de trouver le système occupé est plus élevée.

– *Les modèles tronqués finis* : L'espace d'état original infini est remplacé par un autre fini, le modèle résultant est donc traitable.

La premier modèle tronqué fini était proposé par Wilkinson dans [94]. Il consiste à tronquer la taille de l'orbite à une taille finie q, ainsi l'espace des états du système sera fini.

Un autre modèle plus efficace [89] repose sur l'exclusion des états dont la probabilité stationnaire est négligeable.

Il existe aussi d'autres modèles tronqués finis dont l'idée est de modifier l'espace des états pour introduire, en quelque sorte, l'effet du phénomène de rappel. Le modèle présenté par Fredericks et Reisner [41], réduit la dimension de l'espace d'états, en éliminant la dimension correspondant au nombre de clients en orbite, et introduisant son effet comme un nouveau taux d'arrivée qui dépend de l'état du système.

D'une manière similaire, Marsan et al. [72] ont proposé une méthode qui réduit la taille du problème, en regroupant tous les niveaux du processus QBD (Quasi birth and death) où il y a des clients en orbite en un seul niveau.

Récemment, ARTALEJO et AMADOR [5] ont publié une nouvelle technique qui consiste à remplacer le modèle original dont l'orbite est infinie par un autre modèle avec un groupe de rappel fini, puis utiliser une fonction génératrice pour le calcul des moments et des covariances de nouveaux paramètres de performance qui sont : le nombre de rappels réussis/bloqués et le nombre de nouveaux appels réussis/bloqués durant une période d'occupation (a busy period).

– *Les modèles tronqués généralisés* : Dans ce cas, l'espace d'état d'origine est remplacé par un autre infini, mais assez simplifié pour être soluble. Ces modèles sont apparus pour palier aux problèmes des modèles tronqués finis qui résident dans la plupart du temps dans le besoin en ressources de calcul très puissantes. Ils procurent aussi l'avantage d'une meilleure précision que celle des méthodes tronquées finies.

La première méthode a été proposée par FALIN dans [39] et repose sur l'hypothèse que le taux de rappel devient infini quand le nombre de clients en orbite dépasse un certain seuil Q. Donc, le système peut être vu comme une file d'attente $M/M/1$ standard, et le taux de rappel dépendra du nombre de clients en orbite comme suit :

$$\nu_m = \begin{cases} m\nu & \text{si } 0 \leq m \leq Q \\ \infty & \text{si } m \geq Q+1 \end{cases}$$

Un second modèle a été présenté par NEUTS et RAO [76], il est basé sur l'homogénéisation du processus à partir d'un certain rang Q. Ceci suppose que le nombre de clients autorisés à rappeler est restreint à Q, le taux de rappel devient :

$$\nu_m = min(m, Q)\nu,\ m \geq 0$$

Le modèle approximatif résultant peut être traité par une approche géométrique matricielle (*matrix-geometric approach*) [75].

ARTALEJO et POZO [12] ont pensé à une approximation basée sur le modèle $M/M/2$ au lieu de $M/M/1$. Par conséquent, le taux de rappel ne dépendra

pas uniquement du nombre de clients en orbite m mais aussi du nombre de serveurs occupés k comme suit :

$$\nu_{mk} = \begin{cases} \infty & \text{si } 0 \leq k \leq c-2 \text{ et } m \geq Q \\ m\nu & \text{sinon} \end{cases}$$

En se basant sur le modèle de Neuts et Rao, Ma Jose Domenech-Benlloch et al. [31] ont présenté récemment, deux autres méthodes généralisées pour le modèle avec rappel $M/M/c$ avec des clients impatients qu'ils ont appelées $HM1$ et $HM2$ respectivement. En gardant toujours le principe d'homogénéisation, $HM1$ rapproche le taux de rappel au delà du niveau Q par une estimation de la moyenne de ce dernier $M = E[m \mid m \geq Q]$, le nouveau taux de rappel est alors :

$$\nu_m = \begin{cases} m\nu & \text{si } m < Q \\ M\nu & \text{si } m \geq Q \end{cases}$$

Le sens physique de l'approximation proposée est basé sur l'affectation au taux de rappel, une valeur qui est le nombre moyen de clients en orbite, quand le nombre de ce dernier est supérieur ou égal à Q. La méthode $HM2$ est conçue pour palier aux insuffisances de $HM1$, notamment quand la probabilité de blocage devient faible, cependant, son coût de calcul croît dans cette condition.

1.4 Les FAR à source finie

Les FAR à source finie de clients, appelées aussi *les FAR à entrée quasi-aléatoire* sont motivées par le fait que dans de nombreuses applications, le nombre de clients potentiels est loin d'être infini. Dans ces applications, le flux d'arrivées des clients n'est plus poissonnien, mais dépend du nombre de clients déjà dans le système ; Ceci est dû au fait que le taux de générations de nouveaux appels primaires décroît au fur et à mesure que le nombre de clients dans le système croît [7].

Les FAR avec un nombre fini de clients absolument persistants a été présenté pour la première fois, en 1969, par Kornyshev [60] qui a écrit des équations de

Kolmogorov pour la distribution du nombre de serveurs occupés et le nombre de clients en orbite, a présenté les caractéristiques de performance principales, obtenu des relations entre ces caractéristiques, et a proposé un algorithme numérique dans le cas de temps de service exponentiel. Dans [61], KORNYSHEV a numériquement étudié l'influence des paramètres du système sur les principaux indices de performance.

Plu tard, DE KOK [27] a étudié le cas d'un système à serveur unique et nombre fini de clients, à l'aide de la théorie des processus de régénération, il a obtenu un schéma récursif pour le calcul de la distribution stationnaire de l'état du système et de la longueur de la file d'attente. En fait, le modèle à source finie a été traité comme un cas particulier d'un modèle plus général, où le taux d'arrivée d'un nouvel appel primaire est une fonction générale λ_{ij} du nombre de clients en service (i), et le nombre de clients dans l'orbite (j).

En se basant sur des transformations à l'état discret, OHMURA et TAKAHACHI [77] ont dérivé un ensemble de formules récursives pour le calcul de la distribution stationnaire de l'état du serveur et de la longueur de la queue. Ils ont également proposé l'application de ce modèle à l'analyse du temps d'attente pour l'accès à la mémoire à disque magnétique.

DRAGIEVA [32] a étudié le nombre d'appels dans le système ainsi que le processus de temps d'attente. Toutefois, le processus de temps d'attente a été étudié uniquement dans le cas des temps de service exponentiels ou bien déterministes, et seulement les indices les plus importants ont été obtenus. Il convient de noter aussi que certains résultats semblaient être incorrects [37].

Les FAR à source finie et serveur unique sujet à des vacances, ont fait l'objet des travaux de LI et YANG dans [66] où ils ont développé un algorithme direct permettant d'obtenir la distribution de l'état du serveur ainsi que la longueur de la file d'attente en régime permanent. Ils ont appliqué ce système à l'analyse des performances d'un réseau LAN.

FALIN et ARTALEJO [37] ont employé la même approche de OHMURA et TAKAHACHI [77] pour améliorer les expressions des principaux indices de performance du système $M/G/1//K$ (la distribution d'arrivées de clients, le processus de temps d'attente, etc.).

Dans leur livre [38, section 4.3], FALIN et TEMPLETON ont présenté une analyse plus détaillée du modèle $M/M/s//K$, ils ont exprimé les principaux indices de performance, y compris le processus du temps d'attente dont l'étude est spécialement compliquée pour les FAR à cause de la superposition des deux flux d'arrivée, à savoir le flux d'appels primaires et celui de rappels [40].

Dans cette section, on propose de considérer le modèle $M/M/s//K$ avec rappel étudié par FALIN et TEMPLETON [38, 40, 6].

Considérons un modèle de file d'attente avec rappel, doté de c serveurs parallèles et identiques, et alimenté par une source de clients supposés persistants de taille K finie ; $c < K < \infty$. Chaque client peut être dans l'un de ces trois états :

- Libre,
- En service,
- ou en orbite,

Quand un client i est libre à l'instant t, il peut générer un appel primaire pendant l'intervalle $(t, t + dt)$ avec une probabilité $\lambda dt + o(dt)$. Si au moins un serveur est inactif au moment de son arrivée alors le service démarre. Le service est terminé dans l'intervalle $(t, t+dt)$ avec une probabilité $\mu dt + o(dt)$. Pendant son temps de service, le client ne peut pas générer une demande de service. Après le service, le client redevient libre et peut (de nouveau) générer un appel primaire. Si tous les serveurs sont occupés au moment de l'arrivée de la demande du client i, ce dernier commence à générer un flux poissonnien d'appels répétés à un taux ν jusqu'à ce qu'il trouve au moins un serveur libre. Comme auparavant, dès que son service se termine, le client redevient libre et peut bien évidemment appeler pour le service (appel primaire). Tous les temps impliqués dans ce modèle sont supposés être indépendants les uns des autres.

Le fonctionnement du système peut être décrit par le processus $\{C(t), N(t)\}$, où $C(t)$ est le nombre de serveurs occupés, $N(t)$ le nombre de clients en orbite à l'instant t. Sous les hypothèses précédemment citées, $(C(t), N(t))$ est un processus markovien à espace d'états fini S tel que $S = \{0, 1, ..., c\} \times \{0, 1, ..., K - c\}$. On remarque que$\mid S \mid = (c + 1)(K - c + 1)$. Ce processus admet un générateur infinitésimal dont les éléments sont les suivants :

1. pour $0 \leq i \leq c-1$

$$q_{(i,j)(n,m)} = \begin{cases} (K-i-j)\lambda, & \text{si } (n,m) = (i+1,j), \\ i\mu, & \text{si } (n,m) = (i-1,j), \\ j\nu, & \text{si } (n,m) = (j+1,j-1), \\ -[(K-i-j)\lambda + i\mu + j\nu], & \text{si } (n,m) = (i,j), \\ 0, & \text{sinon.} \end{cases}$$

2. pour $i = c$

$$q_{(c,j)(n,m)} = \begin{cases} (K-c-j)\lambda, & \text{si } (n,m) = (c,j+1), \\ c\mu, & \text{si } (n,m) = (c-1,j), \\ -[(K-c-j)\lambda + c\mu], & \text{si } (n,m) = (c,j), \\ 0, & \text{sinon.} \end{cases}$$

Étant donné que le nombre d'états de la CMTC $\{C(t), N(t)\}$ est fini, en outre, on peut facilement démontrer qu'elle est irréductible, elle est donc ergodique pour toutes les valeurs de λ, ν et μ, et par conséquent, il existe une distribution stationnaire unique.

Notons par $p_{ij}, (i,j) \in S$ la probabilité qu'il y ait i serveurs occupés, j clients en orbite à l'état stationnaire. Les probabilités $p_{ij} = P[C(t) = i, N(t) = j]$ satisfont les équations de Kolmogorov suivantes ($p_{ij} = 0$ si $(i,j) \notin S$) :

1. $[(K-i-j)\lambda + i\mu + j\nu]p_{ij} = (K-i+1-j)\lambda p_{(i-1)j} + (j+1)\nu p_{(i-1)(j+1)} + (i+1)\mu p_{(i+1)j}$ si $0 \leq i \leq c-1$
2. $[(K-c-j)\lambda + c\mu]p_{sj} = (K-c+1-j)\lambda p_{(c-1)j} + (j+1)\nu p_{(c-1)(j+1)} + (K-c-j+1)\lambda p_{s(j-1)}$

Nous donnons ici un résumé de quelques indices de performance exprimés en fonction de p_{ij}, sachant que Falin a établi un algorithme récursif calculant ces probabilités stationnaires [40] :

– Le nombre moyen de clients en orbite

$$N = \sum_{i=0}^{c} \sum_{j=0}^{K-c} j.p_{ij}$$

– La probabilité que tous les serveurs soient occupés

$$P_c = \sum_{j=0}^{K-c} p_{cj}$$

– Le nombre moyen de serveurs occupés

$$Y = \sum_{i=0}^{c} \sum_{j=0}^{K-c} i.p_{ij}$$

– La fréquence d'arrivée d'appels primaires

$$\bar{\lambda} = \lambda - Y - N$$

– Le temps moyen d'attente

$$W = \frac{N}{\bar{\lambda}}$$

Les FAR à entrée quasi-aléatoire sont d'un très grand intérêt pour la modélisation de divers systèmes comme les systèmes à disque mémoire magnétique [77], les réseaux mobiles cellulaires [91], les réseaux informatiques [48], les réseaux locaux (LAN) avec les protocoles CSMA/CD non persistants de topologie en étoile [51] , avec des protocoles à accès aléatoire [54], et avec de multiples protocoles d'accès [58], les systèmes hybrides fibre-coaxial [48], etc. pour n'en citer que quelques-uns.

1.5 Les FAR à source finie hétérogène

Un parcours de la littérature des FAR nous a montré que le cas du modèle à source finie et clients hétérogènes a été traité pour la première fois par l'équipe constituée d'Almasi, Bolch, Roszik et Sztrik. Ils ont d'abord étudié dans [2] le modèle $\vec{M}/M/1//K$ (mono-serveur), puis ils ont généralisé au modèle multi-serveurs

[3], ainsi qu'au modèle à serveur unique non fiable [90].

Dans ce paragraphe, nous allons voir avec plus de détails le modèle de FAR multi-serveurs à source finie hétérogène.

Nous considérons une file d'attente avec rappel $\vec{M}/M/c//K$ avec une station de service comprenant c serveurs homogènes, les appels primaires sont générés par K, $c < K < \infty$ clients *hétérogènes*, i.e. le taux d'arrivées, de rappel et celui de service varient d'un client à un autre. Ainsi si un client i, $(i = 1, ..., K)$ est libre (i.e. il n'est ni en service ni en orbite) à l'instant t, il peut générer un appel primaire durant l'intervalle $(t, t + dt)$ avec la probabilité $\lambda_i dt + o(t)$, si un des serveurs est disponible, le service commence immédiatement et le serveur devient occupé. La durée de service est exponentiellement distribuée avec le paramètre μ_i. Si tous les serveurs sont occupés au moment de l'arrivée de l'appel primaire du client i, ce dernier commence à générer un flux de rappels suivant la loi de Poisson avec le taux ν_i jusqu'à ce qu'il rencontre un serveur disponible. Après le service, le serveur devient libre et le client peut générer un appel primaire. Tous les temps inclus dans ce modèle sont mutuellement indépendants. Notons que, contrairement aux modèles précédents, les paramètres λ, ν et μ sont indexés pour exprimer l'hétérogénéité entre les clients, comme le montre la figure 1.4.

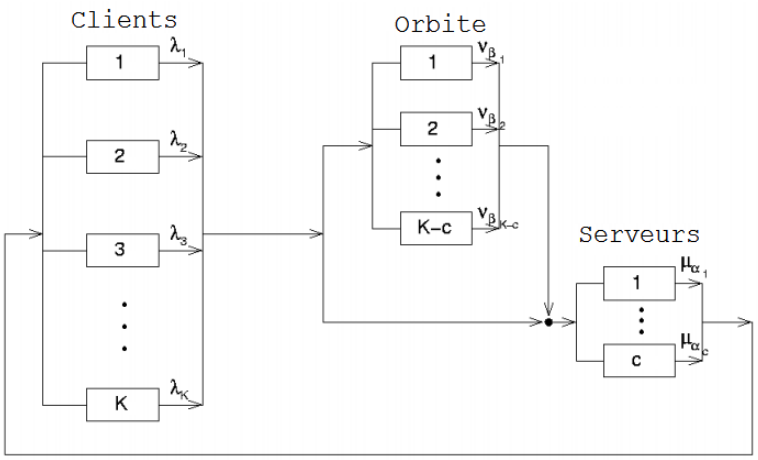

FIG. 1.4 – Le modèle $\vec{M}/M/c//K$

L'évolution de ce système peut être décrite par une chaîne de Markov $X(t)_{t\geq 0}$ à temps continu, ergodique, telle que :

$X(t) = (\alpha_1, ..., \alpha_c, \beta_1, ..., \beta_{N(t)})$

où

- $N(t)$ est le nombre de clients en orbite à l'instant t, c ; le nombre de serveurs,
- Les clients en service sont notés par $\alpha_i,\ i = 1, ..., c$,
- Les clients en orbite sont notés par $\beta_j,\ j = 1, ..., N(t)$.

L'étape suivante serait de calculer les probabilités stationnaires, puis en dériver les différents indices de performance (La probabilité qu'un client soit en orbite ou en service, le taux moyen d'appel d'un client i, le temps moyen d'attente...). Cependant, à cause de la taille d'espace d'états de la chaîne de Markov qui peut être très volumineuse, il est difficile de calculer les indices de performance du système, avec la méthode traditionnelle qui consiste à résoudre l'ensemble des équations au régime stationnaire. Pour simplifier la procédure, les auteurs ont utilisé un outil logiciel, *MOSEL* (*MOdeling, Specification and Evaluation Language*), développé à l'université d'*Erlangen* en *Allemagne* [16, 15], pour formuler et résoudre le problème. Les lecteurs intéressés aux détails de ce travail sont invités à consulter le document [3].

Une généralisation du modèle à source finie hétérogène consiste à considérer un certain nombre n de classes de clients, chacune comporte un ou plusieurs clients homogènes, en d'autres termes, l'intensité d'arrivée d'appels primaires de la $i^{ème}$ classe est λ_i, celle de rappel est ν_i, et le taux de service est μ_i ([36], section 12). En 1983, KULKARNI [64] a proposé d'étudier le modèle multi-classes à serveur unique et distribution générale du temps de service, en considérant deux classes de clients. Pour le cas général du nombre de classes, FALIN [35] a obtenu des formules explicites pour le nombre moyen de clients en orbite pour chaque classe.

Plus tard, les modèles multi-serveurs avec rappel et deux types de clients ont été étudiés dans [19, 21] par CHOI et CHANG. Les auteurs ont donné, de plus, une synthèse des modèles à serveur unique avec rappel et deux classes de clients, ainsi que leurs applications dans [20].

Récemment, GHARBI [42, 43] a proposé une modélisation des systèmes avec rappel avec plusieurs classes de clients, et plusieurs classes de serveurs en se basant sur une alternative très intéressante qui est *les réseaux de Petri stochastiques généralisés colorés* (*RdPSGC*). En plus de l'hypothèse très large pour le nombre de classes de clients ainsi que de serveurs, qui est bien évidemment plus difficile à analyser, elle a

réussi, pour la première fois, à dériver des formules explicites pour un grand nombre d'indices de performance comme le nombre moyen de clients libres, en orbite, et en service pour chaque classe, le taux d'arrivée d'appels primaires, et de rappels de chaque classe de clients, ainsi que le taux et le temps moyen de service de chaque classe de clients.

1.6 Les FAR avec serveurs hétérogènes

Une FAR à serveurs hétérogènes est une FAR à plusieurs serveurs parallèles se caractérisant chacun par un taux de service particulier. Ceci engendre une différence entre les temps d'exécution des serveurs, et par conséquent, différentes disciplines pour accéder au service peuvent être appliquées, comme la *politique du serveur le plus rapide disponible* (*fastest free server policy*), et celle de *service aléatoire* (*random service*), etc.

Comparant aux autres variantes des files d'attente avec rappel, très peu d'articles ont été consacrés à celles avec serveurs hétérogènes, et ceci malgré l'utilité pratique de ces dernières. En effet, la non-homogénéité des serveurs dans les FAR s'avère d'une très grande complexité lors de l'analyse.

Parmi le peu de contributions que nous avons trouvées dans la littérature, les articles de POURBABAI [84, 83, 82] où il a proposé dans le premier, une étude par approximation du modèle $G/M/K/O$ à population homogène infinie et serveurs hétérogènes, avec deux approches de service, à savoir : l'approche de service *aléatoire* où les clients sont assignés aléatoirement à un des serveurs oisifs, et l'approche de système à *entrée ordonnée* (*An Ordered Entry System*), où les serveurs sont ordonnés de 1 à K, et ils sont affectés aux appels dans cet ordre. Dans les deux cas, POURBABAI a présenté un algorithme récursif afin de déduire les différents indices de performance. Dans le second [83], ce modèle est analysé en termes de rapprochement des deux flux d'arrivées primaire et secondaire par une méthode à deux paramètres. Le troisième document [82] était une analyse du modèle $G/G/K$ avec serveurs hétérogènes par une technique récursive. Il est important de noter que tous les résultats sont obtenus par approximation ou simulation.

Dans [33], EFROSININ et BREUER ont étudié le problème de l'allocation optimale des différents serveurs hétérogènes dans une FAR. Pour cela, ils ont considéré le modèle $MAP/PH/K$ avec rappel et serveurs hétérogènes, avec un processus d'arrivée markovien, une distribution exponentielle de rappel et une distribution de service hétérogène de type *phase-type général* (*general phase-type*). Ils ont développé une procédure numérique qui repose sur l'algorithme itératif de HOWARD [49], afin de proposer une politique optimale pour le contrôle d'allocation.

GHARBI et al. [43], ont proposé une approche basée sur les réseaux de Petri stochastiques généralisés colorés (RdPSGC), pour étudier un cas plus général qui considère plusieurs classes de serveurs, et plusieurs classes de clients, c'est les FAR *multi-classes* ([36], section 12).

Certains travaux ont considéré le cas de serveurs *non fiables* ; i.e. qui sont sujets à des pannes et des réparations aléatoires. Dans la littérature, on trouve des documents qui traitent ces modèles aussi bien dans le cas d'un serveur unique que dans le cas multi-serveurs, et dans le cas à source infinie que finie, voir par exemple [8, 1, 93, 30, 4, 90, 88, 92, 53, 50] et aussi [44, 25, 26]. Cependant, malgré son importance pratique, les travaux sont limités et les résultats restent assez modestes dans ce domaine.

D'après notre recherche bibliographique, très peu d'articles sont consacrés à la non fiabilité des serveurs combinée avec leur hétérogénéité. Nous n'avons trouvé que les travaux de Dr. GHARBI [42] qui s'intéresse aux réseaux de Petri et leurs applications aux systèmes avec rappel, et les articles de SZTRIK et ROSZIK [86, 87] qui traitent ce modèle tout en supposant aussi une source finie de clients. Nous allons exposer les principaux résultats obtenus dans ces deux derniers articles.

Le modèle de FAR $M/\vec{M}/c//K$ à serveurs non-fiables, se caractérise, en plus du nombre fini de clients K, par c serveurs sujets à des pannes aléatoires, tels que chaque serveur a son propre taux de service, de panne et de réparation.

Les serveurs peuvent être dans l'un des deux états, soit *opérationnel* (*up*) ou *non-opérationnel* (*down*). Par ailleurs, un serveur opérationnel peut être soit *occupé* (en service) soit *disponible* (inactif). Les appels primaires forment un processus markovien de paramètre λ. Selon la stratégie de service appliquée, la prise en charge des demandes de service se fait par un des serveurs opérationnels libres choisi au

hasard dans le cas de la stratégie aléatoire, ou par le serveur disponible (opérationnel et libre) le plus rapide dans le cas de la stratégie du serveur le plus rapide. Le service associé au $i^{ème}$ serveur se termine dans l'intervalle $(t, t+dt)$ avec la probabilité $\mu_i dt + o(dt)$, $(i = 1, ..., c)$. De plus, chaque serveur i peut tomber en panne pendant l'intervalle $(t, t+dt)$ avec la probabilité $\delta_i dt + o(dt)$ s'il est inactif et avec la probabilité $\gamma_i dt + o(t)$ s'il est actif (occupé). Si $\delta_i = 0, \gamma_i > 0$, on parle de *pannes actives*, par contre si $\delta_i, \gamma_i > 0$ alors les pannes sont dites *indépendantes*. Le réparateur suit une discipline FIFO pour le dépannage des serveurs en panne, et le temps de réparation du $i^{ème}$ serveur est distribué de façon exponentielle avec une moyenne $1/\tau_i$. Si tous les serveurs sont en panne, deux cas différents peuvent être considérés ; à savoir, le cas des *sources bloquées* quand toutes les opérations sont suspendues, y compris la génération d'appels primaires et répétés, à l'exception de la réparation des serveurs, et le cas des *sources intelligentes* (*non bloquées*) où seulement le service est interrompu, toutes les autres opérations continuent. Après le service, le client devient libre, et il peut générer un appel primaire de nouveau, et le serveur est de nouveau inactif de sorte qu'il peut servir un nouvel appel. D'autre part, si tous les serveurs sont soit occupés soit en panne lors de l'arrivée d'un nouveau client, ce dernier entre en orbite et commence à générer un flux poissonnien de rappel de taux ν. Tous les temps impliqués dans ce modèle sont supposés être mutuellement indépendants.

L'état du système à l'instant t peut être décrit par le processus

$$X(t) = \{(\alpha_1(t), ..., \alpha_c(t), N(t))\},\ t \geq 0$$

où $N(t)$ est le nombre de clients en orbite, et $\alpha_i,\ i = 1, ...c$ peut prendre trois valeurs selon l'état du $i^{ième}$ serveur ; si celui-ci est occupé, $\alpha_i(t) = 1$, s'il est opérationnel et inactif, alors $\alpha_i(t) = 0$; autrement ce serveur est en panne et $\alpha_i(t) = -1$.

Les probabilités stationnaires peuvent être définies ensuite comme suit :

$$P[i_1, ..., i_c, j] = \lim_{t \longrightarrow \infty} P[\alpha_1(t) = i_1, ..., \alpha_c(t) = i_c, N(t) = j],$$

où : $i_1, ..., i_c = -1, 0, 1$

$j = 0, ..., K^*$ tel que : $K^* = K - \sum_{i_k, i_k=1} i_k$.

On note aussi le nombre de serveurs occupés à l'instant t par $C(t)$, et le nombre de serveurs disponibles par $A(t)$.

En suivant la méthode classique qui consiste à élaborer les équations de Kolmogorov pour déduire les probabilités stationnaires, les auteurs ont pu dériver les indices de performance suivants [86, 87] :

– L'utilisation du $k^{ième}$ serveur

$$U_k = \sum_{\substack{i_1, i_2, ..., i_c \\ i_k = 1}} \sum_{j=0}^{K^*} P(i_1, ..., i_c, j), \ k = 1, ..., c$$

– Le nombre moyen de clients en service

$$C = \sum_{k=1}^{c} U_k$$

– Le nombre moyen de clients en orbite

$$N = \sum_{i_1, ..., i_c} \sum_{j=1}^{K^*} j P(i_1, ..., i_c, j)$$

– Le nombre moyen de clients dans le système

$$M = N + C$$

– L'utilisation du réparateur

$$U_R = \sum_{\substack{i_1, ..., i_c \\ -1 \in \{i_1, ..., i_c\}}} \sum_{j=0}^{K^*} K^* P(i_1, ..., i_c, j)$$

– L'utilisation des sources

$$U_{SO} = \begin{cases} \frac{E[K - C(t) - N(t); A(t) > 0]}{K}, & \text{pour sources bloquées,} \\ \frac{E[K - C(t) - N(t)]}{K}, & \text{pour sources non-bloquées.} \end{cases}$$

– L'utilisation globale du système

$$U_G = C + K.U_{SO} + U_R$$

– Le taux moyen de génération d'appels primaires

$$\bar{\lambda} = \begin{cases} \lambda.E[K - C(t) - N(t); A(t) > 0], & \text{pour sources bloquées,} \\ \lambda.E[K - C(t) - N(t)], & \text{pour sources non-bloquées.} \end{cases}$$

– Le temps moyen d'attente

$$E[W] = \frac{N}{\bar{\lambda}}$$

– Le temps moyen de réponse

$$E[T] = \frac{M}{\bar{\lambda}}$$

1.7 Conclusion

Depuis leur apparition, la modélisation par files d'attente avec rappel a connu un essor considérable dû en particulier à la diversité et à l'omniprésence de ces dernières : réseaux téléphoniques et sans fil [91, 78], réseaux informatiques , centres d'appels [68], protocoles de communication, etc.

Après avoir présenté les notions de base liées aux files d'attente, l'objectif de ce chapitre était de donner un aperçu global sur l'état de l'art des files d'attente avec rappel markoviennes multi-serveurs. En particulier, nous nous sommes intéressés au modèle $M/M/c$, nous avons trouvé que la considération d'un second flux qui est le flux des appels répétés engendre des difficultés analytiques lors du calcul des probabilités stationnaires, ceci empêche l'existence de formules analytiques explicites pour les paramètres de performance quand le nombre de serveurs dépasse deux. Dans ce cas, des méthodes numériques et des méthodes d'approximation sont employées pour obtenir la distribution stationnaire et certains indices de performance [38, 7].

Nous nous sommes intéressés aussi aux FAR avec source finie de clients, où nous avons exposé les résultats obtenus par Falin et Templeton [38, 40, 6] pour le cas des clients homogènes. Nous avons aussi parlé de la contribution de l'équipe des chercheurs de l'université de *Debrecen* concernant le modèle multi-serveurs à source finie hétérogène et à serveurs hétérogènes [3,86,87] basée sur une méthode numérique en utilisant l'outil logiciel MOSEL.

Cependant, il est important de préciser que les études des modèles avec rappel et serveurs hétérogènes restent à ce jour assez rares et les résultats assez limités.

À cet effet, l'objectif de notre travail dans le cadre de ce mémoire est de proposer une approche algorithmique pour l'analyse des modèles avec rappel et serveurs hétérogènes, en utilisant les réseaux de Petri stochastiques généralisés.

Chapitre 2

Réseaux de Petri

2.1 Introduction

Depuis leur apparition dans la thèse *« communication avec les automates »* présentée par C.A Petri à l'université de *Darmstadt* en 1962 [81], *les réseaux de Petri* n'ont cessé d'évoluer : d'un simple moyen pour l'analyse structurelle des systèmes de communication, à un outil puissant dans la description, la modélisation et l'étude formelle de beaucoup de systèmes distribués et concurrents où la notion d'évènements et d'évolutions simultanées ont une importance, tels que les systèmes manufacturiers, les systèmes de télécommunication, les systèmes informatiques et les réseaux de transport, etc. Ceci afin de permettre leur conception, évaluation et par la suite leur amélioration. L'avantage des réseaux de Petri est qu'ils offrent un moyen graphique et mathématique simple et efficace permettant de concevoir le système modélisé.

Comme outil graphique, les réseaux de Petri apportent une aide importante en visualisant le système modélisé, car ce dernier est vu comme un graphe formé d'un ensemble de places représentant les ressources, et un ensemble de transitions décrivant les opérations ou les évènements qui peuvent surgir. Les places contiennent des jetons indiquant le nombre de ressources disponibles à un instant donné, et sont utilisés pour simuler les activités dynamiques et concurrentes du système. Comme outil mathématique, il est possible de mettre en place des équations d'états, des équations algébriques et d'autres modèles mathématiques gouvernant le comportement du système. Cet outil est utilisé aussi bien par les théoriciens que par les praticiens,

ce qui le rend un intermédiaire puissant entre les laboratoires de recherche et le monde industriel.

Cependant, l'analyse qualitative seule ne suffit pas pour s'assurer qu'un système fonctionne d'une manière correcte. Le besoin d'enrichir le modèle de base des RdP par une spécification temporelle se fit sentir par les chercheurs quelques temps plus tard, afin de permettre une analyse quantitative des systèmes modélisés i.e. de pouvoir répondre à des questions relatives au temps, comme le nombre moyen de requêtes servies par unité de temps, et le taux de perte de paquets dans un réseau de communication... Différentes approches existent pour introduire le concept temporel, incluant la temporisation déterministe d'abord puis stochastique, principalement en associant des délais aux places, jetons ou aux franchissements des transitions. En particulier, les réseaux de Petri *stochastiques RdPS* [73, 74], sont caractérisés par l'association d'une variable aléatoire (généralement une distribution exponentielle négative) au franchissement des transitions.

Les RdPS ont apporté une amélioration non-négligeable dans la modélisation des systèmes dynamiques, en tenant en compte les activités qui consomment une certaine durée de temps. Cependant, la plupart des systèmes pratiques comportent des opérations qui nécessitent un temps aléatoire pour s'exécuter, et d'autres qui surgissent instantanément, comme c'est le cas des opérations d'allocation des ressources, de synchronisation, ou bien des actions purement logiques. Il est clair qu'un RdPS ne peut pas modéliser de telles situations. Pour pallier à ce problème, une autre classe de réseaux de Petri est apparue sous le nom de RdPS *généralisés (RdPSG)* [70, 71].

Comme nous allons le montrer, la technique typique de résolution des RdPSG repose sur la connaissance du *processus stochastique* associé, et la définition de la *chaîne de Markov induite CMI*. Un certain bagage en matière de processus stochastiques et chaînes de Markov est nécessaire afin de pouvoir poursuivre la résolution des RdPSG.

Pour cela, nous avons choisi de commencer ce chapitre d'abord par donner quelques rappels concis sur la théorie des probabilités, notamment les processus stochastiques et les chaînes de Markov (à temps continu et discret). La deuxième partie est consacrée aux réseaux de Petri standards. Leur définition et leur propriétés

les plus importantes sont rappelées. Une discussion sur les méthodes pour vérifier et étudier leur comportement et leurs propriétés sera menée. Ensuite, l'étude des réseaux de Petri stochastiques généralisés sera abordée, en commençant par leur définition, les règles de franchissements qui permettent au RdPSG d'évoluer, et enfin, la technique numérique de résolution des RdPSG sera décrite brièvement. Nous terminons par une conclusion.

2.2 Les chaînes de Markov

2.2.1 Variable aléatoire [67]

Soit Ω un espace fondamental de probabilité, une variable aléatoire est toute application X de Ω vers $\mathbb{R}$ des nombres réels (ou bien un sous-ensemble de $\mathbb{R}$ ou de $\mathbb{N}$), telle que l'inverse de chaque intervalle de $\mathbb{R}$ soit un évènement de Ω :

$$\begin{aligned} X \quad : \quad & \Omega \longrightarrow \mathbb{R} \\ & \omega \longrightarrow X(\omega) \end{aligned}$$

Quand la variable aléatoire prend ses valeurs dans un espace image fini ou infini dénombrable ; i.e. $X(\Omega) = \{x_1, x_2, ...\} \subseteq \mathbb{N}$, on parle d'une variable aléatoire *discrète*, dans le cas contraire, la variable aléatoire est dite *continue*.

2.2.2 Loi exponentielle [79]

La loi exponentielle de paramètre $\lambda \geq 0$, est une variable aléatoire à temps continu, dont la densité de probabilité est définie par :

$$f(t) = \begin{cases} \lambda e^{-\lambda t} & \text{si } t \geq 0 \\ 0 & \text{sinon} \end{cases}$$

Sa fonction de distribution est donnée par,

$$F(t) = \int_0^t \lambda e^{-\lambda x} dx = 1 - e^{-\lambda t},\ t \geq 0$$

et sa moyenne est égale à $\frac{1}{\lambda}$.

Une application très importante pour la loi exponentielle est le "temps jusqu'au prochain appel arrivant dans une centrale téléphonique par exemple", ce temps ne dépend pas de la durée qui s'est écoulée depuis le dernier appel. Cette propriété est appelée *sans mémoire*. En particulier, une variable aléatoire est *sans mémoire* si l'égalité suivante est vérifiée :

$$P\left[X > t + s \mid X > s\right] = P\left[X > t\right]$$

2.2.3 Processus stochastique

Soit T un ensemble d'indices, la collection $X = \{X(t),\ t \in T\}$ de variables aléatoires définies sur le même espace de probabilité est appelé *processus stochastique* [71]. Dans beaucoup d'applications, l'indice t est utilisé pour modéliser le temps. On appelle *espace des états*, l'ensemble E des valeurs prises par l'ensemble des variables aléatoires. Les deux ensembles T et E peuvent être discrets ou continus.

Un processus aléatoire généralise la notion de variable aléatoire, il représente une évolution, généralement dans le temps, de cette dernière.

Une classe particulière des processus stochastiques largement utilisée dans la pratique, notamment dans le domaine des files d'attente, est la classe des *processus de Poisson* [79]. En effet, un processus de Poisson est un processus à temps continu, ordonné et sans mémoire, appliqué dans les cas où un certain évènement se produit à différents instants dans le temps. Un processus de Poisson de paramètre λ peut être décrit par son *processus de comptage* $\{N(t), t \geq 0\}$ représentant le nombre total d'évènements dans l'intervalle $[0, t[$, et qui vérifie les conditions suivantes :

1. Les nombres d'occurrences dans des intervalles de temps disjoints sont indépendants,
2. La probabilité d'une occurrence dans un petit intervalle de temps est propor-

tionnelle à la longueur de cet intervalle, le coefficient de proportionnalité étant λ,

3. La probabilité qu'il y ait plus d'une occurrence dans un petit intervalle de temps est négligeable.

On démontre que les durées de temps s'écoulant entre deux incrémentations successives d'un processus de Poisson de paramètre λ sont des variables aléatoires indépendantes de même *loi exponentielle* de paramètre λ.

Un processus stochastique est *markovien* si, la connaissance de l'état actuel du processus résume toute l'information nécessaire pour connaître son évolution dans le futur, les états précédents n'ont aucune influence. Cette propriété peut s'écrire sous la forme [71] :

$$P\left[X(t) \leq x | X(t_n) = x_n,\, X(t_n - 1) = x_n - 1, \ldots, X(t_0) = x_0\right] \;=$$
$$P\left[X(t) \leq x | X(t_n) = x_n\right], \forall t > t_n > t_{n-1} > \ldots . > t_0$$

La classe des processus markoviens est particulièrement intéressante grâce à cette propriété, connue sous le nom propriété de *perte de mémoire* ou encore *propriété de Markov*, car elle épargne le stockage dans la mémoire centrale les états précédents lors de la modélisation par un processus de Markov.

Nous allons nous focaliser dans les deux sections suivantes sur une classe particulière des processus markoviens, où l'espace des états E est discret. Ceci caractérise les *chaînes de Markov*. Selon que l'ensemble T des intervalles de temps soit discret ou continu, on distingue deux principales catégories :

- Les chaînes de Markov à temps *discret*, si $T \subseteq \mathbb{N}$ (ou $T \subseteq \mathbb{Z}$).
- Les chaînes de Markov à temps *continu*, quand $T \subseteq \mathbb{R}$.

2.2.4 Chaînes de Markov à temps discret (CMTD) [71]

Un processus stochastique à temps discret et à espace discret $X_n, n \in \mathbb{N}$ est une *chaîne de Markov à temps discret* si et seulement si la propriété de Markov suivante est vérifiée :

$$P[X_{n+1} = j_{n+1}|X_n = j_n, \cdots, X_0 = j_0] = P[X_{n+1} = j_{n+1}|X_n = j_n], \forall n \in \mathbb{N}, j_k \in E$$

En d'autres termes, une chaîne de Markov à temps discret (ou *CMTD* en abrégé) a la propriété que son évolution (passage de X_n à X_{n+1}) ne dépend que de l'état courant X_n, et non pas de son passé (les états visités aux instants $0, 1, ..., n-1$). Cette propriété facilite donc la mise en œuvre informatique d'un tel processus. En particulier, il n'est pas nécessaire de conserver en mémoire tout le passé du processus pour effectuer des calculs de performance.

La probabilité $P_{ij}(n) = P[X_n = j|X_{n-1} = i]$ s'appelle *probabilité de transition*, c'est la probabilité que la chaîne se trouve à l'état j à l'instant n sachant qu'elle était à l'état i à l'instant $(n-1)$. Lorsque celle-ci est indépendante de n, la chaîne est dite *homogène dans le temps*, et on écrit [71] :

$$P_{ij} = P[X_n = j|X_{n-1} = i]$$

La propriété de l'homogénéité assure que l'évolution de la chaîne ne dépend pas de n, mais seulement des états concernés. Dans les paragraphes qui suivent, on ne s'intéressera qu'aux chaînes de Markov homogènes.

2.2.4.1 Représentation d'une CMTD homogène [71, 14]

Une CMTD est entièrement définie par la connaissance de sa *matrice de transition* ou de son *graphe d'états.*

Matrice de transition

Soit$(X_n)_n$ une CMTD homogène dont le nombre d'états $|E| = s$. La matrice carrée P d'ordre s et dont les éléments sont les probabilités de transitions entre les états P_{ij}, est appelée *matrice de transition.* Une chaîne de Markov est complètement définie par la connaissance de sa matrice de transition.

Diagramme de transition d'états

On peut également représenter une CMTD homogène par un graphe orienté d'ordre s où les sommets correspondent aux états de la chaîne et les arcs sont étiquetés par les probabilités de transitions non nulles dans P.

Exemple

Considérons une chaîne de Markov à temps discret, et soit $E = \{1, 2, 3\}$, l'ensemble d'états correspondant.

Supposons que la matrice de probabilités de transition de cette chaîne soit :

$$P = \begin{pmatrix} 0.2 & 0.2 & 0.6 \\ 0 & 0.5 & 0.5 \\ 1 & 0 & 0 \end{pmatrix}$$

La figure 2.1 suivante représente le graphe de transitions résultant.

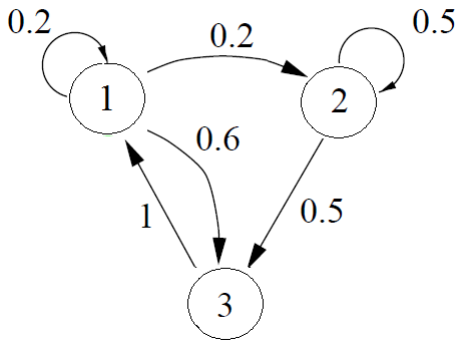

FIG. 2.1 – Diagramme de transition de la CMTD.

Pour décrire l'évolution temporelle d'un système dynamique, la modélisation par chaînes de Markov consiste à définir l'espace d'états dans lequel évolue aléatoirement le système. Ceci permet de calculer les *probabilités d'état stationnaires*. Ces probabilités peuvent être vues comme la probabilité que le système se trouve dans un état donné à un instant choisi *aléatoirement* loin dans le futur. Elles peuvent également être vues comme la proportion de temps que l'on a passé dans cet état au cours d'une très longue observation du système. À partir de ces probabilités, on obtient

des statistiques sur le système, comme le nombre moyen de clients qu'il contient, leur temps de séjour moyen, etc. Cependant, ces probabilités ne sont pas toujours bien définies, en fait, le système doit satisfaire certaines conditions pour que celles-ci existent, dans ce cas, on dit que le système admet un *état stationnaire* (ou bien *permanent*) vers lequel il tend au cours du temps. Dans ce qui suit, nous allons donner quelques définitions nécessaires pour que le système admette une distribution stationnaire, ainsi que la démarche classique de l'étude d'une chaîne de Markov.

2.2.4.2 Évolution d'une chaîne de Markov à temps discret

La distribution initiale, le régime transitoire [14]

La distribution initiale d'une chaîne de Markov à temps discret désigne l'état dans lequel se trouve le système lors du début de l'analyse, elle est représentée par un vecteur de probabilité $\pi^{(0)} = \left[\pi_0^{(0)}, \pi_1^{(0)}, ..., \pi_s^{(0)}\right]$, où $\pi_i^{(0)}$ est la probabilité que le système soit dans l'état i à l'instant initial, $\pi_i^{(0)} = P[X_0 = i]$.

Quand le système est initialement dans l'état i, on a $\pi_i^{(0)} = 1$, et $\pi_j^{(0)} = 0, \forall j \neq i$.

L'étude du régime *transitoire* d'une CMTD consiste à déterminer les vecteurs stochastiques des probabilités des états aux différents instants $\pi^{(n)} = \{\pi_i^{(n)}, i \in E\}$, où : $\pi_i^{(n)} = P[X_n = i]$ est la probabilité que le système soit dans l'état i à l'instant n.

En effet, on a l'égalité suivante :

$$\pi^{(n)} = \pi^{(n-1)}.P = \ldots = \pi^{(0)}.P^n$$

Quand n devient très grand ($n \to \infty$), c.à.d. après l'écoulement d'un temps infini, sous certaines conditions, le vecteur des probabilités des états converge vers un vecteur π ;

$$\pi = \lim_{n \to \infty} \pi^{(n)}$$

On dit dans ce cas là, que le régime *stationnaire* (*permanent*) est atteint, la distribution probabiliste correspondante reste alors stable tout au long du processus après ce temps là. Ceci nous donne la possibilité de calculer plusieurs paramètres de

performance stationnaires du système.

Nous allons voir, maintenant, les propriétés essentielles des CMTD, grâce auxquelles, le régime stationnaire existe.

Apériodicité d'une CMTD [71]

On dit qu'une chaîne de Markov à temps discret est *apériodique* si le PGCD de la longueur de tous les cycles de probabilité non nulle est égal à 1.

De façon équivalente, soit :

$$d(i) = PGCD\left\{k \geq 1 : P^{(k)}(i,i) > 0\right\},$$

où $P^{(k)}(i,j)$ est la probabilité de transition de l'état i à l'état j en k étapes,

$$P^{(k)}(i,j) = P\left[X_{n+k} = j \mid X_n = i\right], \forall n \in \mathbb{N}$$

La chaîne de Markov est apériodique si $\forall i \in E : d(i) = 1$.

Irréductibilité d'une CMTD [71]

On dit qu'une chaîne de Markov à temps discret à valeurs dans E est *irréductible sur* E, si tout état de E est atteignable à partir de tout autre état de E par des transitions de probabilités strictement positives.

De façon équivalente, une CMTD est irréductible sur E si pour toute pair d'états (i,j) de E il existe un entier $n \geq 1$(qui dépend à priori de i et de j) tel que $P^n(i,j) > 0$.

Et en termes de théorie de graphes, une CMTD est irréductible si son graphe d'état est formé d'une seule composante fortement connexe.

Ergodicité d'une CMTD et distribution stationnaire [71]

Une CMTD finie, apériodique et irréductible est dite *ergodique*.

Une CMTD admet une distribution stationnaire si elle est ergodique. Dans ce cas, le vecteur π des probabilités stationnaires existe et est l'unique solution du

système d'équations linéaires suivant :

$$\begin{cases} \pi = \pi P \\ \sum_{i \in E} \pi_i = 1 \end{cases}$$

Processus de naissance et de mort [79] Dans de nombreuses applications, les seules transitions (de la CMTD correspondante) possibles à partir d'un état i sont vers $i-1$ et $i+1$, ceci définit une classe particulière des chaînes de Markov qui s'appelle *les processus de naissance et de mort* (*birth and death process*, *QBD* en abrégé).

En d'autres termes, dans un processus de naissance et de mort, on a :

$$P_{ij} = 0 \text{ si } \mid i-j \mid \neq 1, \text{ et } P_{ij} > 0 \text{ si } \mid i-j \mid = 1$$

Ce qui simplifiera énormément la résolution du système d'équations précédent, permettant d'avoir le vecteur π. Grâce à cette propriété, les processus de naissance et de mort sont très utilisés dans beaucoup de domaines, notamment, en théorie de files d'attente.

2.2.5 Chaînes de Markov à temps continu CMTC

Les concepts introduits dans le cas discret ont tous une analogie dans le cas continu. Dans le cas continu le processus stochastique est observé à des instants arbitraires alors que dans la section précédente le processus est observé à des instants particuliers (discrets).

Un processus stochastique $\{X(t), t \geq 0\}$, est une *chaîne de Markov à temps continu CMTC*, si et seulement si, la propriété de Markov suivante [71] :

$$P[X(t_{n+1}) = j_{n+1} | X(t_n) = j_n, \ldots, X(t_0) = j_0] = P[X(t_{n+1}) = j_{n+1} | X(t_n) = j_n]$$

est vérifiée pour tout $n \in \mathbb{N}$, tout $(n+2) - uplet$ de réels, $t_0 < t_1 < t_2 < \ldots . < t_n < t_{n+1}$, et tout $(n+2) - uplet$ $(j_0, j_1, ..., j_n, j_{n+1})$ d'éléments dans E,

Les instants d'observation $t_0, t_1, \ldots, t_{n+1}$ peuvent être choisis comme on veut

dans le temps, aussi nombreux que l'on veut. Grâce à la propriété de Markov, on n'a pas besoin d'une connaissance détaillée du passé (aux instants $t_0, t_1, \ldots, t_{n-1}$) pour prédire l'état du système à l'instant t_{n+1}, ce dernier ne dépend que de son état à l'instant t_n choisi à un instant quelconque avant t_{n+1}.

La probabilité $P[X(t_{n+1}) = j | X(t_n) = i]$ lorsqu'elle ne dépend pas de n, mais plutôt de la durée qui sépare les deux instants t_n et t_{n+1}, la CMTC correspondante est dite *homogène*, la probabilité précédente devient [71, 14] :

$$P_{ij}(t) = P\left[X(s+t) = j | X(s) = i\right] \text{ pour tout } s \geq 0$$

2.2.5.1 Représentation d'une CMTC homogène [71, 14]

Comme dans le cas des CMTD, les CMTC peuvent être décrites soit par un graphe de transition d'état, soit par une matrice de taux de transitions appelée *générateur infinitésimal.*

Matrice de taux de transition (générateur infinitésimal)

En effet, dans les chaînes de Markov à temps continu, en plus des probabilités de transitions p_{ij} (La probabilité de visiter j en quittant i), on considère ce qu'on appelle des *taux de transition* μ_{ij}. Quand le processus entre dans l'état i, il y reste une durée aléatoire de distribution *exponentielle* de paramètre μ_i, puis saute instantanément vers l'état $j \neq i$, avec la probabilité p_{ij}, le temps de transition de i vers j est exponentiel de paramètre $\mu_{ij} = \mu_i \times p_{ij}$. Ainsi, μ_{ij} est le nombre moyen de transitions de l'état i vers l'état j par unité de temps.

Le générateur infinitésimal Q est une matrice carrée d'ordre $s = |E|$, dont les éléments q_{ij}, $(i \neq j)$ correspondent aux taux de transitions μ_{ij} ; $q_{ij} = \mu_{ij}$, les éléments de la diagonale q_{ii} sont, par définition, égaux à l'opposé de la somme des autres éléments de la ligne :

$$q_{ij} = \begin{cases} \mu_{ij} & \text{si } i \neq j \\ -\sum_{k=1, k \neq i}^{s} \mu_{ik} & \text{si } i = j \end{cases}$$

Ainsi, le générateur infinitésimal Q d'une CMTC caractérise parfaitement le comportement de cette CMTC, comme la matrice de transition P le fait pour une CMTD.

Diagramme de transition

Comme pour les chaînes de Markov à temps discret, on a l'habitude de représenter les chaînes de Markov à temps continu par leur *diagramme de transition*, qui est un graphe orienté, dont les sommets sont les états, et les arcs les transitions qui ont un taux de transition non nul dans Q. Autrement dit, il y aura un arc de l'état i à l'état j si $q(i,j) \neq 0$. Attention au fait qu'on ne représente jamais de transition entre i et i.

Exemple

Considérons une chaîne de Markov à trois états. Autrement dit, soit $E = \{1, 2, 3\}$. Supposons que le générateur infinitésimal de cette chaîne soit :

$$Q = \begin{pmatrix} -0.3 & 0.3 & 0.0 \\ 0.5 & -1.0 & 0.5 \\ 0.2 & 0.4 & -0.6 \end{pmatrix}$$

le diagramme de transition de cette CMTC est illustré dans la figure 2.2 suivante :

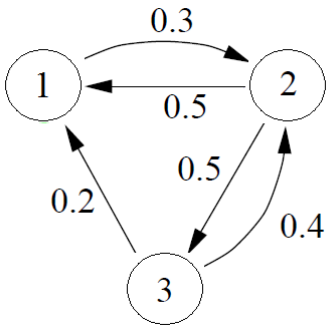

FIG. 2.2 – Diagramme de transition de la CMTC.

2.2.5.2 Étude des chaînes de Markov à temps continu

Il existe une relation étroite entre les CMTC et les CMTD. En effet, pour chaque CMTC définie par son générateur infinitésimal $Q = \|q_{ij}\|$, on peut définir une CMTD dite *chaîne de Markov incluse* (*CMI* en abrégé). Cette dernière est définie par sa matrice de probabilités de transitions $P = \|p_{ij}\|$, les termes de la matrice P sont obtenus comme suit :

On sait que :

$$q_{ij} = \mu_{ij} = \mu_i.p_{ij}$$

On a donc :

$$p_{ij} = \frac{\mu_{ij}}{\mu_i} = \frac{\mu_{ij}}{\sum_{k \neq i} \mu_{ik}} = -\frac{q_{ij}}{q_{ii}}$$

Grâce à ce résultat [71], l'étude de chaînes de Markov à temps continu est énormément facilitée. En effet, pour garantir l'existence d'un régime stationnaire, la CMTC doit être irréductible, on a les résultats suivants [14] :

Résultat 1. Une CMTC est irréductible si et seulement si sa CMTD incluse est irréductible.

Résultat 2. Une CMTC finie et irréductible est *ergodique*.

Résultat 3. Par conséquent, une CMTC finie et irréductible, tend vers une distribution stable π après l'écoulement d'un temps infini. Le vecteur π est l'unique solution du système d'équations matricielles suivant :

$$\begin{cases} \pi.Q = 0 \\ \sum_{i \in E} \pi_i = 1 \end{cases}$$

Où : E est l'espace des états de la chaîne de Markov.

Processus semi-markoviens Les processus *semi-markoviens* sont une extension des CMTC, où le temps de séjour dans les états de la chaîne sont distribués arbi-

trairement [29].

2.3 Les réseaux de Petri

2.3.1 Définition et Concepts de base

Un réseau de Petri (*RdP* en abrégé) est un *graphe biparti orienté* composé de deux types de nœuds

- *Les places,* qui permettent de décrire les états du système modélisé ;
- *Les transitions,* qui représentent les changements d'état. Places et transitions sont reliées par des arcs orientés (figure 2.3).

Les arcs sont représentés par des flèches allant d'une place à une transition et vice versa (notons qu'un arc ne peut jamais relier deux sommets de même nature). À chaque arc est associé un nombre entier positif appelé *poids de l'arc*. Par défaut, le poids d'un arc k est de 1, mais lorsqu'il prend une autre valeur, celle-ci doit être indiquée sur l'arc correspondant. Sémantiquement, le poids d'un arc représente :

- le nombre de ressources nécessaires pour déclencher un évènement, dans le cas où l'arc relie une place à la transition t, et,
- le nombre de ressources libérées suite à l'occurrence de l'évènement représenté par la transition t, dans le cas où l'arc relie cette dernière à une place.

Une place peut contenir un nombre entier de *jetons* ou *marques*. Dans le cas où la place représente une condition logique (machine en marche ou en panne, imprimante à l'état prêt ou non, etc.), la présence d'une marque indique que cette condition est vraie, et fausse dans le cas contraire, et si la place représente une ressource au sens le plus large (serveurs à l'état libre, par exemple), le nombre de jetons indique le nombre de ces ressources. L'ensemble des marques présentes à un instant donné dans les places, constitue le *marquage* du réseau à cet instant.

Le marquage dit *initial* d'un réseau de Petri, indique l'état initial du système modélisé, en décrivant la distribution initiale des marques dans les places.

Définition formelle

L'approche informelle que nous venons de citer nous a permis de nous familiariser avec les concepts des réseaux de Petri et leur structure. Nous allons maintenant donner une définition formelle d'un réseau de Petri [57, 29].

Un réseau de Petri est un quadruplet $R = (P, T, Pré, Post)$, tel que :

- $P = \{p, p_2, p_3, \ldots, p_n\}$, est un ensemble de places, fini, $\mid P \mid = n$;
- $T = \{t_1, t_2, t_3, \ldots, t_m\}$, est l'ensemble des transitions, $\mid T \mid = m$;
- $Pré : P \times T \rightarrow \mathbb{N}$, est l'application d'incidence avant ;
- $Post : P \times T \rightarrow \mathbb{N}$, est l'application d'incidence arrière.

(où $\mathbb{N}$ est l'ensemble des entiers naturels)

$Pré(p_i, t_j)$ est le poids k de l'arc reliant la place p_i à la transition t_j.

$$Pré(p_i, t_j) = \begin{cases} k & \text{si l'arc } (p_i, t_j)\text{existe.} \\ 0 & \text{sinon.} \end{cases}$$

$Post(p_i, t_j)$ est le poids k de l'arc reliant la transition t_j à la place p_i.

$$Post(p_i, t_j) = \begin{cases} k & \text{si l'arc}(t_j, p_i) \text{ existe.} \\ 0 & \text{sinon.} \end{cases}$$

Un réseau de Petri marqué est défini par un couple $R_m = (R, M)$, tel que [57, 29] :

- R est le réseau de Petri défini précédemment ;
- $M : P \rightarrow \mathbb{N}$, est l'application marquage du réseau, telle que $M(p_i)$ est le nombre de jetons dans la place p_i.

Si $\mid P \mid = n$, alors $M(P)$ est un vecteur à n composantes. Le marquage initial est noté par M_0.

Notation matricielle [57]

Les fonctions $Pré$ et $Post$ sont représentées par des matrices W^-, W^+ à n lignes (nombre de places), m colonnes (nombre de transitions) respectivement.

- La matrice $W^- = [w_{ij}^-]$, où $w_{ij}^- = Pré(p_i, t_j)$, est appelée *matrice d'incidence avant.*

– La matrice $W^+ = [w_{ij}^+]$, où $w_{ij}^+ = Post(p_i, t_j)$, est appelée *matrice d'incidence arrière.*

– La matrice $W = W^+ - W^- = [w_{ij}]$, est appelée *matrice d'incidence.*

Exemple

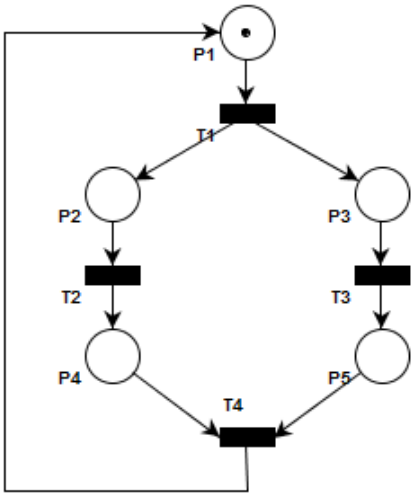

FIG. 2.3 – Exemple d'un réseau de Petri.

La figure 2.3 représente un réseau de Petri où

$P = \{p_1, p_2, p_3, p_4, p_5\}$.

$T = \{t_1, t_2, t_3, t_4\}$.

On a

$$W^- = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \; W^+ = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Par conséquent

$$W = \begin{pmatrix} -1 & 0 & 0 & 1 \\ 1 & -1 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & -1 \end{pmatrix}$$

Le marquage initial de ce RdP est $M_0 = (1, 0, 0, 0, 0)$.

2.3.2 La dynamique d'un réseau de Petri

L'évolution dynamique d'un réseau de Petri consiste à passer d'un marquage à un autre suite au *franchissement* (*tir*) d'une transition qui donne lieu à de nouveaux jetons et fait disparaître d'autres. Dans ce qui suit, nous allons étudier les conditions de franchissement d'une transition ainsi que l'ensemble des marquages accessibles à partir d'un marquage initial M_0.

Nous avons les notations suivantes [57] :

- $^\circ t = \{p \in P | Pré(p,t) > 0\}$ =ensemble des places d'entrée de t ;
- $t^\circ = \{p \in P | Post(p,t) > 0\}$ =ensemble des places de sortie de t ;
- $^\circ p = \{t \in T | Post(p,t) > 0\}$ =ensemble des transitions d'entrée de p ;
- $p^\circ = \{t \in T | Pré(p,t) > 0\}$ =ensemble des transitions de sortie de p.

2.3.2.1 Sensibilisation d'une transition

Une transition t est dite *sensibilisée* (*franchissable*, *tirable* ou encore *validée*) pour un marquage M, si et seulement si le nombre de jetons dans chacune des places en entrée est supérieur à la pondération des arcs reliant chaque place à cette transition, autrement dit [57] :

$$t \text{ est franchissable pour } M \Longleftrightarrow \forall p \in {}^\circ t : M(p) \geq Pré(p,t)$$

Dans l'exemple précédent, t_1 est franchissable pour le marquage M_0. t_2, t_3, t_4 et t_5 ne le sont pas.

2.3.2.2 Franchissement d'une transition

Le *franchissement* ou le *tir* d'une transition t consiste à enlever de chacune des places p en entrée de t le nombre de jetons indiqués sur l'arc entrant à t (i.e. $Pré(p,t)$), et à déposer dans chacune des places p en sortie un nombre de jetons égal au poids de l'arc reliant t à p (i.e. $Post(p,t)$). Le tir d'une transition est supposé être une opération instantanée et indivisible.

Étant donné un marquage actuel M, le franchissement d'une transition t sensibilisée donne naissance à un autre marquage M' défini par [57] :

$$\forall p \in P : M'(p) = M(p) + Post(p,t) - Pré(p,t)$$

On dit aussi que M' est *accessible* à partir de M et on note : $M[t\rangle M'$.

Une transition source est une transition qui n'a aucune place en entrée. Une telle transition est toujours franchissable et son franchissement est déclenché quand l'évènement correspondant se produit.

Une transition puits est une transition qui n'a aucune place en sortie. Lorsque l'évènement correspondant à une telle transition se produit, le tir a lieu en enlevant des marques de toutes les places en entrée de cette transition.

Le franchissement des transitions et le changement de marquages qu'il entraîne, permettent d'analyser la dynamique du système modélisé.

2.3.2.3 Séquence de franchissement et marquages accessibles [65]

Une séquence de franchissement à partir d'un marquage M_1 est représentée par la suite des transitions $S = t_1 t_2 \ldots . t_i \ldots . t_k$ telle que le franchissement de chacune d'elles conduit à un marquage qui sensibilise la suivante. Autrement dit :

$$\exists (k+1) \text{ marquages } M_1, M_2, ..., M_{k+1} : \forall i \in \{1, ..., k\}\ M_i[t_i\rangle M_{i+1}$$

Les marquages $M_2, ..., M_{k+1}$ sont dits *accessibles* à partir de M_1.

L'ensemble de tous les marquages accessibles à partir d'un marquage initial M_0 est appelé *ensemble d'accessibilité*, il est noté $A(R, M_0)$. On a donc :

$$A(R, M_0) = \{M \in \mathbb{N}^p | \exists S \in T^* : M_0[S\rangle M\}$$

2.3.2.4 Vecteur caractéristique et équation d'états [65, 57]

On appelle *vecteur caractéristique associé à une séquence de franchissements* S le vecteur $\bar{S}$ à m éléments, où $m = \mid T \mid$, dont les composantes sont le nombre d'occurrences de chaque transition t dans la séquence S.

Le passage d'un marquage M_{k-1} à un marquage M_k s'écrit sous la forme :

$$M_k = M_{k-1} + W.U_k$$

où U_k est un vecteur binaire tel que :

$$U_k(i) = \begin{cases} 1 & \text{si la transition } t_i\text{est la transition franchie;} \\ 0 & \text{sinon.} \end{cases}$$

On démontre que, si S est une séquence de franchissements, le marquage M atteint après le franchissement de toute la séquence S est donné par :

$$M = M_0 + W.\bar{S}$$

Cette équation s'appelle *équation d'états* du réseau de Petri correspondant [65].

2.3.2.5 Graphe des marquages accessibles [65]

Étant donné un réseau de Petri R ayant un marquage initial M_0, le *graphe d'accessibilité* (ou *graphe de marquages accessibles*, ou encore *graphe d'états*) noté *GMA* est un graphe orienté pondéré dont les nœuds sont les marquages accessibles (les éléments de $A(R, M_0)$) et les arcs correspondent aux franchissements des transitions, c'est-à-dire qu'un nœud m_i est relié à m_j par un arc de poids t si m_j est directement accessible à partir de m_i en franchissant t. $(m_i[t\rangle m_j)$.

Exemple

La figure 2.4 illustre le graphe d'accessibilité de l'exemple considéré dans la figure 2.3 précédente, et dont le vecteur caractéristique est $(1, 2, 2, 1)$.

2.3.3 Propriétés des réseaux de Petri

L'objectif de la modélisation par réseau de Petri est de permettre l'analyse des propriétés qualitatives du système modélisé. Pour cela, l'étude se fait sur le modèle du réseau de Petri correspondant. Parmi ces propriétés, nous citerons celles

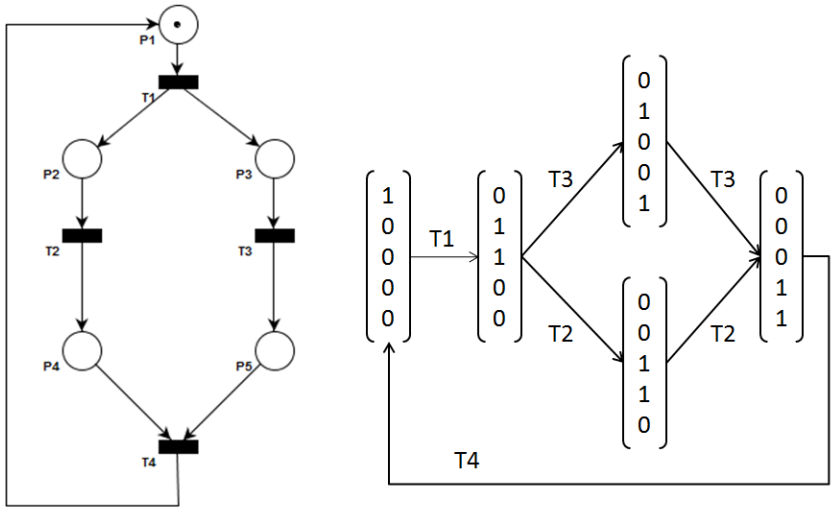

FIG. 2.4 – Graphe des marquages accessibles de RdP.

qui permettent d'affirmer que les spécifications incluses dans le modèle réseau de Petri sont correctes. C'est ainsi que nous pourrons démontrer qu'un réseau (et par conséquent le système modélisé) est sans blocage ou que le nombre d'états pouvant être atteints est fini. On mettra également en évidence les conflits entre plusieurs évolutions possibles.

2.3.3.1 Bornitude

Une place p_i est dite *bornée* pour un marquage initial M_0 s'il existe un entier naturel k, tel que pour tout marquage accessible à partir de M_0, le nombre de jetons dans p_i est inférieur ou égal à k. On dit que p_i est $k-borné$ [57].

Le réseau de Petri R est *borné* pour le marquage initial M_0, si toutes ses places le sont. Autrement dit :

$$(R, M_0) \text{ borné } \iff \exists k \in \mathbb{N}, \forall M \in A(R, M_0), \forall p \in P : M(p) \leq k$$

On dira également que le nombre de marquages accessibles à partir de l'état initial est *fini*, le graphe d'accessibilité équivalent peut donc être construit.

Dans le cas particulier où $k = 1$, le réseau de Petri correspondant est dit *sauf* ou *binaire*.

Exemple Le réseau de Petri de la figure 2.5 suivante n'est pas borné, car le marquage de p_1 ne cesse pas d'augmenter avec l'évolution du réseau.

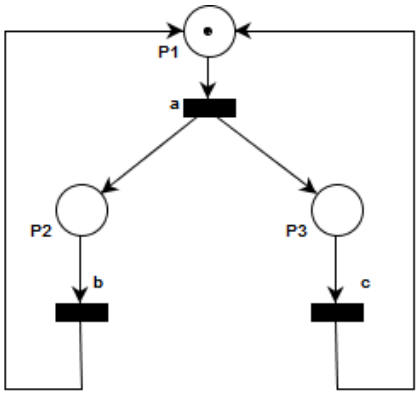

FIG. 2.5 – Exemple d'un RdP non borné.

2.3.3.2 Non blocage

Un *blocage* (ou *état puits*, ou encore *marquage mort*) est un marquage pour lequel aucune transition n'est franchissable [57]. Un réseau de Petri marqué est dit *sans blocage* pour un marquage initial M_0 si aucun marquage accessible n'est un marquage mort.

2.3.3.3 Vivacité [65]

Soit R un réseau de Petri. Une transition t est *quasi-vivante* pour un marquage initial M_0, si elle est franchissable au moins une fois à partir d'un marquage accessible de M_0.

$$t \text{ quasi-vivante pour } M_0 \Longleftrightarrow \exists M \in A(R, M_0) : M[t\rangle$$

R est *quasi-vivant* si toutes ses transitions le sont, i.e.

$$R \text{ est quasi-vivant} \Longleftrightarrow \forall t \in T, \exists M \in A(R, M_0) : M[t\rangle$$

Un réseau de Petri est dit *pseudo-vivant* pour un marquage initial M_0 si tout marquage accessible à partir de M_0 admet au moins une transition franchissable,

ceci revient à dire que le graphe d'accessibilité n'admet aucun marquage mort, ou encore que R est sans blocage. Plus formellement

$$R \text{ est pseudo-vivant} \Longleftrightarrow \forall M \in A(R, M_0), \exists t \in T : M[t\rangle$$

Une transition t est *vivante* pour un marquage initial M_0, si pour tout marquage M_i accessible à partir de M_0, il existe une séquence de franchissement S qui contient la transition t, autrement dit, quelque soit l'évolution, il existera toujours une possibilité de franchir t.

Un réseau de Petri est vivant si toutes ses transitions sont vivantes, c'est-à-dire qu'à partir de tout marquage accessible du marquage initial, toute transition a la possibilité d'être franchie. De façon plus formelle :

$$R \text{ vivant} \Longleftrightarrow \forall M \in A(R, M_0), \forall t \in T, \exists M' \in A(R, M) : M'[t\rangle$$

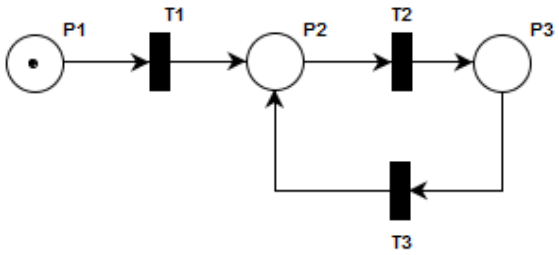

FIG. 2.6 – Exemple d'un RdP non vivant.

Exemple Le réseau de Petri de la figure 2.6 suivante n'est pas vivant car la transition t_1 n'a pas la possibilité d'être franchie dès qu'on passe au marquage $(0, 1, 0)$. Par contre, il est quasi vivant (toutes les transitions sont franchissables au moins une fois) et pseudo vivant (absence de marquage puits).

La propriété de vivacité a un intérêt majeur dans l'étude des systèmes séquentiels car elle est directement liée aux situations de non blocage, dans ce sens qu'un réseau de Petri vivant garantit qu'aucun blocage ne se produit pendant son évolution, il garantit également l'absence de parties mortes i.e. des configurations jamais

atteintes.

2.3.3.4 Réinitiabilité

On dit qu'un réseau de Petri R possède *un état d'accueil* M_a, si M_a est accessible de tous les marquages accessibles du marquage initial M_0.

$$M_a \text{ est un état d'accueil } \Longleftrightarrow \forall M \in A(R, M_0), \exists S \in T^* : M[S\rangle M_a$$

Le réseau de Petri est dit *réinitialisable* (*propre*) s'il admet M_0 comme état d'accueil [57].

Exemple Il est clair que le réseau de Petri de la figure 2.6 admet P_2 et P_3 comme états d'accueil, cependant il n'est pas réinitialisable car on ne peut jamais revenir au marquage initial.

L'intérêt pratique des réseaux de Petri réinitialisables est qu'ils peuvent se remettre dans leur état initial eux même, ceci est intéressant dans la reprise automatique après une panne ou une erreur. Contrairement aux réseaux non réinitialisables où une réinitialisation manuelle est nécessaire.

2.3.3.5 Persistance

On dira que deux transitions sont en *conflit structurel* si elles ont au moins une place commune en entrée. Le conflit structurel ne dépend pas du marquage. En présence de marquage, le franchissement d'une transition en conflit structurel peut empêcher le franchissement de l'autre, on parle dans ce cas de *conflit effectif* [57].

Dans un réseau de Petri à conflit effectif, il est nécessaire de faire un choix de la transition qui va être franchie.

Un conflit effectif signifie qu'il y a un non-déterminisme du réseau, donc que l'évolution du système décrit présente une partie aléatoire.

On dit qu'un réseau de Petri est *simple* si toute transition ne peut être concernée que par un conflit au plus.

Exemple Dans la figure 2.7-(a), t_1 et t_2 sont en conflit structurel. Quand les deux places en entrée sont marquées, on parle d'un conflit effectif, 2.7-(b). Par contre, dans la partie 2.7-(c) on n'a pas de conflit effectif car la transition t_2 n'est même pas sensibilisée.

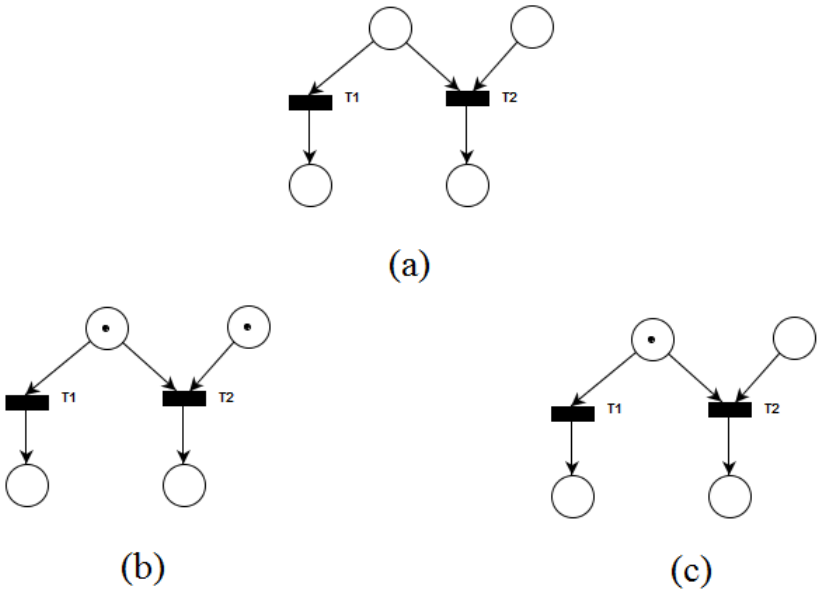

FIG. 2.7 – Exemple d'un conflit structurel.

Un RdP est dit *persistant* pour un marquage initial M_0 si pour tout marquage M_i accessible à partir de M_0, si deux transitions t_j et t_k sont franchissables alors $t_j t_k$ et $t_k t_j$ sont des séquences de franchissement à partir de M_i. Lors d'un conflit effectif dans un réseau de Petri persistant, il n'est pas nécessaire de faire le choix de la transition à franchir car l'ordre n'est pas important.

2.3.3.6 Exclusion mutuelle

Soit R un réseau de Petri, deux places sont en *exclusion mutuelle* ou *mutuellement exclusives* si pour un marquage initial M_0 donné, elles ne peuvent être simultanément marquées quelque soit le marquage M atteint à partir de M_0 [65].

$$p_i, p_j \text{ sont en exclusion mutuelle} \Longleftrightarrow \forall M \in A(R, M_0) : M(p_i).M(p_j) = 0$$

Exemple Les places p_1 et p_2 de la figure 2.8 sont en exclusion mutuelle.

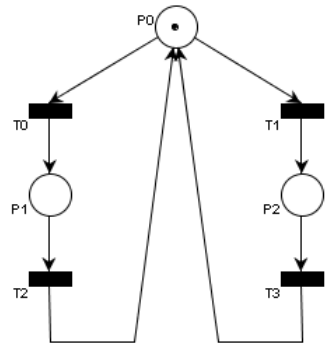

FIG. 2.8 – Exclusion mutuelle entre les places.

On rencontre l'exclusion mutuelle dans tout système comprenant un partage de ressource.

2.3.4 Réseaux de Petri à arcs inhibiteurs

Les réseaux de Petri ordinaires tels qu'ils sont définis s'avèrent, dans certains cas, limités et incapables de modéliser certaines contraintes. Ainsi, d'autres classes de réseaux de Petri ont été définies dans la littérature pour améliorer la capacité des réseaux de Petri afin de modéliser des systèmes plus complexes.

Par exemple, dans un RdP ordinaire, une transition est validée lorsque chacune de ses places d'entrée contient au moins un jeton. Cette règle ne permet pas de réaliser *un test à zéro*, autrement dit, de conditionner le franchissement d'une transition à l'état vide de l'une de ses places d'entrée. Les réseaux de Petri *à arcs inhibiteurs* sont une extension intéressante qui permet de palier à ce problème.

2.3.4.1 Définition [29]

Un réseau de Petri *à arcs inhibiteurs* est un doublet $< R, Inh >$ tel que :

- R est un réseau de Petri ;
- $Inh : P \times T \longrightarrow \mathbb{N}\backslash\{0\}$ est *la fonction d'inhibition.*

Un arc inhibiteur est un arc allant d'une place à une transition (et non pas l'inverse). Graphiquement, il est représenté par une flèche avec un petit cercle à l'extrémité.

La fonction d'inhibition peut être représentée par une matrice à n lignes, m colonnes, où $n = |P|$, $m = |T|$, pour qu'une transition t soit franchissable, la valeur

de marquage de chaque place p en entrée doit être strictement inférieure à $Inh(p,t)$.

2.3.4.2 Franchissement dans un réseau à arcs inhibiteurs [29]

Soit (R, Inh, M), un réseau de Petri à arcs inhibiteurs marqué, où M est son marquage,

$$t \in T \text{ est franchissable } \iff \forall p \in P : M(p) \geq Pré(p,t) \text{ et } M(p) < Inh(p,t).$$

Ainsi t n'est franchissable que si le marquage de chacune des places en entrée est inférieur à la valuation de l'arc inhibiteur reliant cette place à cette transition.

Le franchissement de la transition t à partir de M donne naissance au marquage M' défini comme suit :

$$\forall p \in P : M'(p) = M(p) + Post(p,t) – Pré(p,t)$$

Exemple

La figure 2.9 montre un exemple d'une transition à arc inhibiteur franchissable, et une autre non franchissable.

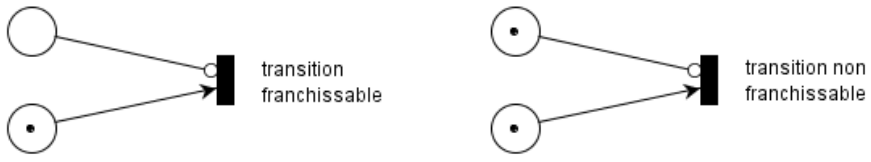

FIG. 2.9 – Exemple de franchissement d'un arc inhibiteur.

2.3.5 Analyse des réseaux de Petri

Avant de passer à l'étape de mise en œuvre, le système ne doit comporter aucune erreur, aucune ambiguïté dans la spécification des fonctions qui le composent. La

détection tardive des erreurs de conception peut coûter très chère. C'est pourquoi toute modélisation doit être suivie d'une étape de validation du modèle de réseau de Petri, qui permet de donner une information précise sur les propriétés du système, en étudiant les propriétés du réseau de Petri équivalent. Plusieurs approches d'analyse existent dans la littérature, à chacune ses avantages et ses inconvénients. Généralement, une méthode d'analyse permet de vérifier les propriétés des réseaux de Petri, et conduit à la définition d'un algorithme de validation. Dans ce paragraphe, nous allons parler de quelques unes de ces méthodes en montrant leurs avantages et inconvénients [65].

2.3.5.1 Analyse par énumération

Elle consiste à construire le graphe des marquages accessibles (*graphe de couverture*). Si celui-ci est fini, la vérification de certaines propriétés, deviendra facile. Le point faible de cette méthode réside dans l'explosion du nombre d'états du graphe de couverture, même si le nombre de places et de transitions reste faible dans le RdP équivalent, ce qui la rend très coûteuse en termes de temps d'exécution et d'espace mémoire et par conséquent très difficile à mettre en œuvre.

2.3.5.2 Analyse par réduction

Pour pallier aux inconvénients de la méthode précédente, l'idée de cette approche est de faire réduire le réseau de Petri de façon à simplifier la construction du graphe de couverture. Cependant, l'application des règles de réduction ne conserve pas toujours la signification physique du réseau. Dans le cas où le réseau analysé ne possède pas les bonnes propriétés, il est difficile de remonter à cause de l'erreur.

2.3.5.3 Analyse structurelle

C'est l'approche la plus efficace qui existe à nos jours. Elle est directement liée à la structure du réseau indépendamment du marquage initial et elle n'a pas recours à tracer le graphe de couverture. Rappelons que chaque réseau de Petri est caractérisé par une équation d'états qui permet de déduire un marquage M' à partir d'un autre

marquage M après une séquence de franchissement S comme suit :

$$M' = M + W.\bar{S}$$

tel que W est la matrice d'incidence, $\bar{S}$ est le vecteur caractéristique associé à S.

Considérons le système d'équations [65] :

$$W^T.X = 0$$

Avec X vecteur d'entiers relatifs de dimension $n =\mid P \mid$

W^Tmatrice transposée de W

et soit $X = V$ une solution de ce système

On montre, à partir de l'équation d'états que le tir de toute séquence de franchissements à partir d'un marquage M maintient la forme linéaire $V^T.M$ constante pour tous les marquages accessibles, en particulier :

$$V^T.M = V^T.M_0$$

$V^T.M = \sum_{i=1}^{n} v_i.M(p_i)$ est appelé *invariant linéaire de places.*

Les invariants permettent de mettre en évidence les différentes propriétés de base et les propriétés spécifiques du RdP, tout en évitant d'énumérer tous les marquages accessibles, car la résolution du système d'équations se ramène à un problème de programmation linéaire en nombres entiers.

Si les composantes v_i du vecteur V sont toutes positives, alors les places intervenant dans l'invariant linéaire $V^T \cdot M = V^T \cdot M_0$ sont toutes bornées. En effet :

$$V^T.M = \sum_{i=1}^{n} v_i.M(p_i) = \sum_{i=1}^{n} v_i.M_0(p_i)$$

Donc :

$$M(p_i) \leq \left| \frac{\sum_{i=1}^{n} v_i.M_0(p_i)}{\sum_{i=1}^{n} v_i} \right|$$

Si chaque place du réseau apparaît dans au moins un invariant, et si toutes les composantes du vecteur V sont positives, alors le réseau est borné [65].

La modélisation par réseaux de Petri a l'avantage d'être simple et spontanée, elle offre en outre la possibilité d'effectuer une analyse *qualitative* des propriétés logiques des systèmes, comme le blocage, la relation entre les évènements (causalité, exclusion mutuelle, concurrence...) et la bornitude des états des systèmes, etc. Ces propriétés sont fondamentales dans le fonctionnement d'un système. Cependant, les méthodes de vérification ne sont pas toujours efficaces, de plus, les propriétés qualitatives seules ne sont pas suffisantes pour garantir le bon fonctionnement d'un système. Modéliser un système doit aussi pouvoir vérifier les paramètres *quantitatifs* tels que le temps moyen d'exécution d'une tâche dans un système, le taux de perte de paquets dans un réseau, etc. Ainsi, d'autres structures de réseaux de Petri prenant en compte la notion de temps sont apparues ; les réseaux de Petri *stochastiques* (*RdPS*), et réseaux de Petri *stochastiques généralisés* (*RdPSG*), que nous allons présenter dans la section suivante.

2.4 Réseaux de Petri stochastiques généralisés

Plusieurs chercheurs ont proposé d'enrichir les modèles de réseaux de Petri pour que l'analyse *quantitative* des systèmes modélisés soit possible. L'introduction des spécifications temporelles a eu lieu avec différentes approches... En particulier, *les réseaux de Petri stochastiques* [74] sont des réseaux de Petri où, à chaque transition est associée une variable aléatoire modélisant le délai de franchissement de cette transition i.e. l'intervalle de temps qui sépare l'instant de déclenchement de l'instant de la fin de l'évènement représenté par la transition.

Le problème principal des RdPS vient du fait que, les durées d'exécution de toutes les actions dans les systèmes modélisés, sont associées à un temps aléatoire, or, ceci n'est pas toujours correct ; c'est le cas par exemple des opérations de synchronisation, des opérations d'allocation des ressources, ou bien des actions purement logiques dans les systèmes informatiques. En effet, les transactions associées à ce genre d'actions doivent être immédiates.

Les modèles RdPS dans lesquels les actions logiques sont représentées par des transitions dont la durée de franchissement est nulle sont connus sous le nom *RdPS généralisés, RdPSG* [71]. Les RdPSG combinent deux types de transitions ; les transitions *immédiates* qui sont représentées par des rectangles noirs, et celles *temporisées*, représentées pas des rectangles vides.

- *Les transitions temporisées* sont des transitions dont les délais de franchissement sont associés à des variables aléatoires déterminant la durée d'exécution des différentes activités. Quand ces délais de franchissement sont des variables aléatoires à distributions exponentielles négatives, on parle de *RdPSG markoviens.*
- *Les transitions immédiates (instantanées)*, se caractérisent par un délai de franchissement nul, permettant ainsi de représenter les actions prioritaires qui ne consomment aucun temps, comme la synchronisation, les opérations logiques, les évènements d'urgence ou les activités prioritaires. Il est à noter que ces transitions sont plus prioritaires que les transitions temporisées. De plus, plusieurs niveaux de priorités peuvent être définis entres les transitions immédiates en définissant un *poids* pour chaque transition.

Nous nous intéressons dans ce mémoire à la classe des RdPSG *markoviens.* Ainsi, dans ce qui suit, le mot RdPSG désignera implicitement un RdPSG markovien.

2.4.1 Définition

Un Réseau de Petri Stochastique Généralisé, RdPSG, est défini formellement par un 8-uplet $< P, T, Pré, Post, Inh, pri, W, M_0 >$ tel que [42] :

- P est l'ensemble des *places* ;
- T est l'ensemble des *transitions* (immédiates et temporisées) ;
- $Pré : P \times T \longrightarrow \mathbb{N}$, est la fonction d'*incidence avant* ;
- $Post : P \times T \longrightarrow \mathbb{N}$, la fonction d'*incidence arrière* ;
- $Inh : P \times T \longrightarrow \mathbb{N}$, la fonction d'*inhibition* ;
- $pri : T \longrightarrow \{0, 1\}$, est la fonction de *priorité*, elle associe à chaque transition temporisée la valeur 0 et à chaque transition immédiate la valeur 1 (la valeur 1 est plus prioritaire que la valeur 0) ;

- $W : T \longrightarrow \mathbb{R}^+$, la fonction qui associe à chaque transition temporisée *un délai de franchissement*, et à chaque transition immédiate *un poids*. Les poids sont utilisés dans le calcul des probabilités de franchissement des transitions immédiates et pour la résolution des conflits entre plusieurs transitions immédiates ;
- $M_0 : P \longrightarrow \mathbb{N}$, est le marquage initial du réseau.

2.4.2 Processus Stochastique associé à un RdPSG

À cause de la présence de transitions immédiates, l'ensemble des marquages accessibles d'un RdPSG contient deux types de marquages [71, 13] :

- Les marquages *tangibles* ; dans lesquels aucune transition immédiate n'est sensibilisée,
- Les marquages *évanescents* ; où il y a au moins une transition immédiate franchissable.

Les marquages tangibles représentent les états où le système modélisé passe un certain temps, les marquages évanescents, cependant, modélisent les états dans lesquels le temps passé est nul. Le processus stochastique associé à un RdPSG est un *processus stochastique semi-markovien*, où la distribution de temps de séjour dans les marquages est une composition de distributions exponentielles négatives et distributions déterministes nulles.

Le temps moyen de séjour dans un marquage évanescent est nul, tandis que, le temps de séjour dans un marquage tangible M est une variable aléatoire correspondant au minimum des temps de franchissement des transitions sensibilisées par ce marquage, autrement dit, c'est une loi exponentielle avec un paramètre λ_M qui est la somme de tous les taux de franchissement de ces transitions [71, 13] :

$$\lambda_M = \sum_{t_k \in S(M)} w(t_k)$$

où $S(M)$ est l'ensemble des transitions franchissables à partir de M.

Par conséquent, le temps moyen de séjour dans ce marquage est donné par :

$$TS_M = \frac{1}{\lambda_M} = \frac{1}{\sum_{t_k \in S(M)} w(t_k)}$$

2.4.3 La dynamique des RdPSG

Tout comme les RdP ordinaires, l'évolution d'un RdPSG se fait par une suite successive de marquages. La fonction de distribution exponentielle négative assure que le développement du processus de marquage n'est plus conditionné par rapport à son passé, le changement de marquage ne dépend que du marquage actuel, les travaux déjà réalisés par les activités interrompues sont perdus. Le seul travail terminé est celui de l'activité qui correspond à la transition qui a provoqué le changement de l'état du système. On appelle cette politique *Resampling* (*interruption*) [71, 13].

Lorsqu'un marquage M est atteint, on distingue deux scénarios pour passer à un autre marquage selon que ce M soit tangible ou évanescent. Soit $S(M)$ l'ensemble de transitions sensibilisées de ce marquage ;

- Si $S(M)$ ne contient que des transitions temporisées (marquage tangible), tous les évènements associés aux transitions sensibilisées commencent à s'exécuter en parallèle, cependant, le changement de l'état du réseau est provoqué par le déclenchement de la transition ayant le plus petit délai de franchissement. Cette politique modélise ce que l'on appelle *modèle concurrentiel*. La probabilité qu'une transition $t_j \in S(M)$ ait le plus petit délai de franchissement est donnée par la formule suivante :

  $$P[t_j \mid M] = \frac{w(t_j)}{\sum_{t_k \in S(M)} w(t_k)}$$

  Il existe une autre politique de franchissement appelée *politique de pré-sélection*, où la transition tirée est choisie selon une autre variable aléatoire indépendamment du délai de franchissement. Toutefois, la politique concurrentielle demeure la plus utilisée en pratique.
- Si M est un marquage évanescent, i.e. $S(M)$ comprend au moins une transi-

tion immédiate, seulement les transitions immédiates ont la possibilité d'être tirées car elles sont plus propriétaires que les transitions temporisées, vu leur temps de franchissement nul. Laquelle des transitions immédiates sera tirée si on en a plusieurs, ce problème ne peut se poser qu'en cas de transitions en conflit effectif, les transitions concurrentes pouvant être simultanément tirées. Si $S(M)$ contient plusieurs transitions en conflit, une seule transition pourra être tirée avec une certaine probabilité qui dépend du poids de chaque transition en conflit, soit $C(M) \subset S(M)$ l'ensemble des transitions immédiates en conflit entre elles, la probabilité qu'une transition $t_j \in C(M)$ soit tirée est donnée par :

$$P[t_j \mid M] = \frac{w(t_j)}{\sum_{t_k \in C(M)} w(t_k)}.$$

Le nouveau marquage M' résultant après le franchissement d'une transition t_j (temporisée ou immédiate) à partir d'un marquage M est défini alors comme suit : $M' = M - Pré(., t_j) + Post(., t_j)$.

Cette sémantique de transition que nous venons de citer concerne la sémantique de *serveur fini*. Dans la sémantique de serveur infini, la même transition est tirée simultanément plusieurs fois [47]. La multiplicité de franchissement dépend du nombre de marquage dans chaque place en entrée. On appelle *degré de franchissement* (*enabling degree*) d'une transition t_j dans le marquage M et on note $ED(t_j, M)$ la quantité $ED(t_j, M) = \min_{p_i \in t_j}\{M(p_i)/Pré(p_i, t_j)\}$, le taux de franchissement devient dépendant du marquage et est donné par $w(t_j).ED(t_j, M)$.

2.4.4 Résolution numérique des RdPSG

Le processus stochastique associé à un RdPSG k-borné, ayant M_0 comme marquage initial, est un processus *semi-markovien* à temps continu, irréductible, homogène, et à espace d'états fini [71, 13].

Les processus semi-markoviens peuvent être analysés en identifiant une *chaîne de Markov à temps discret incluse* CMTDI, qui décrit les transitions entre les états (marquages). Dans le cas des RdPSG, la chaîne de Markov induite CMI peut être reconnue en se focalisant sur les états du processus semi-markovien en oubliant le

concept de temps. Les spécifications du RdPSG sont suffisantes pour calculer les probabilités de transition de la chaîne.

La matrice de transition U de la CMI est obtenue en utilisant l'expression suivante :

$$u_{ij} = \frac{\sum_{t_k \in E_j(M_i)} w(t_k)}{\lambda_i}$$

Ainsi, toutes les probabilités de transitions de la CMI sont calculables indépendamment de la nature (temporisée, immédiate) de la transition qui a causé le changement d'état.

Ordonnons les marquages de telle façon que les marquages évanescents correspondent aux premières entrées de la matrice, et les marquages tangibles aux dernières, la matrice de probabilité U peut être décomposée comme suit [71, 13] :

$$U = A + B = \begin{pmatrix} C & D \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ E & F \end{pmatrix}$$

La matrice A décomposée, à son tour, en deux sous-matrices C et D, où C contient les probabilités de passage de marquages évanescents vers des marquages évanescents, D celle des transitions de marquages évanescents à tangibles. D'une manière similaire, B est composée des sous-blocs E et F, tel que E correspond aux transitions tangibles-évanescent et F tangibles-tangibles.

Pour calculer les probabilités stationnaires, une première approche consiste à utiliser la technique classique des CMTD, i.e. en résolvant le système d'équations suivant :

$$\begin{cases} \pi = \pi.U \\ \sum \pi_i = 1 \end{cases}$$

On sait déjà à l'avance que les probabilités stationnaires des marquages évanescents sont nulles, car le temps moyen de séjour est nul. Le point faible de cette approche réside non seulement dans la redondance de calcul de ces probabilités mais aussi dans l'augmentation de la taille de la matrice U quand le nombre des marquages évanescents croît, ce qui rend la solution très coûteuse en termes de temps de calcul

et d'espace mémoire.

Cette méthode peut être optimisée pour restreindre le calcul aux marquages tangibles uniquement. En effet, la CMI précédente doit être réduite pour ne contenir que des marquages tangibles. La chaîne résultant est appelée *chaîne de Markov incluse réduite CMIR*. Notons que cette opération de réduction n'a aucune influence sur le comportement dynamique du processus, tant que ce dernier passe un temps nul dans les marquages éliminés (évanescents).

Il a été démontré [70,69] que la fonction des probabilités U' de la nouvelle chaîne de Markov CMIR est obtenue de la manière suivante :

$$U' = F + E.H$$

telle que :

$$H = \begin{cases} \left(\sum_{k=0}^{n_0} C^k\right).D & \text{Pas de boucles entre marquages évanescents,} \\ [I - C]^{-1}.D & \text{il existe des boucles entre les marquages évanescents.} \end{cases}$$

Une fois la matrice des probabilités de transitons est construite, la technique standard est utilisée pour calculer les probabilités stationnaires de la CMIR et par conséquent, celles des marquages tangibles. Le calcul des paramètres des performances définis sur les RdPSG pourra aussi être fait.

2.5 Conclusion

Réputés très puissants dans les laboratoires de recherche, ainsi que dans le monde industriel, les réseaux de Petri sont un outil de modélisation graphique et mathématique, utilisé dans la description et l'étude des systèmes caractérisés par être concurrents, parallèles, asynchrones, non-déterministes, et/ou stochastiques.

Les réseaux de Petri ont été développés par C.A. Petri en 1962 [81] afin de modéliser et d'analyser les propriétés structurelles des systèmes de communication. Leur champ d'application s'est étendu à une variété de domaines, tels les systèmes

de fabrication d'automates, les réseaux informatiques, les réseaux de communication, les réseaux de trafic aérien, le domaine de l'économie, et les réseaux mobiles cellulaires. L'avantage de la modélisation par RdP inclut la facilité de développement des modèles, la représentation graphique des états du système, et la simplicité du calcul des propriétés structurelles du système. La modélisation par RdP offre, aussi, la possibilité d'effectuer une analyse qualitative des propriétés logiques des systèmes.

Les *RdP stochastiques généralisés* (*RdPSG*), sont une extension des RdP classiques qui associent des transitions temporisées aux transitions immédiates. Leur principal avantage réside dans la possibilité et la facilité de représentation des opérations dont l'exécution nécessite un certain temps, telles que le service d'une requête de client, aussi bien que les évènements instantanés comme les opérations logiques, la synchronisation, et la concurrence. Un autre avantage de la modélisation par RdPSG est qu'ils permettent de calculer les paramètres quantitatifs du système modélisé tels que le nombre moyen de serveurs oisifs dans un réseau, le flux d'arrivées de clients dans un système, etc. Il peut être démontré que le graphe d'accessibilité d'un RdPSG markovien est isomorphe à une chaîne de Markov incluse. Ainsi, en utilisant des techniques bien connues pour les chaînes de Markov, le calcul des différents indices de performance peut être fait. Les RdPSG sont des modèles formels très puissants pour analyser les performances des différents systèmes.

Ce chapitre est une fenêtre ouverte sur le réseaux de Petri. Nous avons commencé par présenter un rappel sur les concepts de base des processus stochastiques, particulièrement les chaînes de Markov à temps discret et à temps continu, afin de bien entamer les réseaux de Petri stochastiques généralisés. Nous avons ensuite parlé des RdP ordinaires, qui constituent la base de tous les RdP de haut-niveau. Puis nous nous sommes focalisés sur l'étude des RdPSG.

Chapitre 3

Proposition d'une Approche d'Analyse

3.1 Introduction

Les modèles de files d'attente avec rappel étudiés dans la littérature supposent dans la plupart des cas que les clients et les serveurs sont homogènes. Cependant, dans beaucoup d'applications pratiques, comme les systèmes de télécommunication, les réseaux mobiles cellulaires, et les réseaux informatiques, le flux d'arrivée des clients peut varier d'un groupe de clients à l'autre, et les temps de service peuvent être différents d'un serveur à l'autre. Ce genre de système est connu par les systèmes avec rappel *hétérogènes*. L'omission du caractère d'hétérogénéité pendant le processus de modélisation et d'analyse de ces systèmes peut engendrer des résultats erronés, qui ne reflètent pas les caractéristiques réelles du système étudié.

Les files d'attente avec rappel étaient pour longtemps le moyen standard utilisé pour analyser les performances des systèmes à files d'attente avec rappel. Cependant, l'introduction du phénomène de l'hétérogénéité (en plus du flux de rappel) provoque une énorme difficulté analytique, ce qui explique le manque de résultats élaborés dans ce domaine. D'une manière générale, la plupart des résultats sont obtenus soit par simulation ou par approximation, et les résultats explicites n'existent que pour quelques cas particuliers exigeant des contraintes sur certains paramètres comme la capacité de la population, le nombre de serveurs, l'homogénéité des clients et

serveurs, etc.

Les réseaux de Petri stochastiques généralisés RdPSG, constituent une alternative très puissante pour la représentation graphique, l'étude, et l'analyse qualitative et quantitative des performances des systèmes avec rappel hétérogènes, et ceci grâce à leur aptitude à représenter les phénomènes de blocage liés aux appels répétés, aux aspects de synchronisation, de concurrence, de parallélisme ainsi que d'autres phénomènes stochastiques dans un même modèle. Ainsi, un intérêt particulier a été accordé aux RdPSG dans le domaine de la fiabilité et de l'évaluation des performances.

L'objectif du présent chapitre est de proposer une approche algorithmique basée sur les RdPSG, pour la modélisation ainsi que l'évaluation des performances des systèmes avec rappel et source finie de clients, et à deux classes de serveurs hétérogènes. Cette approche permet d'effectuer l'analyse qualitative, ainsi que le calcul des différents indices de performance exacts de ces systèmes, tout en considérant deux politiques de service : la politique de *service aléatoire*, et la politique du *service le plus rapide*. Par ailleurs, d'un point de vue d'implémentation, cette technique est optimisée car elle permet d'obtenir directement le générateur infinitésimal Q, tout en évitant de passer par le graphe d'accessibilité GMA, et de la chaîne de Markov à temps continu réduite.

Nous allons commencer par une description détaillée du modèle en question, puis nous allons présenter notre solution pour chacune des politiques de service ; expliquer la structure et la dynamique du RdPSG équivalent, procéder à son analyse quantitative pour démontrer qu'il est ergodique, tracer la chaîne de Markov réduite sous-jacente, ce qui nous permettra de déduire les taux de transitions constituant le générateur infinitésimal, et finalement développer les formules de calcul des différents indices de performance. Notons que nous avons développé un outil en $C\#$ pour implémenter cette approche, nous allons aussi exposer quelques expérimentations, d'une part, afin de valider notre modèle, en comparant les valeurs de quelques indices de performance obtenus par notre programme dans le cas homogène avec celles obtenus par Falin et Templeton [38], et d'autre part, afin de comparer les deux disciplines de service, et d'étudier l'effet du phénomène de rappel et autres para-

mètres sur le temps de réponse moyen du système. Nous terminerons notre chapitre par une conclusion.

3.2 Les systèmes avec rappel à deux classes de serveurs et source finie

3.2.1 Description mathématique

Nous considérons un système avec rappel à source limitée de clients homogènes de taille L ($L < +\infty$), et à deux classes de serveurs. À tout instant, chaque client est soit *libre*, *en service* ou bien *en orbite*. Les clients arrivent dans le système suivant un processus quasi-aléatoire de taux λ. Ainsi, un client libre génère un appel primaire pendant l'intervalle $(t, t + dt)$ avec la probabilité $\lambda dt + o(dt)$, indépendamment des autres clients.

La station de service comprend, quant à elle, deux classes de serveurs C_1 et C_2 ; la classe C_1 (respectivement C_2) est composée de S_1 (respectivement S_2) serveurs identiques, dont les temps de service sont indépendants et distribués selon une loi exponentielle de paramètre μ_1(respectivement μ_2). Les serveurs de la première classe (C_1) sont supposés être plus rapides que ceux de la classe C_2 ($\mu_1 > \mu_2$). Le nombre total de serveurs étant S ($S = S_1 + S_2$). Chaque serveur peut être soit dans l'état *libre* soit dans l'état *occupé*.

Lors de l'arrivée d'un appel d'un client, si tous les serveurs sont occupés, le client rejoint l'orbite et commence à générer un flux d'appels répétés distribué exponentiellement avec le taux ν jusqu'à ce qu'il trouve un serveur libre. Cependant, si un ou plusieurs serveurs (de la classe C_1 ou C_2) sont libres au moment de la réception de la requête d'un service, selon la politique de service adoptée, deux scénarios différents peuvent avoir lieu :

- Dans la politique d'accès aléatoire ; la requête est destinée à l'un des serveurs libres, choisi aléatoirement parmi ceux de la première classe C_1 ou ceux de la deuxième classe C_2,
- Dans la politique du service le plus rapide (prioritaire) ; la requête est affectée

aléatoirement à un serveur de la classe C_1(supposée être la plus rapide) si au moins un serveur de cette classe est libre, sinon, un serveur choisi aléatoirement dans la classe C_2 prend en charge la requête.

Les temps d'inter-arrivée, les temps de service, et les temps de rappel sont supposés être mutuellement indépendants.

Dans ce qui suit, nous allons décrire notre méthode de modélisation et d'analyse de ces systèmes avec rappel à deux classes de serveurs, en utilisant les réseaux de Petri stochastiques généralisés (RdPSG), et ce, en considérant les deux disciplines de service.

3.3 Analyse du modèle avec service aléatoire

Dans cette section, nous allons présenter le RdPSG décrivant les systèmes avec discipline du service aléatoire, puis, à travers une étude stochastique, nous allons décrire la structure de la CMTC sous-jacente en fonction du nombre de clients en orbite, et du nombre de serveurs occupés dans chaque classe, ce qui nous permettra d'obtenir les différents éléments (taux de transitions) du générateur infinitésimal, et par la suite, déduire l'algorithme permettant l'obtention automatique de ce générateur.

3.3.1 Description du RdPSG

Le RdPSG décrit dans la figure 3.1 modélise un système avec rappel à source finie de clients et à deux classes de serveurs. Dans ce modèle :

- La place *ClientsLibres* contient les clients libres,
- La place *Orbite* contient les clients en orbite,
- La place *Choix* comprend un client (libre ou en orbite) demandant le service,
- La place *S1Libre* représente le nombre de serveurs libres de la classe C_1,
- La place *S1occupé* contient le nombre de serveurs occupés de la classe C_1.
- La place *S2Libre* représente le nombre de serveurs libres de la classe C_2,
- La place *S2occupé* contient le nombre de serveurs occupés de la classe C_2,
- La transition *Arrivée* correspond à l'arrivée d'un client (Appel primaire),

- La transition immédiate *DébSer1* est associée au début du service d'un client par un serveur de la classe C_1,
- La transition *FinSer1* représente la fin du service d'un client qui était servi par un serveur de la classe C_1,
- La transition immédiate *DébSer2* est associée au début du service d'un client par un serveur de la classe C_2,
- La transition *FinSer2* représente la fin du service d'un client qui était servi par un serveur de la classe C_2,
- La transition immédiate *Blocage* décrit l'entrée en orbite d'un client qui a trouvé tous les serveurs occupés,
- La transition *Rappel* correspond à l'arrivée d'un appel répété, i.e. provenant d'un client en orbite.

Initialement, l'orbite est vide, tous les clients sont libres et tous les serveurs sont disponibles. Ainsi, le marquage initial peut être exprimé sous cette forme :

$$\begin{aligned} M_0 &= \{M(ClientLibres), M(Choix), M(Orbite), M(S1Libre), M(S1occupé), \\ &\quad M(S2Libre), M(S2occupé)\} \\ &= \{L, 0, 0, S_1, 0, S_2, 0\} \end{aligned}$$

L'arrivée d'un appel primaire est traduit dans le RdPSG par le franchissement de la transition *Arrivée*, dont le taux de franchissement est égal à λ, et la sémantique de service est *à serveurs infini* (#), parce que tous les clients libres peuvent générer des appels primaires, indépendamment les uns des autres. La place *Choix* est alors marquée. Selon les marquages des deux places *S1Libre* et *S2Libre*, on a les cas de figures suivants :

- Si les deux places sont vides, ce qui indique qu'aucun serveur n'est libre dans les deux classes, les deux transitions *DébSer1* et *DébSer2* restent insensibilisées, à l'opposé de la transition immédiate *Blocage* qui, quant à elle, est sensibilisée par l'absence de jetons dans les deux places *S1Libre* et *S2Libre* (arcs inhibiteurs) et la présence de jetons dans *Choix*. Ainsi, cette transition est franchie,

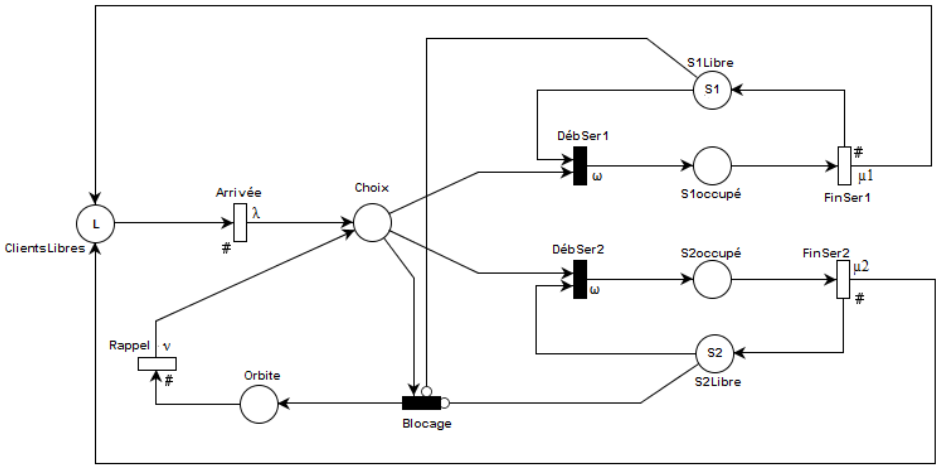

FIG. 3.1 – RdPSG modélisant les systèmes avec rappel, deux classes de serveurs et service aléatoire.

et le client générant l'appel primaire entre en orbite, et devient une source d'un flux d'appels répétés distribués exponentiellement avec le taux ν.

Le franchissement de la transition *Rappel* correspond à l'arrivée d'un appel répété d'un client en orbite. Cette transition est à sémantique de serveurs infinis, car tous les clients en orbite peuvent déclencher des rappels ;

– Si une des deux places *S1Libre* ou *S2Libre* est marquée et l'autre ne contient aucun jeton, ce qui revient à dire qu'il y a des serveurs libres d'une classe, et tous les serveurs de l'autre classe sont occupés. Dans ce cas, la transition immédiate correspondant à la classe ayant des serveurs disponibles (*DébSer1* pour la classe C_1, et *DébSer2* pour C_2) sera tirée. Ainsi, le client commence son service, et le serveur passe à l'état occupé. Notons que la transition *Blocage* reste inhibée par la classe qui contient des serveurs libres ;

– Dans le cas où les deux places sont marquées (existence de serveurs disponibles dans les deux classes), les deux transitions immédiates *DébSer1* et *DébSer2*, qui sont déjà en *conflit structurel*, entrent en *conflit effectif*. Comme les deux classes n'ont aucune priorité l'une par rapport à l'autre, dans le cas de la

discipline du service aléatoire, le même poids :

$$w(DébSer1) = w(DébSer2) = \omega$$

est affecté aux deux transitions. Par conséquent, le tir de l'une ou l'autre de ces dernières est probabiliste, et la probabilité de tirer chaque transition est donnée par la formule suivante :

$$\begin{aligned} P[tir(DébSer1)] &= P[tir(DébSer2)] \\ &= \frac{w(DébSer1)}{w(DébSer1) + w(DébSer2)} \\ &= \frac{w(DébSer2)}{w(DébSer1) + w(DébSer2)} \\ &= \frac{\omega}{2\omega} \\ &= \frac{1}{2}. \end{aligned}$$

Les deux classes ont donc les mêmes chances d'être choisies pour servir la demande du client.

Quand le service du client aura fini, la transition temporisée représentant la fin de service de la classe du serveur qui a pris en charge le client sera franchie, il s'agit de *FinSer1* pour un serveur de la classe C_1, et *FinSer2* pour la classe C_2. Les taux de franchissement de ces deux transitions dépendent des taux de service des classes en question (μ_1 pour la classe C_1, et μ_2 pour la classe C_2). Comme plusieurs serveurs peuvent être occupés en même temps, la sémantique de ces deux transitions est à serveurs infinis. Après la fin de service, le client retourne à l'état libre (un jeton dans la place *Libre*) et le serveur est de nouveau oisif (places *S1Libre* ou bien *S2Libre*).

3.3.2 Analyse du RdPSG

Quelque soit la valeur de L, S_1 et S_2, la conservation du nombre de clients et de serveurs des deux classes, donne les équations suivantes :

$$\begin{cases} M(S1Libre) + M(S1occupé) = S_1 \\ M(S2Libre) + M(S2occupé) = S_2 \\ M(ClientLibres) + M(S1occupé) + M(S2occupé) + M(Orbite) = L \end{cases} \quad (3.1)$$

À partir de ces trois équations, l'état du système en régime stationnaire peut être décrit en définissant les trois variables i, j, k, qu'on appelle un *micro-état*, telles que :

- i représente le nombre de clients servis par un serveur de la classe C_1 (dans la place *S1occupé*),
- j représente le nombre de clients servis par un serveur de la classe C_2 (dans la place *S2occupé*), et
- k représente le nombre de clients en orbite, i.e. dans la place *Orbite*.

Ainsi, ayant les micro-états (i, j, k), on peut facilement déduire le marquage dans chaque place, étant donné que $M(S1Libre) = S_1 - i$, $M(S2Libre) = S_2 - j$ et $M(ClientLibres) = L - (i + j + k)$.

D'un autre coté, en appliquant le système d'équation (3.1), on peut déduire que :

$$\begin{cases} 0 \leq i \leq S_1 \\ 0 \leq j \leq S_2 \\ 0 \leq k \leq L - S \text{ où } S = (S_1 + S_2) \end{cases} \quad (3.2)$$

La CMTC correspondant au modèle RdPSG proposé dans le cas de service aléatoire est donnée dans la figure (3.2).

3.3.3 Algorithme de construction du générateur infinitésimal

En analysant la figure (3.2), nous avons constaté que le nombre total des états de la CMTC réduite, qui correspond au nombre total des marquages tangibles, est égal à n où $n = (S_1 + 1).(S_2 + 1)[(1 + L - (S_1 + S_2)]$. Le générateur infinitésimal Q est donc de l'ordre $[(S_1 + 1).(S_2 + 1).(1 + L - S)]^2$. tel que $S = S_1 + S_2$.

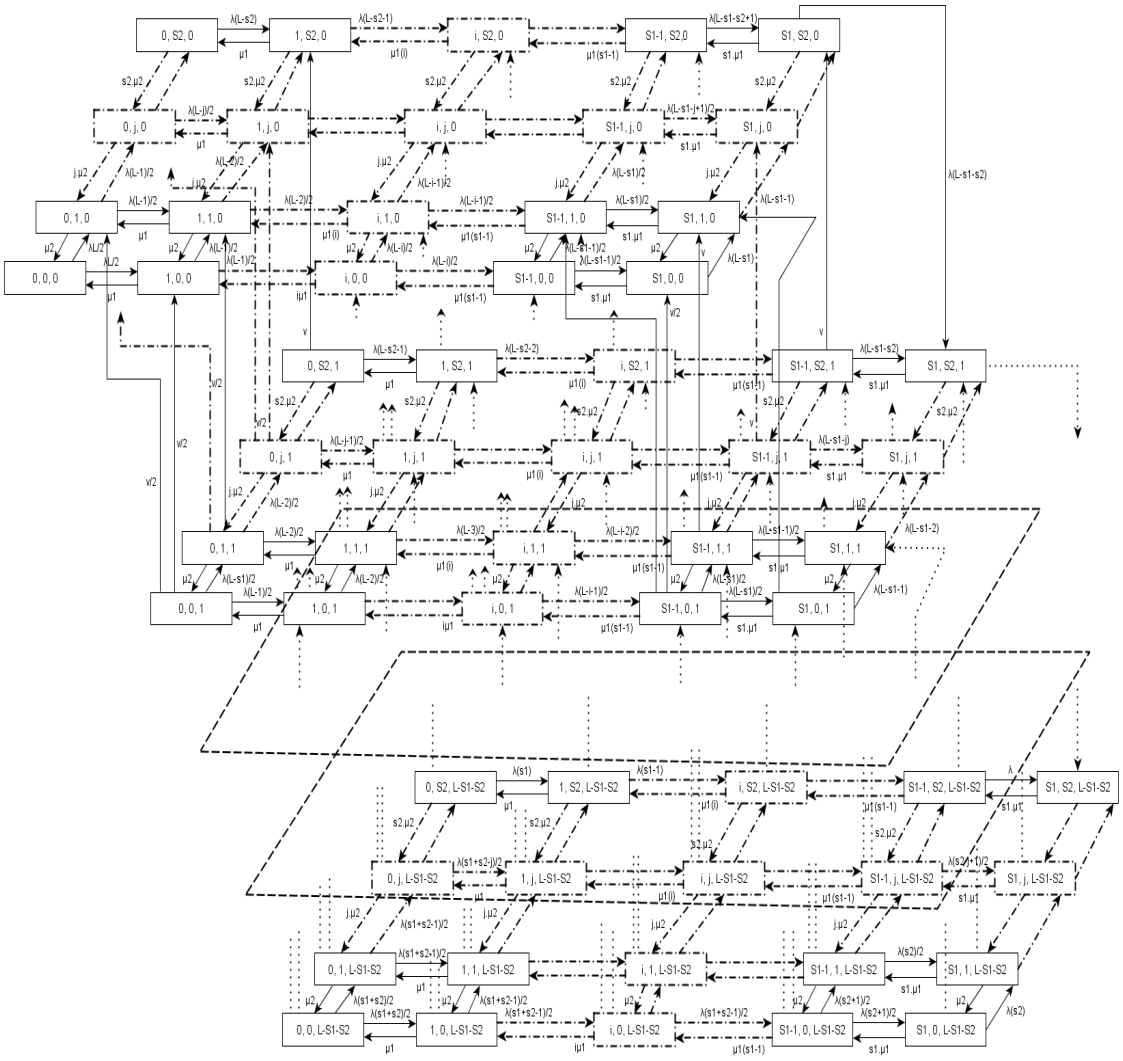

FIG. 3.2 – La CMTC réduite décrivant le modèle avec rappel et service aléatoire.

Il est construit de la manière suivante :

$$Q[(i,j,k),(x,y,z)] = \begin{cases} q[(i,j,k),(x,y,z)] & si (i,j,k) \neq (x,y,z); \\ -\sum_{(l,m,n)\neq(i,j,k)} q[(i,j,k),(l,m,n)] & si (i,j,k) = (x,y,z). \end{cases}$$

Les taux $q[(i,j,k)(x,y,z)]$ sont les taux de transition entre l'état (i,j,k) et l'état (x,y,z). Ils sont donnés par :

- $[0 \leq i < S_1, 0 \leq j < S_2] : (i,j,k) \xrightarrow{\frac{1}{2}(L-i-j-k)\lambda} (i+1,j,k)$ et $(i,j,k) \xrightarrow{\frac{1}{2}(L-i-j-k)\lambda} (i,j+1,k)$
- $[0 \leq i < S_1] : (i,S_2,k) \xrightarrow{(L-i-S_2-k)\lambda} (i+1,S_2,k)$,
- $[0 \leq j < S_2] : (S_1,j,k) \xrightarrow{(L-S_1-j-k)\lambda} (S_1,j+1,k)$,

- $[0 \leq k < L - S] : (S_1, S_2, k) \xrightarrow{(L-S-k)\lambda} (S_1, S_2, k+1)$,
- $[0 < i \leq S_1] : (i, j, k) \xrightarrow{i\mu_1} (i-1, j, k)$,
- $[0 < j \leq S_2] : (i, j, k) \xrightarrow{j\mu_2} (i, j-1, k)$,
- $[0 \leq i < S_1, 0 \leq j < S_2, 0 < k \leq L - S] : (i, j, k) \xrightarrow{\frac{1}{2}k\nu} (i+1, j, k-1)$ et $(i, j, k) \xrightarrow{\frac{1}{2}k\nu} (i, j+1, k-1)$,
- $[0 \leq i < S_1, 0 < k \leq L - S] : (i, S_2, k) \xrightarrow{k\nu} (i+1, S_2, k-1)$,
- $[0 \leq j < S_2, 0 < k \leq L - S)] : (S_1, j, k) \xrightarrow{k\nu} (S_1, j+1, k-1)$,

Ces transitions nous permettent de construire l'algorithme donné ci-dessous, pour calculer les valeurs des différents éléments du générateur infinitésimal Q.

Algorithm 3.1 Construction du générateur infinitésimal pour la politique aléatoire.

```
Début
Pour k=0 à L-S
   Pour i=0 à S1-1
      Pour j=0 à S2-1
         Q[(i,j,k),(i+1,j,k)]=1/2(L-i-j-k)λ //admission en C1
         Q[(i,j,k),(i,j+1,k)]=1/2(L-i-j-k)λ //admission en C2
         Q[(S1,j,k),(S1,j+1,k)]=(L-S1-j-k)λ //admission en C2,
                                              C1 étant complet
      FinPour
      Q[(i,S2,k),(i+1,S2,k)]=(L-i-S2-k)λ //admission en C1,
                                           C2 étant complet
   FinPour
FinPour
Pour k=0 à L-S-1
   Q[(S1,S2,k),(S1,S2,k+1)]=(L-S-k)λ //Entrée en orbite
FinPour
Pour k=0 à L-S
   Pour i=1 à S1
      Pour j=0 à S2
         Q[(i,j,k),(i-1,j,k)]=iμ1 //Fin service en C1
```

```
            FinPour
         FinPour
         Pour j=1 à S2
            Pour i=0 à S1
               Q[(i,j,k),(i,j-1,k)]=jμ2 //Fin service en C2
            FinPour
         FinPour
      FinPour
      Pour k=1 à L-S
         Pour i=0 à S1-1
            Pour j=0 à S2-1
               Q[(i,j,k),(i+1,j,k-1)]=1/2.k.ν//Rappel et admission en C1
               Q[(i,j,k),(i,j+1,k-1)]=1/2.k.ν//Rappel et admission en C2
            FinPour
            Q[(i,S2,k),(i+1,S2,k-1)]=kν //Rappel et admission en C1
                                          (C2 complet)
         FinPour
         Pour j=0 à S2-1
            Q[(S1,j,k),(S1,j+1,k)]=kν //Rappel et admission en C2
                                        (C1 complet)
         FinPour
      FinPour
      Fin.
```

3.4 Analyse du modèle avec discipline du service le plus rapide

Pour éviter toute répétition, nous n'allons détailler dans ce paragraphe, que les points caractérisant la politique du service le plus rapide par rapport à la politique aléatoire. Les places et les transitions ainsi que les variables définies dans la section précédente restent valables.

3.4.1 Description du RdPSG

Sous l'hypothèse que le temps moyen de service des serveurs appartenant à la classe C_2 soit *supérieur* à celui des serveurs de la classe C_1, il est plus judicieux d'affecter les demandes de clients à ces derniers afin d'assurer un temps de réponse meilleur. Il s'agit de la discipline du *service le plus rapide* ou encore la discipline *prioritaire*. La souplesse des RdPSG permet d'obtenir facilement le modèle correspondant à cette discipline. Il est représenté dans la figure (3.3).

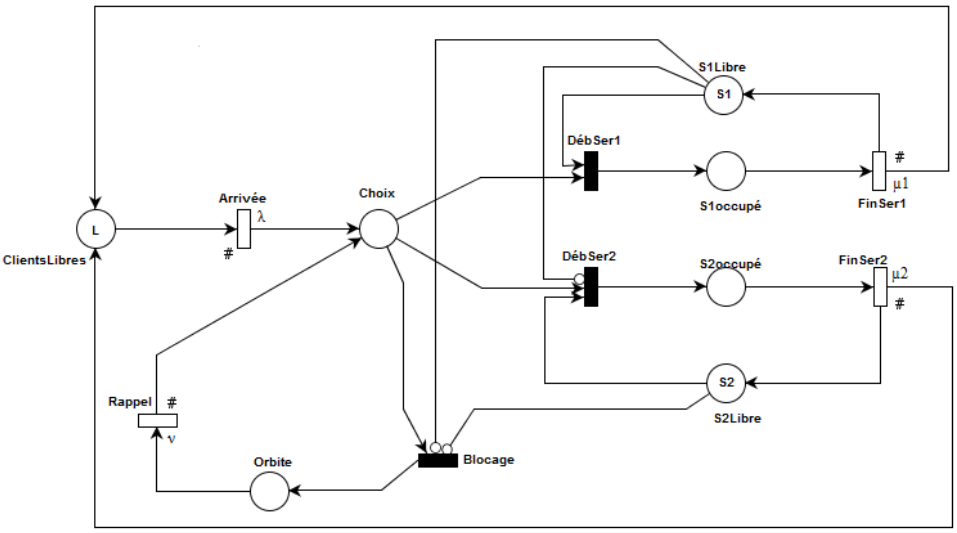

FIG. 3.3 – RdPSG modélisant les systèmes avec rappel, deux classes de serveurs et discipline du service le plus rapide.

En effet, le franchissement de la transition *Arrivée*, qui traduit l'arrivée d'un appel primaire, provoque le marquage de la place *Choix*. Si au moins un serveur de la classe C_1 est libre (place *S1Libre* marquée), il prendra en charge l'appel du client (franchissement de la transition *DébSer1*), la transition *DébSer2* étant inhibée, elle ne peut pas être franchie malgré la présence de serveurs libres de la classe C_2, ce qui exprime la priorité entre ces deux classes. Dans le cas contraire, i.e. si la place *S1Libre* est vide et des serveurs de la classe C_2 sont disponibles, la transition *DébSer2* devient franchissable et l'un de ses serveurs s'occupe de l'appel.

Quant à la transition *Blocage*, qui est inhibée par les deux places *S1Libre* et *S2Libre*, elle n'est franchissable que s'il n'y a aucun serveur oisif, ni de la classe

C_1 ni de la classe C_2, le client rejoint donc l'orbite, et commence à générer un flux d'appels répétés distribué exponentiellement avec le taux ν, comme le représente la transition *Rappel*.

3.4.2 Analyse et Construction du générateur infinitésimal

L'analyse du modèle de RdPSG se fait de la même manière que pour le système précédent (discipline aléatoire). La CMTC induite et réduite de ce système est donnée dans la figure (3.4).

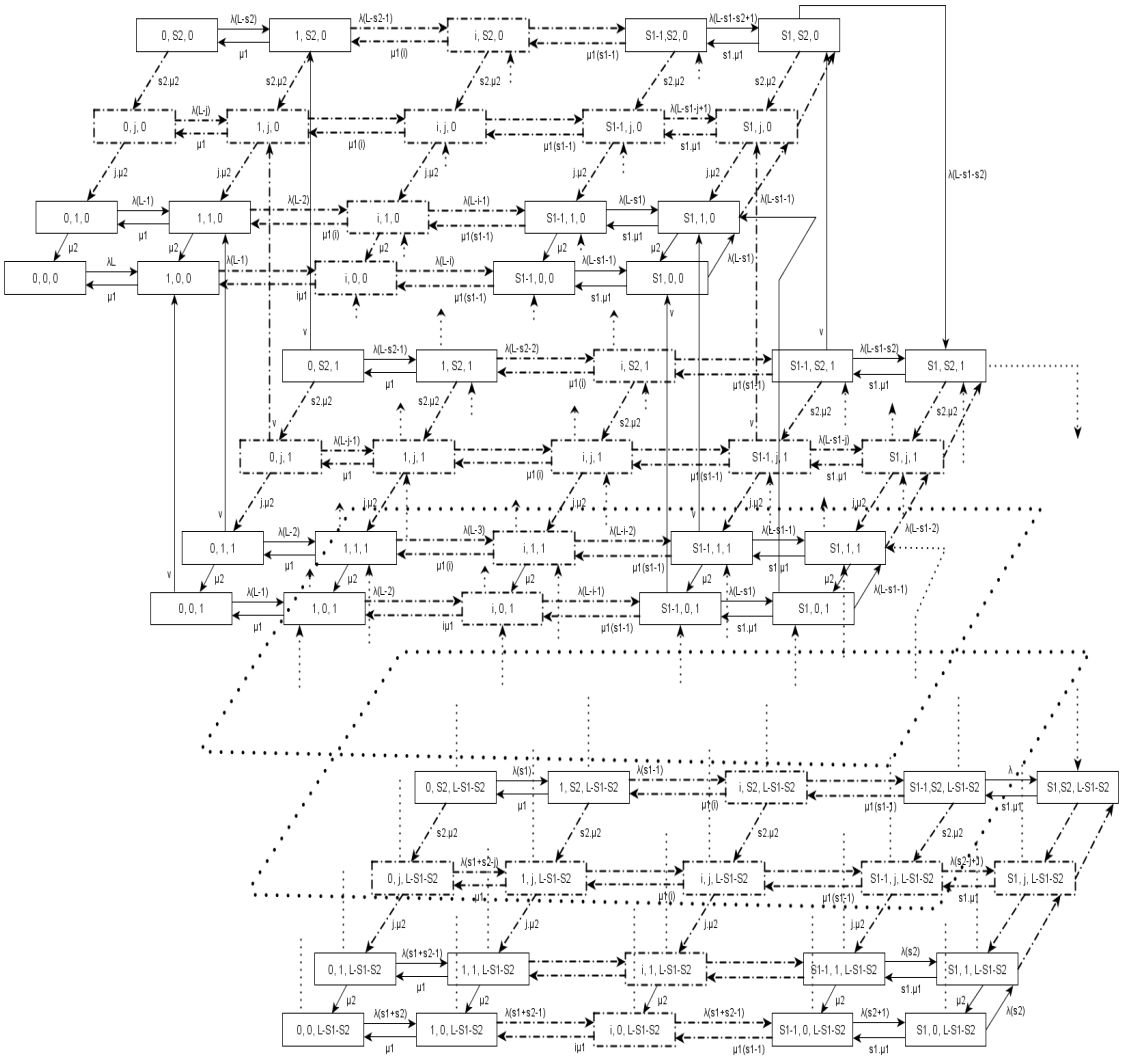

FIG. 3.4 – La CMTC réduite décrivant le modèle avec rappel et service le plus rapide.

Les taux de transition $q[(i, j, k)(x, y, z)]$ pour la discipline du service le plus rapide sont donnés par :

- $[0 \leq i < S_1] : (i, j, k) \xrightarrow{(L-i-j-k)\lambda} (i+1, j, k)$,
- $[0 \leq j < S_2] : (S_1, j, k) \xrightarrow{(L-S_1-j-k)\lambda} (S_1, j+1, k)$,
- $[0 \leq k < L - S] : (S_1, S_2, k) \xrightarrow{(L-S-k)\lambda} (S_1, S_2, k+1)$,
- $[0 < i \leq S_1] : (i, j, k) \xrightarrow{i\mu_1} (i-1, j, k)$,
- $[0 < j \leq S_2] : (i, j, k) \xrightarrow{j\mu_2} (i, j-1, k)$,
- $[0 \leq i < S_1, 0 < k \leq L - S] : (i, j, k) \xrightarrow{k\nu} (i+1, j, k-1)$,
- $[0 \leq j < S_2, 0 < k \leq L - S)] : (S_1, j, k) \xrightarrow{k\nu} (S_1, j+1, k-1)$,

L'algorithme permettant d'obtenir le générateur infinitésimal, est donc le suivant :

Algorithm 3.2 Algorithme de construction du générateur infinitésimal pour la discipline du service le plus rapide.

```
Début
Pour k=0 à L-S
   Pour j=0 à S2
      Pour i=0 à S1-1
         Q[(i,j,k),(i+1,j,k)]=(L-i-j-k)λ//admission en C1
      FinPour
   FinPour
   Pour j=0 à S2-1
      Q[(S1,j,k),(S1,j+1,k)]=(L-S1-j-k)λ//admission en C2,
      C1 complet
   FinPour
FinPour
Pour k=0 à L-S-1
   Q[(S1,S2,k),(S1,S2,k+1)]=(L-S-k)λ //entrée en orbite
FinPour
Pour k=0 à L-S
   Pour i=1 à S1
      Pour j=0 à S2
         Q[(i,j,k),(i-1,j,k)]=iμ1 //Fin service en C1
      FinPour
```

```
      FinPour
      Pour j=1 à S2
         Pour i=0 à S1
            Q[(i,j,k),(i,j-1,k)]=jμ2 //Fin service en C2
         FinPour
      FinPour
   FinPour
   Pour k=1 à L-S
      Pour j=0 à S2
         Pour i=0 à S1-1
            Q[(i,j,k),(i+1,j,k-1)]=kν //rappel et admission en C1
         FinPour
      FinPour
      Pour j=0 à S2-1
         Q[(S1,j,k),(S1,j+1,k-1)]=kν //rappel et admission en C2,
                                        C1 complet
      FinPour
   FinPour
   Fin.
```

3.5 Indices de performance

Le but de cette section est l'obtention des formules des principaux indices de performance, qui sont les mêmes pour les deux politiques de service. Comme les deux modèles proposés sont bornés pour leurs marquages initiaux M_0 et ces derniers sont des états d'accueil, les processus sous-jacents sont ergodiques. Par conséquent, la distribution stationnaire π existe et est unique. En applicant les algorithmes ci-dessus, le générateur infinitésimal Q peut être automatiquement obtenu pour chaque politique de service, puis, la distribution des probabilités stationnaires $\pi = (\pi_1, \pi_2, ..., \pi_n)$ peut être calculée en résolvant le système d'équations linéaires (3.3).

$$\begin{cases} \pi.Q = 0 \\ \sum_{i=1}^{n} \pi_i = 1 \end{cases} \tag{3.3}$$

À partir du vecteur des probabilités stationnaires π, nous pouvons calculer la formule exacte de plusieurs paramètres de performance des systèmes avec rappel à source finie de clients et deux classes de serveurs. Notons par $M_i(p)$ le nombre de jetons dans la place p dans le marquage M_i, A l'ensemble des marquages tangibles accessibles, par $A(t)$ l'ensemble des marquages tangibles accessibles par la transition t, et par $\pi_{i,j,k}$ la probabilité que le système soit dans le micro-état (i, j, k) à l'état stationnaire.

– Le nombre moyen de clients libres :

Il correspond au nombre moyen de jetons dans la place *ClientsLibres*,

$$n_{CliLib} = \sum_{i:M_i \in A} M_i(ClientsLibres).\pi_i = \sum_{k=0}^{L-S} \sum_{i=0}^{S_1} \sum_{j=0}^{S_2} (L - i - j - k).\pi_{i,j,k}$$

– Le nombre moyen de clients en orbite :

Ce qui correspond au nombre moyen de jetons dans la place *Orbite*,

$$n_{Orb} = \sum_{i:M_i \in A} M_i(Orbite).\pi_i = \sum_{k=0}^{L-S} \sum_{i=0}^{S_1} \sum_{j=0}^{S_2} k.\pi_{i,j,k}$$

– Le nombre moyen de serveurs occupés appartenant à la classe C_1 :

Notons que c'est aussi le nombre moyen de clients en service par la classe C_1, il correspond au nombre moyen de jetons dans la place *S1occupé*,

$$n_{OccC_1} = \sum_{i:M_i \in A} M_i(S1occupé).\pi_i = \sum_{k=0}^{L-S} \sum_{i=0}^{S_1} \sum_{j=0}^{S_2} i.\pi_{i,j,k}$$

– Le nombre moyen de serveurs occupés appartenant à la classe C_2 :

C'est aussi le nombre moyen de clients en service par la classe C_2, il correspond au

nombre moyen de jetons dans la place *S2occupé*,

$$n_{OccC_2} = \sum_{i:M_i \in A} M_i(S2occupé).\pi_i = \sum_{k=0}^{L-S} \sum_{i=0}^{S_1} \sum_{j=0}^{S_2} j.\pi_{i,j,k}$$

– Le nombre moyen de serveurs occupés :

Ça correspond à la somme de nombre moyen de serveurs occupés des deux classes

$$n_{Occ} = n_{OccC_1} + n_{OccC_2} = \sum_{k=0}^{L-S} \sum_{i=0}^{S_1} \sum_{j=0}^{S_2} (i+j).\pi_{i,j,k}$$

– Le nombre moyen de clients dans le système :

Il correspond au nombre moyen de clients en orbite plus ceux en service par C_1, plus ceux en service par C_2,

$$n = n_{Orb} + n_{Occ} = \sum_{k=0}^{L-S} \sum_{i=0}^{S_1} \sum_{j=0}^{S_2} (i+j+k).\pi_{i,j,k}$$

– Le nombre moyen de serveurs disponibles appartenant à la classe C_1 :

Il correspond au nombre moyen de jetons dans la place *S1Libre*,

$$n_{LibC_1} = \sum_{i:M_i \in A} M_i(S1Libre).\pi_i = \sum_{k=0}^{L-S} \sum_{i=0}^{S_1} \sum_{j=0}^{S_2} (S_1 - i).\pi_{i,j,k} = S_1 - n_{OccC_1}$$

– Le nombre moyen de serveurs disponibles appartenant à la classe C_2 :

il correspond au nombre moyen de jetons dans la place *S2Libre*,

$$n_{LibC_2} = \sum_{i:M_i \in A} M_i(S2Libre).\pi_i = \sum_{k=0}^{L-S} \sum_{i=0}^{S_1} \sum_{j=0}^{S_2} (S_2 - j).\pi_{i,j,k} = S_2 - n_{OccC_2}$$

– Le nombre moyen de serveurs libres :

Ça correspond à la somme de nombre moyen de serveurs libres des deux classes

$$n_{Lib} = n_{LibC_1} + n_{LibC_2} = S - n_{Occ}$$

– La fréquence d'arrivée d'appels primaires (taux moyen de génération d'appels

primaires) :

Il correspond au débit de la transition *Arrivée*,

$$\begin{aligned}\bar{\lambda} &= \sum_{i:M_i \in A(Arrivée)} \lambda.M_i(ClientsLibres).\pi_i \\ &= \sum_{k=0}^{L-S}\sum_{i=0}^{S_1}\sum_{j=0}^{S_2} \lambda.(L-i-j-k).\pi_{i,j,k} \\ &= \lambda.n_{CliLib}\end{aligned}$$

– La fréquence d'arrivée d'appels répétés (taux moyen de génération d'appels répétés) :

Elle correspond au débit de la transition *Rappel*,

$$\begin{aligned}\bar{\nu} &= \sum_{i:M_i \in A(Rappel)} \nu.M_i(Orbite).\pi_i \\ &= \sum_{k=0}^{L-S}\sum_{i=0}^{S_1}\sum_{j=0}^{S_2} \nu.k.\pi_{i,j,k} \\ &= \nu.n_{Orb}\end{aligned}$$

– La fréquence de service de la classe C_1 :

Elle correspond au débit de la transition *FinSer1*,

$$\begin{aligned}\bar{\mu_1} &= \sum_{i:M_i \in A(FinSer1)} \mu_1.M_i(S1occupé).\pi_i \\ &= \sum_{k=0}^{L-S}\sum_{i=0}^{S_1}\sum_{j=0}^{S_2} \mu_1.i.\pi_{i,j,k} \\ &= \mu_1.n_{OccC_1}\end{aligned}$$

– La fréquence de service de la classe C_2 :

Elle correspond au débit de la transition *FinSer2*,

$$\begin{aligned}\bar{\mu_2} &= \sum_{i:M_i \in A(FinSer2)} \mu_2.M_i(S2occupé).\pi_i \\ &= \sum_{k=0}^{L-S}\sum_{i=0}^{S_1}\sum_{j=0}^{S_2} \mu_2.j.\pi_{i,j,k} \\ &= \mu_2.n_{OccC_2}\end{aligned}$$

– La fréquence totale de service :

C'est la somme des fréquences de service des deux classes

$$\bar{\mu} = \bar{\mu_1} + \bar{\mu_2}$$

– La disponibilité de s serveurs appartenant à la classe C_1 $(1 \leq s \leq S_1)$:

C'est la probabilité qu'au moins s serveurs de la classe C_1 soient oisifs

$$A_{sC_1} = \sum_{i:M_i(S1Libre)\geq s} \pi_i = \sum_{k=0}^{L-S}\sum_{i=0}^{S_1-s}\sum_{j=0}^{S_2} \pi_{i,j,k}$$

– La disponibilité de s serveurs appartenant à la classe C_2 $(1 \leq s \leq S_2)$:

C'est la probabilité qu'au moins s serveurs de la classe C_2 soient oisifs

$$A_{sC_2} = \sum_{i:M_i(S2Libre)\geq s} \pi_i = \sum_{k=0}^{L-S}\sum_{i=0}^{S_1}\sum_{j=0}^{S_2-s} \pi_{i,j,k}$$

– La disponibilité de s serveurs dans le système (des deux classes) :

$$A_s = \sum_{i:M_i(S1Libre)+M_i(S2Libre)\geq s} \pi_i = \sum_{k=0}^{L-S}\sum_{i=0}^{S_1}\sum_{j=0,i+j\leq S-s}^{S_2} \pi_{i,j,k}$$

– L'utilisation de s serveurs au moins de la classe C_1 :

Ce qui correspond à la probabilité qu'au moins s serveurs de la classe C_1 soient occupés

$$U_{sC_1} = \sum_{i:M_i(S1occupé)\geq s} \pi_i = \sum_{k=0}^{L-S}\sum_{i=s}^{S_1}\sum_{j=0}^{S_2} \pi_{i,j,k}$$

– L'utilisation de s serveurs au moins de la classe C_2 :

Ce qui correspond à la probabilité qu'au moins s serveurs de la classe C_2 soient

occupés

$$U_{sC_2} = \sum_{i:M_i(S2occupé)\geq s} \pi_i = \sum_{k=0}^{L-S} \sum_{i=0}^{S_1} \sum_{j=s}^{S_2} \pi_{i,j,k}$$

– L'utilisation de s serveurs au moins dans le système (des deux classes) :

$$U_s = \sum_{i:M_i(S1occupé)+M_i(S2occupé)\geq s} \pi_i = \sum_{k=0}^{L-S} \sum_{i=0}^{S_1} \sum_{j=0,i+j\geq s}^{S_2} \pi_{i,j,k}$$

– La probabilité de blocage d'un client primaire :

$$B_p = \frac{\sum_{j:M_j\in A} \sum_{x=1}^{L-S} x.\lambda.P[M_j(ClientsLibres) = x, M_j(S1occupé) = S_1, M_j(S2occupé) = S_2]}{\bar{\lambda}}$$

$$= \frac{\sum_{k=0}^{L-S} (L-k-S).\lambda.\pi_{S_1,S_2,k}}{\bar{\lambda}}$$

– La probabilité de blocage d'un client en orbite :

$$B_r = \frac{\sum_{j:M_j\in A} \sum_{x=1}^{L-S} x.\nu.P[M_j(Orbite) = x, M_j(S1occupé) = S_1, M_j(S2occupé) = S_2]}{\bar{\nu}}$$

$$= \frac{\sum_{k=1}^{L-S} k.\nu.\pi_{S_1,S_2,k}}{\bar{\nu}}$$

– La probabilité de blocage :

$$B = B_p + B_r$$

– Le temps moyen d'attente :

C'est la durée moyenne entre le moment d'arrivée du client et le moment de début de son service. En utilisant la formule de Little, on obtient

$$\bar{W} = \frac{n_{Orb}}{\bar{\lambda}}$$

– Le temps moyen de réponse :

$$\bar{R} = \frac{n}{\bar{\lambda}}$$

3.6 Mise en œuvre, Tests, et Résultats

Nous nous intéressons dans cette section, à l'implémentation de notre approche, ainsi qu'aux principaux résultats issus de nos expérimentations sur différents systèmes. En effet, nous avons développé un outil logiciel d'évaluation des performances des systèmes avec rappel à deux classes de serveurs hétérogènes, qui permet de faire une évaluation quantitative en calculant les différents paramètres de performance. Nous avons voulu à travers les différents tests que nous avons effectués, expliciter l'impact du phénomène de rappel sur les performances des systèmes étudiés. Les hypothèses théoriques rejoignent ainsi les résultats pratiques et deviennent plus perceptibles et palpables. Ainsi, nous présentons l'influence des différents paramètres du système ; à savoir le taux de génération des appels primaires, le taux de rappel, ainsi que le nombre de serveurs dans chaque classe sur un des indices des performances les plus significatifs qui est le temps de réponse moyen.

L'interface de notre application a été réalisée sous le système Windows, en utilisant l'outil *C#* de Microsoft Visual Studio 2010. Ce dernier offre les outils et fonctionnalités utiles au développement d'un environnement convivial et interactif. La fenêtre principale de cette application donne la main à l'utilisateur pour saisir les paramètres en entrée du système à analyser, comme le montre la figure 3.5.

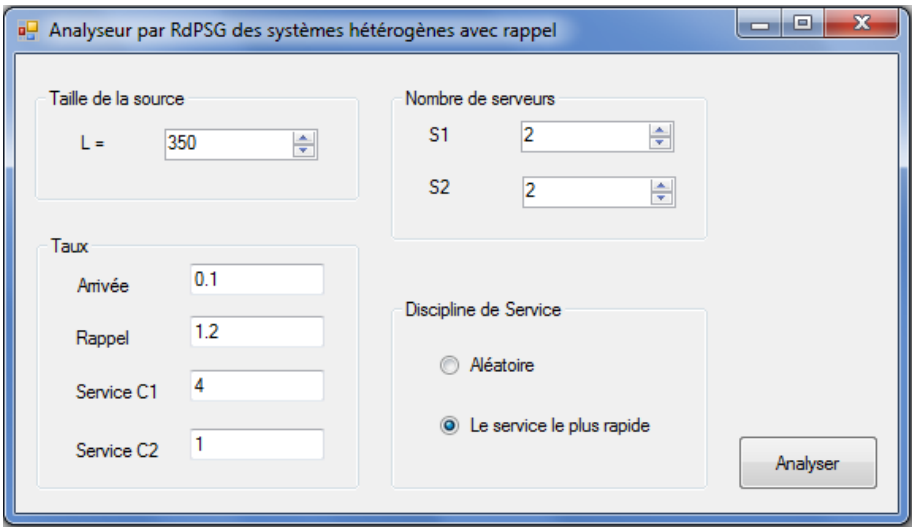

FIG. 3.5 – Fenêtre principale de l'outil d'analyse des performances des systèmes hétérogènes avec rappel.

Une autre fenêtre apparaît alors, pour afficher les paramètres de performance

calculés à l'aide des formules données dans la section précédente. Cette fenêtre donne aussi à l'utilisateur, la possibilité de sauvegarder les résultats numériques obtenus (voir la figure 3.6).

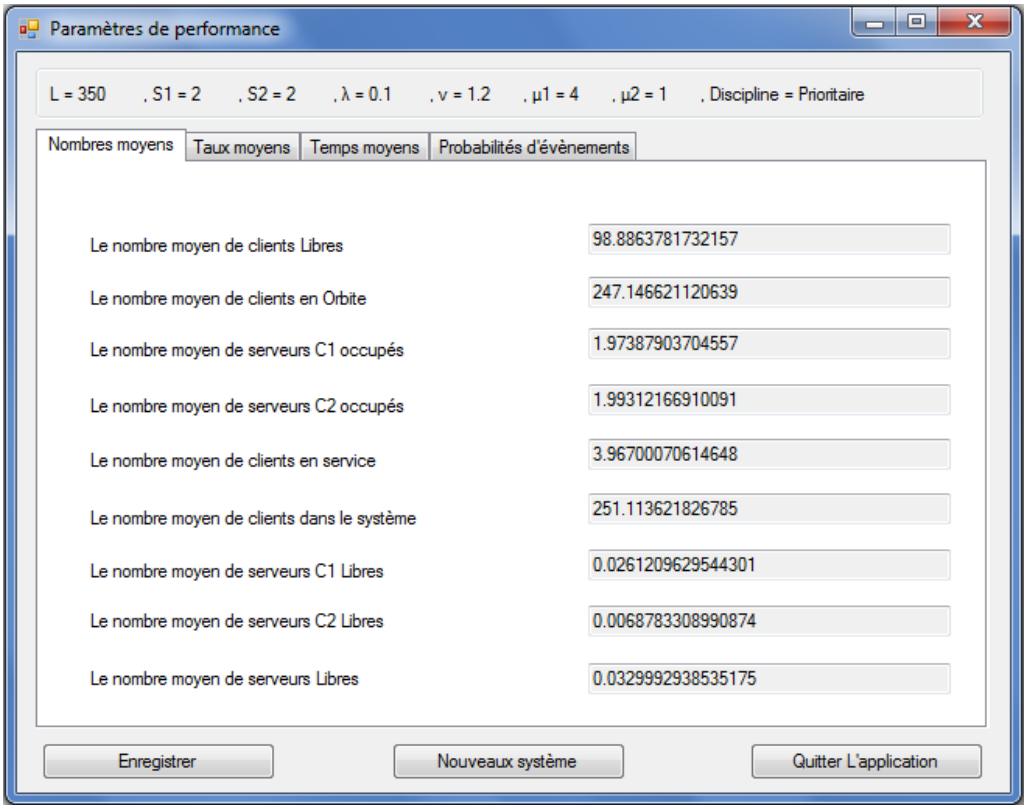

FIG. 3.6 – Affichage des résultats numériques des indices de performance.

Afin de nous assurer des résultats obtenus par notre application, et par conséquent, valider notre approche, nous avons considéré un exemple avec un seul serveur de la classe C_1, et trois serveurs de la classe C_2, avec les mêmes taux de service ($\mu_1 = \mu_2 = 1$, pour simuler une seule classe homogène), et nous avons comparé les résultats avec ceux obtenus par un programme Pascal donné dans le livre de FALIN et TEMPLETON [38], où ils ont considéré un système avec une seule classe de quatre serveurs ayant $\mu = 1$. Dans le tableau 3.1, nous pouvons constater que les deux modèles donnent exactement les mêmes valeurs des indices de performance (au moins à cinq chiffres après la virgule).

	Homogène	Hétérogène	
		Aléatoire	Le plus rapide
Nombre de serveurs	4	S_1= 1,S_2= 3	S_1= 1,S_2= 3
Taille de la source	20	20	20
Tx. de génération d'appels primaires	0.1	0.1	0.1
Taux de service	1	μ_1= 1, μ_2= 1	μ_1= 1, μ_2= 1
Taux de rappel	1.2	1.2	1.2
Nombre moyen de serveurs occupés	1.800 748	C_1 : 0.521 865 C_2 : 1.278 882 Tot : 1.800 747	C_1 : 0.672 602 C_2 : 1.128 145 Tot : 1.800 747
Nombre moyen de clients en orbite	0.191 771	0.191 771	0.191 771
Tx moy. génération appels primaires	1.800 748	1.800 748	1.800 748
Temps d'attente moyen	0.106 495	0.106 495	0.106 495

TAB. 3.1 – Validation dans le cas homogène.

Nous présentons dans ce qui suit, une série d'expérimentations qui vise à étudier l'effet des paramètres en entrée du système sur le temps moyen de réponse. Les paramètres en entrée des systèmes analysés sont résumés dans le tableau 3.2.

	L	S_1	S_2	λ	ν	μ_1	μ_2
Figure 3.7	50	5	2	variable	0.1	8	2
Figure 3.8	50	4	2	0.5	variable	8	2
Figure 3.9	30	variable	4	2	1	6	1
Figure 3.10	30	4	variable	2	1	6	1

TAB. 3.2 – Paramètres en entrée du programme.

Dans la figure 3.7, nous illustrons l'effet que génère la variation du taux d'arrivée des clients dans le système sur le temps moyen de réponse, nous comparons, de plus, les performances des deux disciplines de service ; la discipline aléatoire et celle de la classe la plus rapide, en faisant varier le taux d'arrivée des clients primaires dans le système.

Comme on peut le constater, le temps de réponse croît avec la croissance du taux d'arrivée des requêtes primaires dans le système. Ceci est dû à l'augmentation du temps d'attente des clients dans l'orbite. De plus, les temps de réponse obtenus pour la discipline du service le plus rapide, s'avèrent toujours plus courts que ceux de la discipline aléatoire, ce qui reflète l'efficacité de la discipline du service le plus

rapide.

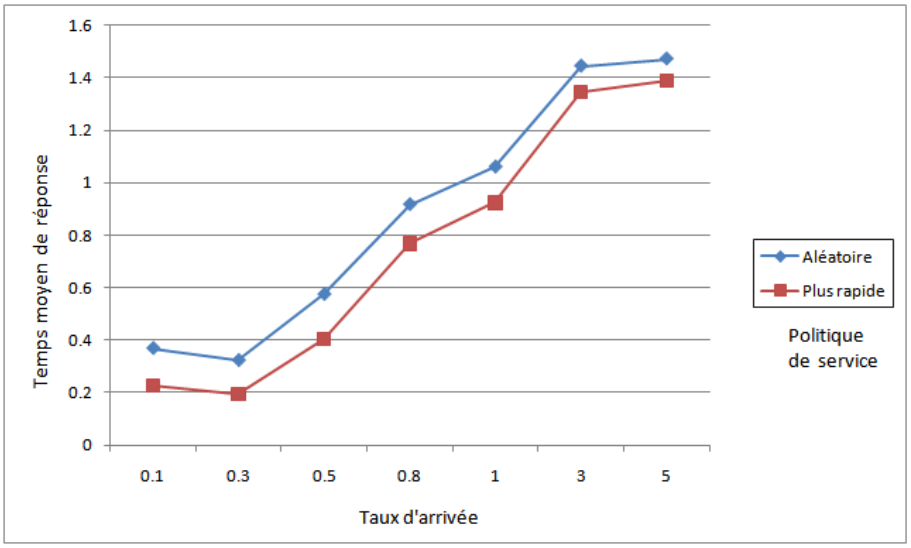

FIG. 3.7 – Influence du taux d'arrivée sur le temps moyen de réponse.

Le graphe de la figure 3.8 met en évidence l'influence du taux de rappel sur le temps de réponse moyen du système étudié, pour les deux disciplines de service. En effet, le temps de réponse décroît en fonction de l'intensité du flux des appels répétés. Cette influence est plus significative quand l'intensité est faible. De plus, les performances obtenues avec la discipline du service le plus rapide sont toujours meilleures que celles de la discipline aléatoire.

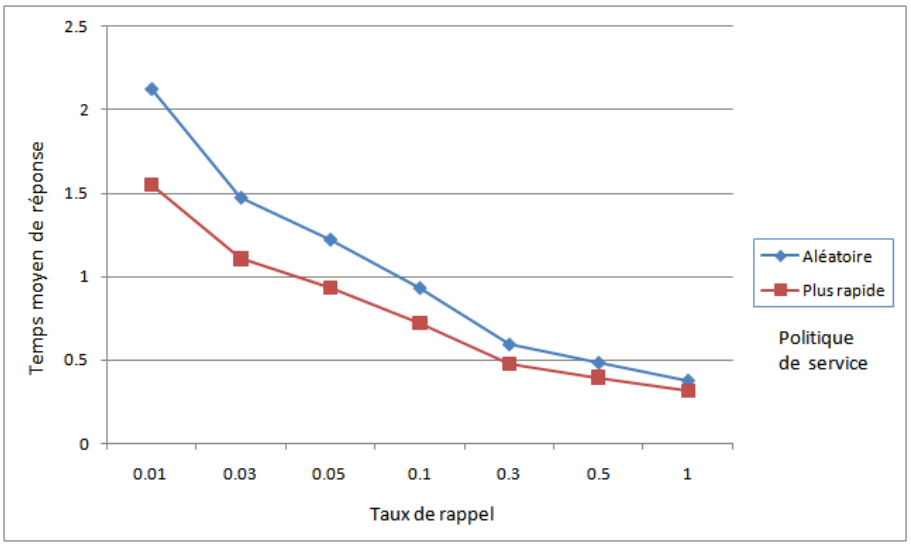

FIG. 3.8 – Influence du taux de rappel sur le temps de réponse moyen.

A partir des figures 3.9, et 3.10, nous pouvons conclure que le nombre de serveurs

de chacune des deux classes influe sur le temps de réponse. Cependant, la vitesse d'influence du nombre de serveurs de la classe C_1 est plus rapide que celle de C_2, car les serveurs de la classe C_1 sont plus rapides. Dans la figure 3.9, le temps de réponse atteint l'optimum et se stabilise après un certain temps (nombre de serveurs = 12), il est donc inutile d'investir dans de nouveaux serveurs de la classe C_1. Il est à noter que les performances de la politique la plus rapide restent toujours meilleures que celles de la politique aléatoire.

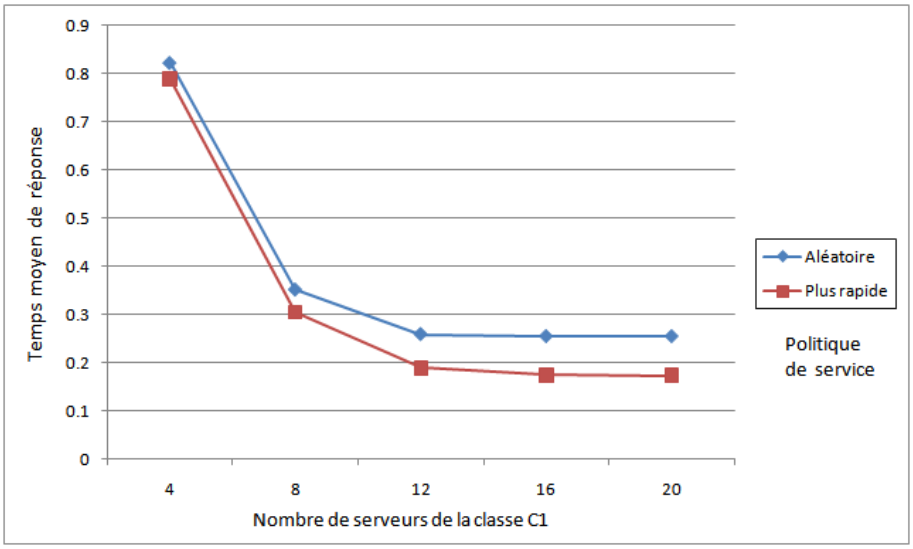

FIG. 3.9 – Influence du nombre de serveurs de la classe C_1 sur le temps de réponse moyen.

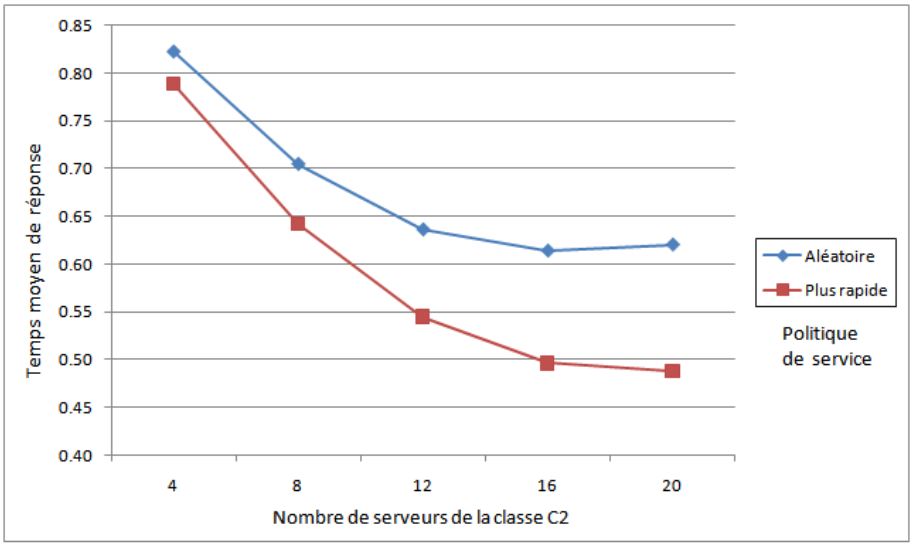

FIG. 3.10 – Influence du nombre de serveurs de la classe C_2 sur le temps de réponse moyen.

3.7 Conclusion

Notre but dans ce chapitre, était de présenter une nouvelle approche algorithmique permettant la modélisation et l'évaluation des performances des systèmes avec rappel, source finie de clients, et deux classes de serveurs hétérogènes, à l'aide des RdPSG. La particularité de cette approche réside dans la construction du générateur infinitésimal sans avoir à générer ni le graphe des marquages accessibles, ni la chaîne de Markov à temps continu sous-jacente.

Les RdPSG sont un outil graphique et mathématique très performant qui permettent la représentation des différentes caractéristiques des systèmes modélisés, telles que la source finie de clients, l'hétérogénéité des serveurs, la priorité entre les classes de serveurs, etc. ainsi que les problèmes de synchronisation et de blocage qui caractérisent les systèmes étudiés.

Nous avons aussi pu, grâce à la flexibilité de ce formalisme, considérer deux disciplines de service ; à savoir la discipline du service aléatoire, et celle du service le plus rapide, et ceci en effectuant de légères modifications sur le modèle RdPSG.

D'autre part, ce formalisme permet une analyse de propriétés qualitatives et quantitatives du système modélisé. Ainsi, un autre avantage de l'approche proposée dans ce mémoire est la possibilité d'exprimer les différents indices de performance en fonction des éléments de base du réseau de Petri stochastique, et des probabilités stationnaires, mais surtout indépendamment des marquages de l'ensemble d'accessibilité.

Conclusion Générale

Ce travail de recherche s'inscrit dans le cadre de l'évaluation des performances des systèmes avec rappel à source finie de clients et deux classes de serveurs hétérogènes. À l'aide des réseaux de Petri stochastiques généralisés, nous avons conçu et mis en œuvre une approche algorithmique pour la modélisation et l'évaluation des performances de ces systèmes.

La théorie des files d'attente avec rappel a vu le jour dans les années quarante du siècle passé, quand les chercheurs se sont aperçus que le modèle standard des files d'attente n'était pas le meilleur moyen pour modéliser les systèmes téléphoniques, car il ne tient pas compte du flux des appels répétés.

Dans le premier chapitre, nous avons présenté un état de l'art sur les principaux travaux et résultats obtenus dans le domaine des files d'attente avec rappel, qui est le modèle habituellement utilisé pour modéliser les systèmes avec rappel. Il est à noter que, vu les difficultés analytiques engendrées par la prise en considération du phénomène de rappel, des formules explicites n'existent que pour certaines variantes de FAR assez simples et particulières où les serveurs doivent être homogènes, et leur nombre ne dépasse pas deux. Pour les autres modèles, les chercheurs se sont dirigés vers la proposition de méthodes d'approximation et de simulation ainsi que des méthodes numériques pour élaborer les formules des indices de performance. Tel est le cas pour notre modèle qui suppose, en plus du phénomène de rappel, une source finie de clients, et deux classes de serveurs hétérogènes. Ceci nous a incités à utiliser les réseaux de Petri stochastiques généralisés (RdPSG), qui font l'objet du deuxième chapitre.

En effet, les RdPSG sont un moyen graphique et mathématique d'une grande puissance descriptive, qui permet de modéliser d'une manière naturelle, le phéno-

mène de rappel, la limitation de la source de clients, et la multiplicité des serveurs dans chaque classe. Ils permettent de plus, de faire l'analyse qualitative et quantitative (i.e. l'évaluation des performances) de ces systèmes.

Le troisième chapitre a été consacré à la conception et la mise en œuvre de notre approche d'évaluation des performances des systèmes avec rappel à source finie de clients et deux classes de serveurs hétérogènes, qui repose sur les RdPSG. Nous avons considéré deux politiques de service, qui sont la politique du service aléatoire, et la politique du service le plus rapide. L'avantage de la méthode que nous avons conçue est qu'elle propose pour chaque politique de service, un algorithme permettant de calculer directement le générateur infinitésimal, sans passer ni par le graphe des marquages accessibles, ni par la chaîne de Markov réduite équivalente. Par ailleurs, nous avons développé des formules de calcul des indices de performance en fonction des paramètres du système et indépendants des marquages accessibles.

Ce travail est ouvert à des perspectives visant à l'enrichir et l'améliorer. De nouvelles voies de recherche peuvent être envisagées, comme la généralisation du nombre de classes de serveurs à $n \in \mathbb{N}$, qui peut se faire à l'aide des RdPSG colorés, la considération de l'hétérogénéité des clients, qui est d'une grande importance pratique, et la prise en compte de la non fiabilité des serveurs, qui est plus proche des systèmes réels où les serveurs sont sujets à des pannes et des réparations aléatoires.

Bibliographie

[1] Aissani, A., and Artalejo, J. On the single server retrial queue subject to breakdowns. *Queueing Systems 30* (1998), 309–321.

[2] Almàsi, B., G.Bolch, and J.Sztrik. Heterogeneous finite-source retrial queues. *Journal of Mathematical Sciences 121* (2004), 2590–2596.

[3] Almàsi, B., Roszik, J., and J.Sztrik. Multiserver retrial queues with finite number of heterogeneous sources. *In Proc. of 6th International Conference on Applied Informatics 2* (2004), 19–26.

[4] Almàsi, B., Roszik, J., and Sztrik, J. Homogeneous finite-source retrial queues with server subject to breakdowns and repairs. *Mathematical and Computer Modelling 42* (2005), 673–682.

[5] Amador, J., and Artalejo, J. R. On the distribution of the successful and blocked events in the M/M/c retrial queue : A computational approach. *Applied Mathematics and Computation 190* (2007), 1612–1626.

[6] Artalejo, J. Retrial queues with a finite number of sources. *J.Korean Math.Soc. 35* (1998), 503–526.

[7] Artalejo, J., and Gómez-Corral, A. *Retrial Queueing Systems : A Computational Approach.* Springer-Verlag, 2008.

[8] Artalejo, J. R. New results in retrial queueing systems with breakdown of the servers. *Statistica Neerlandica 48* (1994), 23–36.

[9] Artalejo, J. R. Accessible bibliography on retrial queues. *Mathematical and Computer Modelling 30*, 3-4 (1999), 1–6.

[10] Artalejo, J. R. A classified bibliography of research on retrial queues : Progress in 1990-1999. *Top 7* (1999), 187–211.

[11] Artalejo, J. R. Accessible bibliography on retrial queues : Progress in 2000-2009. *Mathematical and Computer Modelling* (2010).

[12] Artalejo, J. R., and Pozo, M. Numerical calculation of the stationary distribution of the main multiserver retrial queue. *Annals of Operations Research 116* (2002), 41–56.

[13] Balbo, G. Introduction to Generalized Stochastic Petri Nets. In *SFM* (2007), pp. 83–131.

[14] Baynat, B. *Théorie des files d'attente, des chaînes de Markov aux réseaux à forme produit*. Hermes Science Publications, Paris, 2000.

[15] Begain, K., Barner, J., Bolch, G., and Zreikat, A. I. The performance and reliability modelling language MOSEL and its application. *International Journal of Simulation 3* (2003), 66–80.

[16] Begain, K., Bolch, G., and Herold, H. Practical performance modeling, application of the MOSEL language. *Kluwer Academic Publisher, Boston* (2001).

[17] Bonald, T., and Roberts, J. W. Congestion at flow level and the impact of user behaviour. *Computer Networks 42*, 4 (2003), 521–536.

[18] Brockmeyer, E., Halstrom, H. L., and A.Jensen. *The life and works of A.K. Erlang*. Copenhagen : The Copenhagen Telephone Company, 1948.

[19] Choi, B., and Chang, Y. MAP1,MAP2/M/c retrial queue with the retrial group of finite capacity and geometric loss. *Mathematical and Computer Modeling 30* (1999), 99–114.

[20] Choi, B., and Chang, Y. Single server retrial queues with priority calls. *Mathematical and Computer Modeling 30* (1999), 7–32.

[21] Choi, B. D., Chang, Y., and Kim, B. MAP1/MAP2/M/c retrial queue with guard channels and its applications to cellular networks. *TOP 7*, 2 (1999), 231–248.

[22] Choi, B. D., Shin, Y., and Ahn, W. C. Retrial queues with collision arising from unslotted CMSA/CD protocol. *Queueing Systems 11*, 4 (1992), 335–356.

[23] Clos, C. An aspect of the dialing behaviour of subscribers and its effect on the trunk plant. *Bell Systems Technical Journal 27* (1948), 424–445.

[24] Cohen, J. Basic problems of telephone traffic theory and the influence of repeated calls. *Philips Telecomunication Review 18*, 2 (1957), 49–100.

[25] Cordeiro, J. D., and Major, J. *Unreliable Retrial Queues in a Random Environment*. PhD thesis, Air Force Institute Of Technology, USA, 2007.

[26] Crawford, B. P. Approximate analysis of an unreliable M/M/2 retrial queue. Master's thesis, Air Force Institute Of Technology, USA, 2007.

[27] de Kok, A. G. Algorithmic methods for single server systems with repeated attempts. *Statistica Nederlandica 38* (1984), 23–32.

[28] Deul, N. Stationary conditions for multiserver queueing systems with repeated calls. *Elektronische Informationsverarbeitung und Kybernetik 16* (1980), 607–613.

[29] Diaz, M. *Les réseaux de Petri - Modèles fondamentaux*. Hermès Sciences Publications, Paris, 2001.

[30] Djellab, N. On the M/G/1 retrial queue subjected to breakdowns. *RAIRO-Operations Research 36* (2002), 299–310.

[31] Domenech-Benlloch, M. J., Gimenez-Guzman, J. M., Pla, V., Martinez-Bauset, J., and Casares-Giner, V. Generalized truncated methods for an efficient solution of retrial systems. *Mathematical Problems in Engineering 2008* (2008).

[32] Dragieva, V. I. Single-line queue with finite source and repeated calls. *Problems of Information Transmission 30* (1994), 283–289.

[33] Efrosinin, D., and Breuer, L. Threshold policies for controlled retrial queues with heterogeneous servers. *Annals of Operations Research 141* (2006), 139–162.

[34] Erlang, A. K. Solution of some problems in the theory of probabilities of significance in automatic telephone exchanges. *Elektrotkeknikeren 13* (1917).

[35] Falin, G. The influence of inhomogeneity of the composition of subscribers on the functioning of telephone systems with repeated calls. *Eng. Cybernet. Rev. 21* (1983), 21–25.

[36] Falin, G. A survey of retrial queues. *Queueing Systems 7*, 2 (1990), 127–167.

[37] Falin, G., and Artalejo, J. A finite source retrial queue. *European Journal of Operational Research 108*, 3 (1998), 409–424.

[38] Falin, G., and Templeton, J. *Retrial Queues*. Chapman et Hall, 1997.

[39] Falin, G. I. Calculation of probability characteristics of a multiline system with repeat calls. *Moscow University Computational Mathematics and Cybernetics*, 1 (1983), 43–49.

[40] Falin, G. I. A multiserver retrial queue with a finite number of sources of primary calls. *Mathematical and Computer Modelling 30* (1999), 33–49.

[41] Fredericks, A. A., and Reisner, G. A. Approximations to stochastic service systems,with an application to a retrial model. *The Bell System Technical Journal 58*, 3 (1979), 557–576.

[42] Gharbi, N. *Évaluation des performances et de la fiabilité des systèmes multiclasses avec rappel à l'aide des réseaux de Petri stochastiques colorés*. PhD thesis, Université des Sciences et de la Technologie Houari Boumediene, Bab Ezzouar, Alger, Algérie, 2007.

[43] Gharbi, N., Dutheillet, C., and Ioualalen, M. Colored stochastic Petri nets for modelling and analysis of multiclass retrial systems. *Mathematical and Computer Modelling 49* (2009), 1436–1448.

[44] Gharbi, N., and Ioualalen, M. GSPN analysis of retrial systems with servers breakdowns and repairs. *Applied Mathematics and Computation 174* (2006), 1151–1168.

[45] Gomez-Corral, A. Stochastic analysis of a single server retrial queue with general retrial times. *Naval Research Logistics 46* (1999), 561–581.

[46] Greenberg, B. S., and Wolf, R. W. An upper bound on the performance of queues with returning customers. *Journal of Applied Probability 24*, 2 (1987), 466–475.

[47] Hiraishi, K. Performance Evaluation of Workflows Using Continuous Petri Nets with Interval Firing Speeds. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci. E91-A*, 11 (2008), 3219–3228.

[48] Houck, D. J., and Lai, W. S. Traffic modeling and analysis of hybrid fiber-coax systems. *Comput. Netw. ISDN Syst. 30*, 8 (1998), 821–834.

[49] Howard, R. *Dynamic Programming and Markov Processes.* Wiley, 1960.

[50] Jain, M., and A.Mishra. Reliability analysis of unreliable server retrial queue with bulk arrivals. *Pakistan Journal of Statistics 24* (2008), 285–300.

[51] Janssens, G. The quasi-random input queueing system with repeated attempts as a model for collision-avoidance star local area network. *IEEE Transactions on Communications 45* (1997), 360–364.

[52] Jonin, G., and Sedol, J. Telephone systems with repeated calls. *in Proceedings of the 6th International Teletraffic Congress (ITC '70), Munich, Germany* (1970), 435.1–435.5.

[53] J.Wang. On the single server retrial queue with priority subscribers and server breakdowns. *Journal of Systems Science and Complexity 21* (2008), 304–315.

[54] Kalmychkov, A., and Medvedev, G. Probability characteristics of Markov local area networks with random-access protocols. *Automatic Control and Computer Science 24* (1990), 38–45.

[55] Kernane, T. Conditions for stability and instability of retrial queueing systems with general retrial times. *Statistics and Probability Letters 78* (2008), 3244–3248.

[56] Kernane, T., and Aïssani, A. Stability of retrial queues with versatile retrial policy. *Journal of Applied Mathematics and Stochastic Analysis* (2006).

[57] Khansa, W. *Réseaux de Petri P-temporels : contribution à l'étude des systèmes à évènements discrets.* PhD thesis, Université de Savoie, 1997.

[58] Khomichkov, I. Study of models of local networks with multiple-access protocols. *Automation and Remote Control 54* (1993), 1801–1811.

[59] Kleinrock, L. *Queueing Systems. Volume 1 : Theory.* John Wiley, New York, 1975.

[60] Kornyshev, Y. N. Design of a fully accessible switching system with repeated calls. *Telecommunications 23* (1969), 46–52.

[61] Kornyshev, Y. N. A system with repeated calls and finite number of sources. *Elektrusyaz (en russe) 5* (1977), 58–62.

[62] Kosten, L. On the influence of repeated calls in the theory of probabilities of blocking. *De Ingenieur 59* (1947), 1.

[63] Kosten, L. Stochastic theory of service systems. *Pergamon Press, Oxford 103* (1973).

[64] Kulkarni, V. On queueing systems with retrials. *J. Appl. Probab. 20* (1983), 380–389.

[65] LADET, P. Réseaux de Petri. *Techniques de l'Ingénieur, traité Informatique industrielle* (1989).

[66] Li, H., and Yang, T. A single-server retrial queue with server vacations and a finite number of input sources. *European Journal of Gperational Research 85* (1995), 149–160.

[67] Lipschutz, S. *Probabilités. Cours et problèmes.* Mac Graw Hill, 1973.

[68] Mandelbaum, A. Call centers (centres), research bibliography with abstracts, tech. rep. *Faculty of Industrial Engineering and Management Technion-Israel Institute of Technology, Technion City, Israel* (2004).

[69] Marsan, M. A., Balbo, G., and Conte, G. *Performance models of multiprocessor systems.* MIT Press, Cambridge, MA, USA, 1987.

[70] Marsan, M. A., Balbo, G., and Conte, G. A Class of Generalized Stochastic Petri Nets for the Performance Analysis of Multiprocessor Systems. *ACM Transactions on Computer Systems 2*, 1 (May 1984), 93–122.

[71] Marsan, M. A., Balbo, G., Conte, G., Donatelli, S., and Franceschinis, G. *Modelling with Generalized Stochastic Petri Nets.* John Wiley and Sons, 1995.

[72] Marsan, M. A., Carolis, G. D., Leonardi, E., Cigno, R. L., and Meo, M. Efficient estimation of call blocking probabilities in cellular mobile tele-

phony networks with customer retrials. *IEEE Journal on Selected Areas in Communications 19*, 2 (2001), 332–346.

[73] Molloy, M. *On the Integration of Delay and Throughput Measures in Distributed Processing Models.* PhD thesis, University of California, Los Angeles, 1981.

[74] Molloy, M. K. Performance Analysis Using Stochastic Petri Nets. *IEEE Transaction on Computers 31*, 9 (1982), 913–917.

[75] Neuts, M. F. Matrix-geometric solutions in stochastic models : An algorithmic approach. *Johns Hopkins Series in the Mathematical Sciences, Johns Hopkins University, Baltimore, Md, USA 2* (1981).

[76] Neuts, M. F., and Rao, B. M. Numerical investigation of a multiserver retrial model. *Queueing Systems 7*, 2 (1990), 169–189.

[77] Ohmura, H., and Takahashi, Y. An analysis of repeated call model with a finite number of sources. *Electronics and Communications in Japan 68* (1985), 112–121.

[78] Onur, E., Deliç, H., Ersoy, C., and çaglayan, M. U. Measurement-based replanning of cell capacities in GSM networks. *Computer Networks 39*, 6 (2002), 749–767.

[79] Pardoux, E. *Processus de Markov et applications : Algorithmes, réseaux, génome et finance, cours et exercices corrigés.* DUNOD, 2007.

[80] Pearce, C. Extended continued fractions, recurrence relations and two-dimensional Markov processes. *Advances in Applied Probability 21* (1989), 357–375.

[81] Petri, C. A. *Kommunikation mit Automaten.* PhD thesis, University of Bonn, West Germany, 1962.

[82] Pourbabai, B. Asymptotic analysis of G/G/K queueing-loss system with retrials and heterogeneous servers. *International Journal of Systems Science.*

[83] Pourbabai, B. Analysis of a G/M/K/O queueing loss system with heterogeneous servers and retrials. *International Journal of Systems Science 18* (1987), 985–992.

[84] Pourbabai, B. Markovian queueing systems with retrials and heterogeneous servers. *Computers and Mathematics with Applications 13* (1987), 917–923.

[85] Riordan, J. Stochastic service systems. *Wiley, New York* (1962).

[86] Roszik, J., and J.Sztrik. Performance analysis of finite-source retrial queueing systems with heterogeneous non-reliable servers and different service policies. *Institute of Informatics, Univ. Debrecen 6* (2004).

[87] Roszik, J., and J.Sztrik. Performance analysis of finite-source retrial queues with nonreliable heterogeneous servers. *Journal of Mathematical Sciences 146* (2007), 6033–6038.

[88] Sherman, N., and Kharoufeh, J. An M/M/1 retrial queue with unreliable server. *Operations Research Letters 34* (2006), 697–705.

[89] Stepanov, S. N. Markov models with retrials : the calculation of stationary performance measures based on the concept of truncation. *Mathematical and Computer Modelling 30*, 3-4 (1999), 207–228.

[90] Sztrik, J., Almàsi, B., and Roszik, J. Heterogeneous finite-source retrial queues with server subject to breakdowns and repairs. *Journal of Mathematical Sciences 132*, 5 (2006), 677–685.

[91] Tran-Gia, P., and Mandjes, M. Modeling of customer retrial phenomenon in cellular mobile networks. *IEEE Journal on Selected Areas in Communications 15*, 8 (1997), 1406–1414.

[92] Wang, J. Reliability analysis of M/G/1 queues with general retrial times and server breakdowns. *Progress in Natural Science 16* (2006), 464–473.

[93] Wang, J., Cao, J., and Li, Q. Reliability analysis of the retrial queue with server breakdowns and repairs. *Queueing Systems 38* (2001), 363–380.

[94] Wilkinson, R. I. Theories for toll traffic engineering in the USA. *The Bell System Technical Journal 35*, 2 (1956), 421–514.

[95] Wolf, R. *Stochastic modeling and the theory of queues*. Prentice-Hall, 1989.

[96] Yang, T., Posner, M. J. M., Templeton, C., J. G., and Li, H. An approximation method for the M/G/1 retrial queue with general retrial times. *European Journal of Operational Research 76*, 3 (August 1994), 552–562.

[97] Yang, T., and Templeton, J. G. C. A survey on retrial queues. *Queueing Systems 2* (1987), 203–233.

Résumé

Les systèmes avec rappel sont des systèmes dans lesquels les clients qui trouvent tous les serveurs occupés ou non disponibles, rappellent ultérieurement pour le service, à des intervalles de temps aléatoires. Cependant, la prise en considération du phénomène d'appels répétés a introduit de grandes difficultés analytiques. En fait, des résultats explicites détaillés existent pour certaines files d'attente avec rappel particulières, avec des hypothèses contraignantes sur certains paramètres, tel que la taille de la source de clients, le nombre de serveurs, l'homogénéité des serveurs, etc.

Il est important de préciser que les études des modèles avec rappel et serveurs hétérogènes restent à ce jour très rares et les résultats obtenus sont assez limités. À cet effet, le but du sujet est la proposition d'une approche numérique permettant l'analyse des performances des systèmes avec rappel à source finie de clients et deux classes de serveurs hétérogènes, en utilisant le modèle des réseaux de Petri stochastiques généralisés. Ce formalisme de haut niveau permet une description simple du comportement des systèmes complexes avec phénomène de rappel. Par ailleurs, il offre un bon moyen de génération automatique de la chaîne de Markov correspondante, pour l'analyse des performances. Cependant, la génération et la résolution de la chaîne de Markov déduite, nécessite souvent un espace mémoire très large et un temps d'exécution très long, puisque la taille de l'espace d'états croît exponentiellement en fonction de la taille de la source de clients et du nombre de serveurs. Ainsi, nous nous intéressons dans ce projet à la conception et l'implémentation d'une approche numérique permettant l'obtention automatique du générateur infinitésimal et le calcul des indices de performances à l'état stationnaire sans avoir à générer ni le graphe d'accessibilité ni la chaîne de Markov, et ceci, en considérant deux disciplines de service : la discipline du service aléatoire, et celle du service le plus rapide.

Mots clés : Systèmes avec rappel, Source finie de clients, Serveurs hétérogènes, Réseaux de Petri stochastiques généralisés, Générateur infinitésimal, Indices de performance.