REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

Ministère de l'enseignement supérieur et de la recherche scientifique

Université Saad Dahleb Blida 1

Mémoire de fin d'étude

En vue d'obtenir le diplôme de Master en informatique

Spécialité : Système d'informatique et Réseaux

Thème :

Conception et Réalisation d'un simulateur pour l'analyse des performances des systèmes de files d'attentes avec vacance

Devant le jury composé de:

M^me^ Boutoumi Bachira Promotrice.

M^me^ Toubaline Nesrine Présidente.

M^me^ Nasri Ahlem Examinatrice.

Présenté par :

M^me^ Aggoun Amina.

M^elle^ Berrak Khawla.

Année universitaire 2020/2021

Remerciement

Nous tenons à remercier Allah , qui nous a donné la force , la capacité et la patience durant ces longues années d'étude et qui nous a donné le courage pour accomplir ce travail.

Nous tenons à remercier notre promotrice Mme : Boutoumi Bachira pour son encadrement et ses conseils durant la réalisation de ce travail

Nos sincères remerciements vont à tous nos enseignants pour leurs disponibilités et leurs soutien ,tout particulièrement Mr Oueld khaoua notre responsable de spécialité.

Nous remercions infiniment nos chers parents qui nous ont soutenu et encouragé tout au long de nos études , nos frères , nos sœurs , mon mari khaled (Amina) qui m'a aidé et encouragé pour achever ce travail , tous les membres de nos familles pour leur contribution et leur soutien .

Nous remercions nos amies pour leur soutien et encouragement

Nous tenons à remercie toute personne qui a participé de près ou de loin à l'exécution de ce modeste travail.

Résumé

Les files d'attente apparaissent naturellement dans beaucoup de situations de la vie courante. Les files d'attente avec vacance sont des files d'attente dans lesquels le serveur prend des vacances d'une tâche primaire pour s'occuper d'une ou plusieurs tâches secondaires ou pour se reposer ou pour accomplir quelques tâches de maintenance préventive, pendant ce temps les clients attendent d'être servis. Cette classe de files d'attente peut se trouver dans plusieurs applications.

Cependant, d'un point de vue technique, la considération du phénomène de vacance des serveurs 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 files d'attentes avec vacance.

Afin d'analyse des performances dans ce vaste domaine qui est l'objectif de notre travail nous allons procéder à travailler avec des modèles de files d'attente avec vacances en incorporant les politiques de vacances, nous allons développer un outil de simulation basé sur les techniques de simulation à événement discret afin d'évaluer les performances des systèmes de files d'attentes avec vacance.

Mots clés:
File d'attente avec vacance, Simulation à temps discret, Disciplines de vacance, Evaluation des performances, GUI.

Abstract

Queues naturally appear in many everyday situations. Vacant queues are queues in which the server takes a vacation from a primary task to take care of one or more secondary tasks or to rest or perform some preventive maintenance tasks, Meanwhile customers are waiting to be served. This queue class can be found in multiple applications.

However, from a technical point of view, the consideration of the phenomenon of server vacancy has caused a lot of analytical difficulties. This makes the conventional methods and results obtained in the classical queue theory inadequate. This explains the researchers' use of approximation and simulation techniques and numerical algorithms in the development of performance indices in the field of vacancy queues.

In order to analyze the performance in this vast area which is the objective of our work we will proceed to work with models of queues with vacation by incorporating holiday policies, we will develop a simulation tool based on discrete event simulation techniques to evaluate the performance of vacancy queue systems.

Key words:

Queue with vacancy, Discrete time simulation, Vacancy disciplines, Performance evaluation, GUI.

Table des matières

Liste des figures et Pseudo algorithmes

Liste des tableaux

Liste des abréviations

FCFS : First come first served.

FEL: Future Event List.

DES : Discrète évent simulation.

Introduction générale

De nombreux domaines utilisent des observations en fonction du temps (ou plus rarement, d'une variable d'espace). Dans les cas les plus simples, ces observations se traduisent par une courbe bien définie. En réalité, des sciences de la Terre aux sciences humaines, les observations se présentent souvent de manière plus ou moins erratique. L'interprétation de ces observations est donc soumise à une certaine incertitude qui peut être traduite par l'utilisation des probabilités pour les représenter.

Le phénomène d'attente est observé dans notre vie quotidienne. Quand nous nous rendons à la poste, à la gare, à l'hôpital, bien souvent nous devons faire une *« Queue »*pour obtenir de l'argent, un billet, une consultation. Ce phénomène d'attente se forme lorsque le client arrive de façon aléatoire, non aléatoire ou déterministe pour se faire servir. La modélisation de ce dernier a pour but de calculer ses performances et de déterminer ses caractéristiques afin d'aider les gestionnaires à prendre des décisions .Cependant, l'étude de la théorie des files d'attente a seulement commencé au début de vingtième siècle avec le travail de mathématicien Danois A.K.Erlang.il a réalisé la première analyse mathématique d'un modèle de file d'attente des systèmes téléphoniques dans un centre d'appel à Copenhague.

L'évolution rapide de cette théorie a été appliquée dans divers domaines, les systèmes informatiques et réseaux de communications cité dans [1].Nous trouvons aussi leurs applications dans les flux de trafics (voiture, avion), l'ordonnancement (planification, programme d'un ordinateur), ou encore le dimensionnement (banque, poste, téléphone).

De nombreux systèmes de files d'attente du monde réel peuvent être modélisés en tant que modèle de vacances avec différente Stratégies. La notion de files d'attente avec vacance, ou bien les systèmes de files d'attente avec vacance du serveur (Queuing vacation ou Queuing Systems with server vacation, en anglais), sont une extension des files d'attente classiques ou les serveurs sont toujours disponible. Cependant, dans les systèmes de files d'attentes avec vacance, les serveurs peuvent devenir non disponibles pour une période de temps. Durant cette période d'absence, le serveur prend des vacances d'une tâche primaire pour s'occuper d'une ou plusieurs tâches secondaires ou pour se reposer.

Motivé par une masse importante d'applications, Les files d'attente avec vacance ont été largement étudiés. Un nombre important de travaux ont été réalisées au début des années 80 dans ce domaine cité dans [2], [3], [4], [5], de plus les modèles de vacances de serveur se présentent comme étant l'image plus réaliste et plus flexible des files d'attente réelles.

Parmi les techniques de modélisation du monde réel, c'est la simulation qui est constitué un outil pratique pour étudier des systèmes complexes qui ne peuvent pas être modélisés avec précision pour une analyse mathématique exacte ou approximative. Un scénario de simulation pour un système complexe peut être mis en place avec autant de détails que nécessaire - essentiellement autant de détails qu'il serait possible de traiter dans les limites du temps de simulation qui peut être passé et la complexité de la simulation pour laquelle on est prêt à

coder. Les simulations sont également utiles pour effectuer des contrôles croisés des résultats obtenus par analyse.

Dans ce mémoire, nous nous sommes intéressé a travailler dans un cadre plus pratique, ce dernier consiste à modéliser et développé un outil de simulation permettant de présenter l' approche de simulation à évènement discret ainsi que les modèles utilisé avec les déférentes stratégies de vacances adoptées, de plus nous avons développé les principaux indices de performances liée à ce modèle.

Pour atteindre cet objectif, nous allons planifier la suite de ce rapport en cinq chapitres comme suit :

- ✓ Le premier chapitre est réservé sur la définition des variables aléatoires, le processus stochastique, processus markoviens, les chaines de Markov, processus de poisson et le processus de naissance et de mort.
- ✓ Le deuxième chapitre est consacré à la description des modèles de files classique et files d'attente avec vacance en définissant leurs caractéristiques.
- ✓ Dans le troisième chapitre, on définira les concepts de la simulation à événement discret ainsi que leurs différentes approches une section qui sera consacré à présenter quelques travaux connexes.
- ✓ Le quatrième chapitre est consacré à la conception de l'outil de simulation que nous allons réaliser son fonctionnement.
- ✓ Le cinquième et le dernier chapitre qui montre l'implémentation de notre outil de simulation, y compris les résultats de la simulation et une étude expérimentale.

Enfin on conclure notre rapport avec une conclusion générale.

Chapitre 1

1. Rappel des probabilités

1.1 Introduction

L'étude de tels systèmes d'attente nous conduit à étudier les processus stochastiques. Ces processus permettent de modéliser des systèmes dont le comportement n'est que partiellement prévisible. De plus Les chaînes de Markov sont un outil fondamental pour modéliser les processus en théorie des files d'attente et en statistiques.

Nous récapitulons ici brièvement quelques concepts importants en statistiques et en probabilités, ainsi nous allons introduire les concepts de processus stochastique, le processus de markovien, quelques notions des chaines de Markov, le processus de Poisson et le processus de naissance et de mort qui joue un rôle pour l'analyse mathématique de ces modèles. Il est en effet nécessaire de disposer d'un minimum de connaissances dans ce domaine. Nous terminons le chapitre par une conclusion.

1.2 Variables aléatoires

Une expérience aléatoire est une action ou une observation dont le résultat est incertain préalablement à sa réalisation. Un espace de probabilité est un triplet $(\Omega, \mathcal{F}, P)$ où Ω, l'espace d'échantillonnage est l'ensemble de toutes les réalisations possibles de l'expérience appelés éléments aléatoires, individuellement dénotés par ω. La collection $\mathcal{F}$ des événements aléatoires est un σ-champ des sous-ensembles de Ω. Nous dirons que l'événement $\mathrm{A} \in \mathcal{F}$ est réalisé si le résultat de l'expérience est un élément de A. Une mesure P est assignée aux éléments de $\mathcal{F}$, $P(A)$ étant la probabilité de A.

Une mesure de probabilité sur un espace mesurable $(\Omega, \mathcal{F})$ est une fonction d'ensemble $P : \mathcal{F} \rightarrow [0,1]$ satisfaisant les axiomes de probabilité [6] :

- $P[A] \geq 0$, pour tout $\mathrm{A} \in \mathcal{F}$;
- $P[\Omega] = 1$;
- Additivité dénombrable : pour une collection disjointe$\{\mathrm{A_j} \in \mathcal{F} \; ; \; j \in \mathbb{N}\}$,

$$P\left[U_j \mathrm{A}_j\right] = \sum_j \mathrm{P}\left[\mathrm{A_j}\right].$$

Les systèmes de file d'attente fournissent de nombreux exemples de variables aléatoires soit discrètes ou continuent par exemple ils sont utilisées : pour représenter le nombre de clients dans le système au moment t, pour générer le temps de service d'un client ou le temps d'inter arrivée des clients …etc. [7]

Les deux types de variables aléatoires sont :

- **Variables discrètes**

Une variable aléatoire X est dite discrète si elle peut prendre au plus un nombre dénombrable de valeurs$x_1, x_2, \ldots$ .La probabilité que la variable aléatoire X prenne la valeur x_i est donnée par :

$$p(x_i) = P[X = x_i] \ , \text{ pour } i = 1, 2, \ldots$$

$p(x)$ Est appelée fonction de masse de probabilité, nous avons :

$$\sum_{i=1}^{x=\infty} p(x_i) = 1.$$

- **Variables continues**

Une variable aléatoire X est dite continue s'il existe une fonction non négative.

$F(x)$ Telle que pour n'importe quel ensemble de nombres réels B:

$$\mathrm{P}\,[X \in B] = \int_B f(x)dx \ \text{ et } \int_{-\infty}^{\infty} f(x)dx = 1.$$

$F(x)$ Est appelée fonction de densité de probabilité de la variable aléatoire continue X. [6]

1.3 Processus stochastiques

L'étude mathématique d'un système d'attente, se fait le plus souvent par l'introduction d'un processus stochastique approprie. Généralement, on qualifie de processus stochastique tout phénomène d'évolution temporelle dont l'analyse peut être soumise au calcul des probabilités. Du point de vue de l'observation, un processus stochastique est constitué par l'ensemble de ses réalisations. Une réalisation est obtenue par l'expérience qui consiste à enregistrer une suite d'événements au cours du temps. Le caractère aléatoire de l'évolution se montre par le fait que la répétition de l'expérience conduit à une autre séquence temporelle.[8]

Un processus stochastique $\{X_t \ , t \in T\}$ est une famille de variables aléatoire indexées par le temps t et définies sur un même espace de probabilité (Ω, A, P). Les variables aléatoires prennent des valeurs dans un espace E dénombrable ou non. Les processus stochastiques peuvent être classifiés en se basant sur l'espace d'état E et l'indice t désignant le temps, t ϵ T.

- Si T est infini dénombrable, par exemple $T = N$, alors $X_t \ , t \in T$ est un processus à temps discret.
- Si T est un ensemble d'intervalles par exemple $T = R^+$, alors $X_t \ , t \in T$ est un processus à temps continu. [9]

1.4 Processus markoviens

Le processus de Markov fournit un outil simple de modélisation d'une classe particulière de systèmes à espace d'états discret. L'analyse des processus de Markov est un préliminaire nécessaire à l'étude des systèmes de files d'attente.

Le processus $\{X_t , t \geq 0\}$ est dit de Markov si pour tous les $t_1 < t_2 < t_3 < \ldots < t_n < t_{n+1}$, pour tous $x_1 , x_2 , x_3 , \ldots , x_{n+1}$

$$P\left[X_{t_{n+1}} = x_{t_{n+1}} \middle| X_{t_1} = x_{t_1}, X_{t_2} = x_{t_2}, \ldots, X_{t_n} = x_{t_n}\right] = P\left[X_{t_{n+1}} = x_{t_{n+1}} \middle| X_{t_n} = x_{t_n}\right].$$

Cela traduit que la probabilité de n'importe quel comportement futur, le présent étant connu, n'est pas modifié par toute connaissance supplémentaire du passé, ce qui caractérise la propriété de d'absence de mémoire. [9]

1.4.1 Les chaînes de Markov discrètes(CMTD)

Une chaîne de Markov à temps discret $\{X_n, \mathrm{n} = 0, 1, 2, \ldots\}$ est un processus stochastique qui satisfait la relation suivante, appelée la propriété de Markov : Pour tous les nombres naturels *n* et tous les étatsX_n.

$$\mathrm{Prob}\{X_{n+1} = x_{n+1} | X_n = x_n, X_{n-1} = x_{n-1}, \ldots, X_0 = x_0\}$$

$$= \mathrm{Prob}\{X_{n+1} = x_{n+1} | X_n = x_n\}.$$

Ils donnent la probabilité conditionnelle d'effectuer une transition de l'état $x_n = i$ à l'état $x_{n+1} = j$ lorsque le paramètre temps augmente de n à $n + 1$. Ils sont dénotés par :

$$P_{ij}(n) = Prob\{X_{n+1} = j | X_n = i\}.$$

La matrice *P(n),* formée en plaçant$P_{ij}(n)$ dans la ligne *i* et la colonne j, pour tous les *i* et *j,* est appelée matrice de probabilité de transition ou matrice de chaîne.

$$P(n) = \begin{pmatrix} p_{00}(n) & p_{01}(n) & p_{02}(n) & \cdots & p_{0j}(n) & \cdots \\ p_{10}(n) & p_{11}(n) & p_{12}(n) & \cdots & p_{1j}(n) & \cdots \\ p_{20}(n) & p_{21}(n) & p_{22}(n) & \cdots & p_{2j}(n) & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ p_{i0}(n) & p_{i1}(n) & p_{i2}(n) & \cdots & p_{ij}(n) & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{pmatrix}.$$

Figure1.1 : Matrice *P(n).*

Notez que les éléments de la matrice *P(n)* satisfont aux deux propriétés suivantes :

$$0 \leq P_{ij}(n) \leq 1$$

Et pour tout ce que i :

$$\sum_{tous\ j} P_{ij} = 1\ .$$

Une matrice qui satisfait ces propriétés est appelée matrice de Markov ou matrice stochastique.

Une chaîne de Markov est dit être homogène si pour tous les états i et j et pour tous $n = 0, 1, 2,...$ et $m \geq 0$ on a :

$$\text{Prob}\{X_{n+1} = j | X_n = i\} = \text{Prob}\{X_{n+m+1} = j | X_{n+m} = i\}.$$

Comme nous venons de le voir, dans une chaîne homogène de Markov à temps discret, les probabilités de transition $P_{ij}(n) = \text{Prob}\{X_{n+1} = j | X_n = i\}$ sont indépendantes de n et sont par conséquent écrites comme suit [10] :

$$P_{ij} = \text{Prob}\{X_{n+1} = j | X_n = i\}, pour\ tous\ n = 0,1,2, ...$$

1.4.2 Les chaînes de Markov continues(CMTC)

Processus stochastique $\{X(t), t \geq 0\}$ est une chaîne de Markov continue si pour tous les états n, et pour toute séquence $t_0, t_1, \dots, t_n, t_{n+1}$ telle que $t_0 < t_1 < \cdots < t_n < t_{n+1}$,

$$\text{Prob}\{\text{X}(t_{n+1}) = x_{n+1} | \text{X}(t_n) = x_n, \text{X}(t_{n-1}) = x_{n-1}, \dots, \text{X}(t_0) = x_0\}$$

$$= \text{Prob}\{\text{X}(t_{n+1}) = x_{n+1} | \text{X}(t_n) = x_n\}.$$

Où $X(t)$ indique l'état de la chaîne de Markov au temps t s. Lorsque la chaîne de Markov en temps continu est homogène, ces probabilités de transition dépendent de la différence $\tau - s$ plutôt que des valeurs réelles de s et t. Dans ce cas, nous simplifions la notation en écrivant :

$$P_{ij}(\tau) = Prob\{X(s + \tau) = j | X(s) = i\}\ pour\ tous\ \ s \geq 0$$

Ceci indique la probabilité d'être dans l'état j après un intervalle de longueur τ, étant donné que l'état courant est l'état i. Il dépend de la longueur τ mais pas sur s, le moment spécifique auquel cet intervalle de temps commence. Il s'ensuit que [10]:

$$\sum_j p_{ij}\ (\tau) = 1\ pour\ tous\ les\ valeurs\ de\ \tau$$

1.5 Processus de Poisson

Dans le formalisme des files d'attente, les arrivées des clients sont caractérisées par l'ensemble des instants ou dates d'arrivée de chaque client. Cette collection des dates d'arrivée s'appelle le processus des arrivées. Lorsque les dates d'arrivées sont imprévisibles, elles sont modélisées par des variables aléatoires (processus de Poisson).

Le processus $(N_t)\ t \in R^+$est appelé processus de comptage si c'est un processus croissant, c'est-à-dire si pour tout $s \leq t$, $N_t \leq N_s$. La variables aléatoire $N_t - N_s$ est alors appelée accroissement du processus sur] s, t].

Un processus de comptage $(N_t)\ t \in R^+$est appelé processus à accoisements indépendants si pour tout $n \in N^*$ et pour tous $t_1, t_2, \ldots, t_n$ tels que $t_1 < t_2 < \ldots < t_n$, les accroissements $N_{t1} - N_{t0}$, $N_{t2} - N_{t1}, \ldots, N_{tn} - N_{tn-1}$ sont des variables aléatoires indépendantes .

Le processus est dit stationnaire, si pour tout s et t, l'accroissement $N_{t+s} - N_s$ a la même loi que N_t .

Un processus à accroissements indépendants stationnaires $(N_t)\ t \in R^+$ est dit à événements rares si :

$$\lim_{h \to 0+} P[N_h > 0] = 0 \ et \ si \lim_{h \to 0+} \frac{[N_h > 1]}{[N_h = 1]} = 0.$$

Le processus de comptage $(N_t)\ t \in R^+$ tel que $N_0 = 0$ est un processus de poisson si :

1. $\{N_t, t \geq 0\}$ est stationnaire.
2. $\{N_t, t \geq 0\}$ est un processus à accroissements indépendants.
3. $\{N_t, t \geq 0\}$ est processus à événements rares.
4. Il existe $\lambda > 0$ tel que, pour tous $t \geq 0$, la variable aléatoire N_t suit la loi de poisson de paramètre λt.

Remarque : un processus qui vérifier la 4éme condition vérifier certainement la 3éme condition. **[**11]

1.6 Processus de naissance-mort

Le processus naissance-mort qui est aussi une généralisation du processus de Poisson. Que l'espace d'état soit $\{0, 1, 2, \ldots\}$ qui pourrait être fini. Dans le processus naissance-mort, la transition se fait de l'état $k (k > 0)$ soit à l'état $k + 1$ (c'est-à-dire qu'une naissance se

produit) au taux λk ou à l'état $k-1$ (c'est-à-dire qu'un décès se produit) au taux μk et les naissances et les décès se produisent indépendamment les uns des autres. Lorsque le processus est à 0, seule une naissance peut se produire au taux λ_0. Si aucun décès ne se produit (c'est-à-dire, $\mu k = 0$), alors le processus est appelé un processus de naissance pure et si en plus le taux de natalité est constant (c'est-à-dire : $\lambda_k = \lambda$), le processus devient un processus de Poisson.

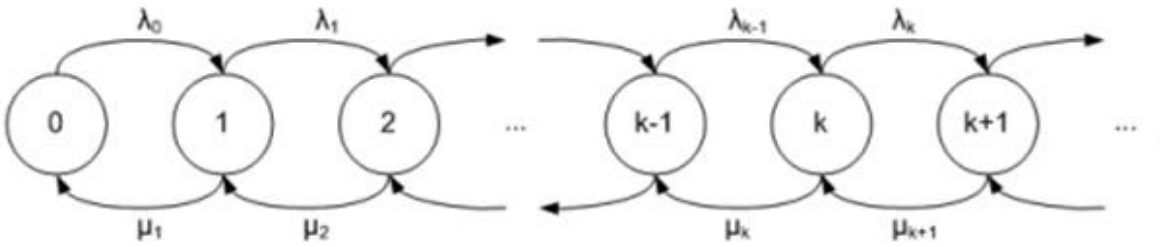

Figure1.2 : Diagramme de transition d'état de processus naissance-mort.

On peut nous rappeler que le processus de naissance-mort a la propriété de Markov. Pour ce processus, nous avons

$$P(X(t+\Delta t) = k+1 \mid X(t) = k) = \lambda_k \Delta t + o(\Delta t),$$

$$P(X(t+\Delta t) \geq k+2 \mid X(t) = k) = o(\Delta t),$$

$$P(X(t+\Delta t) = k-1 \mid X(t) = k) = \mu_k \Delta t + o(\Delta t),$$

$$P(X(t+\Delta t) \leq k-2 \mid X(t) = k) = o(\Delta t).$$

De nombreux systèmes de mise en file d'attente sont modélisés comme processus naissance-mort en représentant simplement une arrivée par une naissance et un départ par une mort et en assumant la propriété de Markov pour le processus. [9]

1.7 Conclusion

Dans ce chapitre nous avons introduit les modèles mathématiques nécessaires pour l'étude du formalisme des files d'attente telles que les variables aléatoires, processus stochastique, processus markovien, quelques notions de bases des chaines de Markov, processus de poisson, et le processus de naissance et de mort. Ceci aidera le lecteur à mieux assimiler la problématique abordée dans la suite de ce mémoire.

Dans le prochain chapitre, nous nous intéressons à présenter les files d'attente classiques avec vacances et leurs caractéristiques.

Chapitre 2

2 Les files d'attente classique et avec vacances

2.1 Introduction

La théorie des files d'attente est une technique de la Recherche opérationnelle qui permet de modéliser un système admettant un phénomène d'attente, de calculer ses performances et de déterminer ses caractéristiques pour aider les gestionnaires dans leurs prises de décisions.

Les files d'attente peuvent être considérées comme un phénomène caractéristique de la vie contemporaine. On les rencontre dans de très nombreux domaines et sous diverses formes. Citons quelques exemples parmi tant d'autres : queue à un guichet, saturation d'un trafic routier, d'un réseau de télécommunications, gestion d'un stock de production, maintenance d'un équipement informatique, mouvements de populations, prévisions météorologiques, etc. L'étude mathématique des phénomènes d'attente constitue un champ d'application important des processus stochastiques.

On parle de phénomène d'attente chaque fois que certaines unités appelées "clients" se présentent d'une manière aléatoire à des "stations" afin de recevoir un service dont la durée est généralement aléatoire.

Il n'y a pas de théorie des files d'attente en tant que tel. La théorie des files d'attente est un formalisme mathématique qui permet de mener des analyses quantitatives à partir de la donnée des caractéristiques du flux d'arrivées et des temps de service. Dans certaines situations, le service devient temporairement non disponible aux clients. Un tel système est dit système d'attente avec vacance.

Au cours de ce chapitre nous allons présenter des modèles de files d'attente simple et les files d'attente avec vacance et leurs caractéristiques et nous allons introduire les mesures de performance approprie.

2.2 Les files d'attente

2.2.1 Développement de la théorie des files d'attentes

Les origines du formalisme des files d'attente datent du début du XXème siècle et principalement des travaux de deux mathématiciens : le mathématicien danois A.K.Erlang avec ses travaux sur les réseaux téléphoniques et le russe A.A. Markov avec la création des modèles markoviens [12].

C'est en 1909 que les bases de ce formalisme sont jetées, grâce à l'article du mathématicien danois A.K. Erlang „"The theory of probabilities and telephone conversations". Les premiers résultats sont variés : Erlang observe le caractère poissonnier des arrivées des appels à un central téléphonique, et le caractère exponentiel des durées des appels; il réussit à calculer de

manière relativement simple la probabilité d'avoir un appel rejeté. La notion d'équilibre stationnaire d'un système d'attente est introduite.

À partir des années 30, les travaux de plusieurs mathématiciens tels que Molina, Fry, Pollaczek aux États-Unis, Kolmogorov et Khintchine en Russie, Palm en Suède, ou Crommelin en France permettent à la théorie des files d'attente de se développer lentement. Ce sont ensuite les années 50 qui verront l'essor important de la théorie.

Les applications de ces travaux sont alors très pratiques, et concernent les disciplines de recherche opérationnelle et génie industriel. On peut citer les flux de trafic (véhicule, avion, personnes, télécommunications), l'ordonnancement, c'est-à-dire la planification, par exemple les patients dans les hôpitaux, les programmes d'un ordinateur, etc . . .ou encore le dimensionnement (banque, poste, réseaux, téléphone, ordinateur).

Dans les années 80, cette discipline devient beaucoup plus mathématique, et la littérature regorge d'articles décrivant des techniques ou des astuces mathématiques permettant de trouver des solutions exactes aux modèles.

2.2.2 Définitions

File d'attente : l'ensemble des clients qui attendent d'être servis, à l'exclusion de celui qui est en train de se faire servir.

Système d'attente : l'ensemble des clients qui font la queue, y compris celui qui se fait servir. [13]

File d'attente simple : est un système composé d'un espace d'attente, d'un ou plusieurs serveurs et des clients qui arrivent, attendent, se font servir selon des règles de priorité données et quittent le système, Une représentation graphique d'une file d'attente classique est donnée par la figure (2.1) sous-dessous.

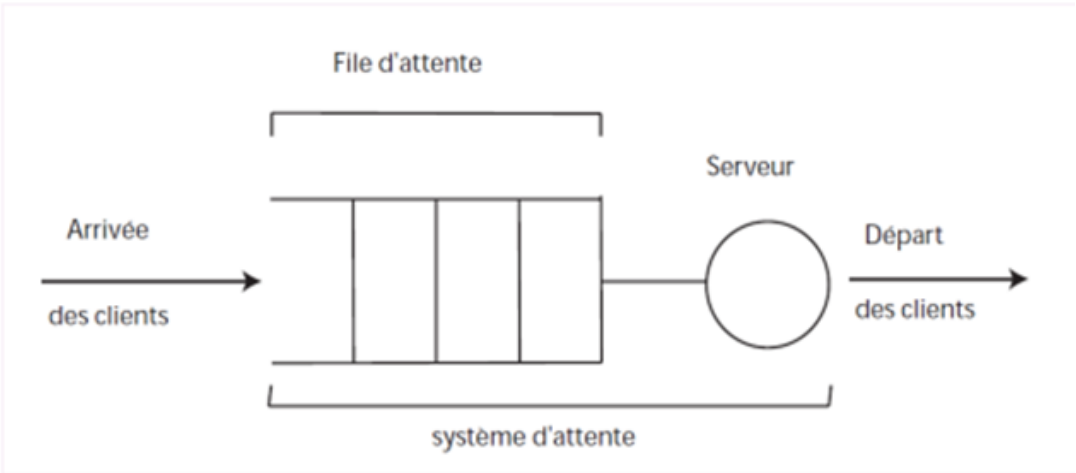

Figure 2.1 : Système de file d'attente.

La théorie des files d'attente consiste à modéliser et à analyser de nombreuses situations en apparence très diverses. Elle fut développée pour fournir des modèles permettant de prévoir le comportement de systèmes répondant à des demandes aléatoires.

Les files d'attente sont généralement considérées comme un outil très puissant d'évaluation des mesures de performance de divers systèmes réels. [14]

2.2.3 Description d'une file d'attente

Un système de files d'attente classique se base principalement sur les éléments suivants [15] :

- **Processus d'arrivée des clients**

Dans le formalisme des files d'attente, les arrivées des clients sont caractérisées par l'ensemble des instants ou dates d'arrivée de chaque client. Cette collection des dates d'arrivée s'appelle le processus des arrivées. Lorsque les dates d'arrivées sont imprévisibles, elles sont modélisées par des variables aléatoires, et le processus des arrivées est alors une collection de variables aléatoires, c'est-a-dire un processus stochastique.

Soit a_n la date d'arrivée du n^{ième} client. Le processus des arrivées est alors la collection$\{a_n, n = 1,2, \dots\}$ de variables aléatoires. Lorsque les arrivées sont déterministes, ces variables aléatoires sont des constantes, le processus des arrivées devient une collection de réels positifs.

- **Inter-arrivée**

On appelle inter-arrivée la différence entre deux dates d'arrivées successives. On note en général la n^{ième} inter-arrivée τ_n qui s'exprime comme :

$$\tau_n = a_{n+1} - 1.$$

Ainsi le processus des arrivées peut être complètement défini par la donnée de la suite des inter-arrivées $\{\tau_n, n = 1,2, \dots\}$, si la date d'arrivée du premier client a_n est connue. En effet, en sommant l'équation pour i=1 à n−1, on obtient :

$$a_n = a_1 + \sum_{i=1}^{n-1} \tau_i.$$

Souvent les arrivées des clients sont indépendantes. Ceci veut dire que les durées séparant deux arrivées successives sont des variables aléatoires indépendantes et de même loi .Les différentes formes du processus d'arrivées sont : arrivées régulières (les instants séparant deux arrivées successives sont constants et égaux à $1/\lambda$), arrivée Poissonnière (les instants d'arrivées forment un processus de Poisson), arrivées suivant une loi d'Erlang d'ordre k

(C'est un processus où les arrivées sont Poissonnières mais seulement les clients d'ordre multiple de k sont considérés),etc.

- **Taux d'arrivée**

On appelle taux d'arrivée des clients, ou intensité des arrivées, le nombre moyen de client qui arrivent dans le système par unité de temps.

- **Arrivées par groupe**

Aux instants d'arrivée, plusieurs clients arrivent simultanément. On parle alors d'arrivées par groupe .La donnée de la loi caractérisant la taille du groupe est alors nécessaire pour déterminer le comportement du processus des arrivées.

- **Clients impatients**

Les clients sont dits impatients lorsqu'ils quittent le système avant d'être servis. Cela peut arriver soit dès l'arrivée, s'ils jugent la file trop importante, on parle alors de découragement, soit après avoir attendu et c'est alors un abandon.

- **Le processus de service**

Un client entre dans une file d'attente pour utiliser des ressources qui sont modélisées par un serveur. Chaque client entrant dans une file d'attente va donc utiliser le serveur pendant une certaine durée qui dépend du client. Le processus de service pourra être d'une complexité extrême, dans la pluparts des systèmes de file d'attente la durée de service de chaque client est indépendante des autres, et qu'elles obéissent toutes à une même loi de distribution: on parle de variables indépendantes et identiquement distribuées (i.i.d).

Soit $\sigma_n$le service demandé par le n^{ième} client de la file (n^{ième} arrivée). Les durées de service de tous les clients sont donc décrits par le processus de service $\{\delta_n, n = 1,2, \dots\}$, peut être déterministe ou aléatoire.

Comme pour les arrivées, le service peut être fourni par groupe, comme par exemple dans le cas d'un ordinateur avec traitement parallèle, ou encore un modèle de visite guidée ou d'embarquement dans un train.

Le processus de service peut dépendre du nombre de clients en attente, par exemple plus rapide si la file grossit, on parle alors de service dépendant de l'état du système.

Le processus de service peut être stationnaire ou non. Souvent les durées de service sont σ_n deux à deux indépendantes.

La date de départ ou date de sortie est la date de fin de service des clients. Elle est en général notée d_n et peut s'exprimer en fonction de la date d'arrivée, le temps d'attente et la durée de service effectivement reçue. Si w_n dénote le temps d'attente du nième client et si le service reçu est le service demandé, on a pour chaque client :

$$d_n = a_n + \delta_n + w_n .$$

- **Capacité de la file**

La capacité de la file à accueillir des clients en attente de service peut être finie ou infinie. Soit k la capacité de la file (incluant le ou les clients en service). Une file à capacité illimitée $k=+\infty$ vérifie. Lorsque la capacité de la file est limitée et qu'un client arrive alors que cette dernière est pleine, le client est perdu.

- **Le nombre de serveurs**

La capacité de service dépend de la capacité de chaque serveur et du nombre de serveurs disponibles. Le terme « serveur » représente ici la ressource et, en général, on suppose qu'un serveur ne traite qu'un client à la fois. Les systèmes de files d'attente fonctionnent avec serveur unique ou serveurs multiples.

- ✓ **Systèmes de files d'attente avec un serveur unique**

Ce type de systèmes comprend un seul serveur qui offre le service. Les exemples de systèmes de files d'attente avec serveur unique sont nombreux : les petits magasins avec une seule caisse, tels que les dépanneurs, certains cinémas, certains lave-autos et établissements de restauration rapide avec guichet unique.

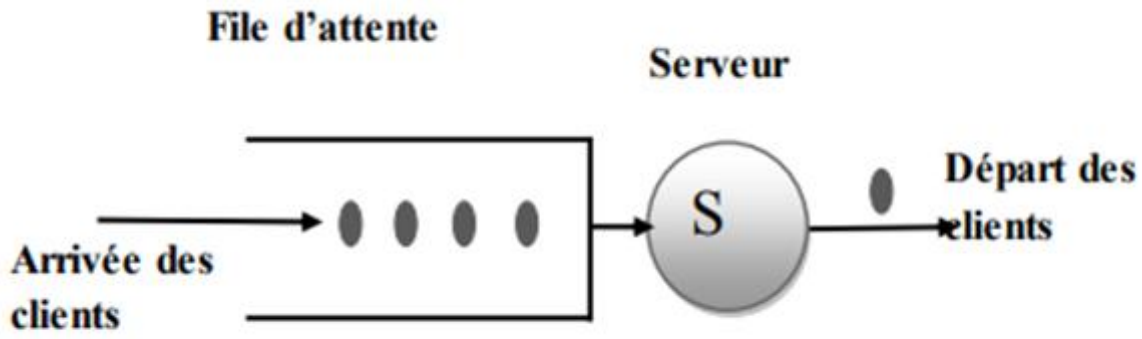

Figure 2.2 : Système de file d'attente avec un serveur unique.

- ✓ **Systèmes de files d'attente avec plusieurs serveurs**

Ce type de systèmes comprend plusieurs serveurs qui fournissent le service. Les serveurs sont en parallèle ou en série.

a)Système de files d'attente avec plusieurs serveurs en parallèles

Chaque client ne requiert le service que d'un seul serveur et tous les serveurs sont capables de fournir ce service. Dans ce type de files d'attente, on a S serveurs en parallèle. Le client

entrant au système n'est pas obligé de visiter tous les S serveurs. Si chaque serveur est doté d'une file, au moment de son arrivée le client choisit l'une d'entre elles ; bien sûr il doit choisir la file la moins longue ; ou bien il rejoint une position donné dans la file si elle est unique, puis il sera sélectionné de cette file selon la politique adoptée dans le système. Si n, le nombre de clients présentes dans le système est inférieur à S, ce qui revient à dire tant que les serveurs ne sont pas tous occupés, alors la file ne se constitue pas et tout client arrivant est immédiatement pris en charge par l'un des serveurs libres.

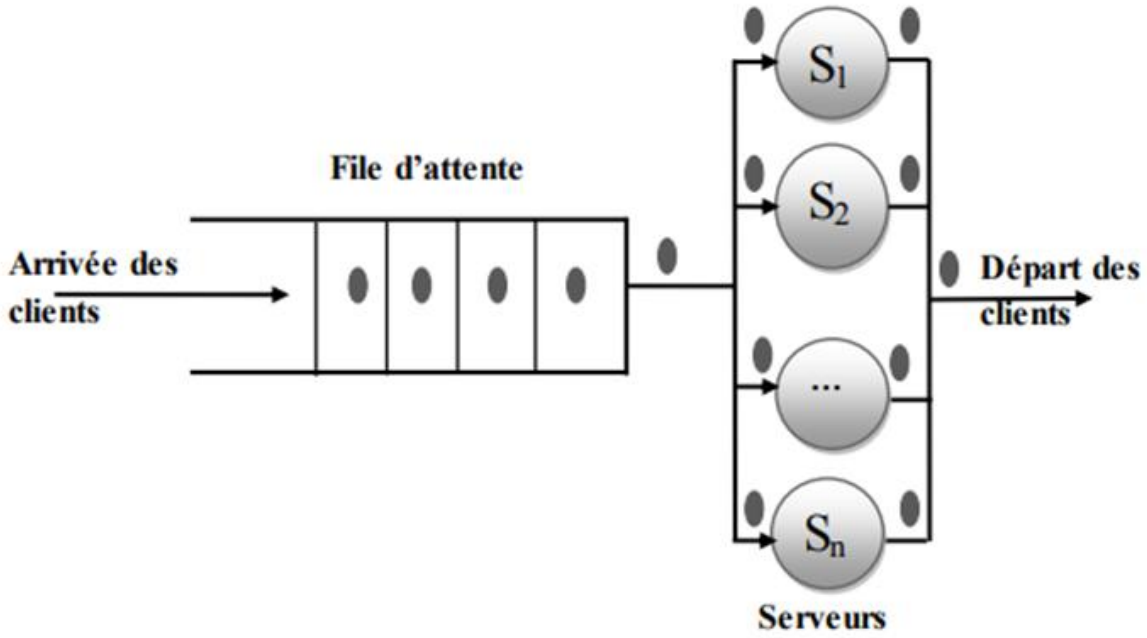

Figure 2.3 : Système de file d'attente avec plusieurs serveurs en parallèles .

b) Système de files d'attente avec plusieurs serveurs en série

Dans ce type de files d'attente, on a S serveurs en série. Le client entrant au système doit visiter plusieurs serveurs successifs dans un ordre fixe pour recevoir le service. Ce type est appelé aussi système de files d'attente en cascade, on les rencontre dans une chaine de fabrication, dans la circulation des dossiers dans une administration…etc.

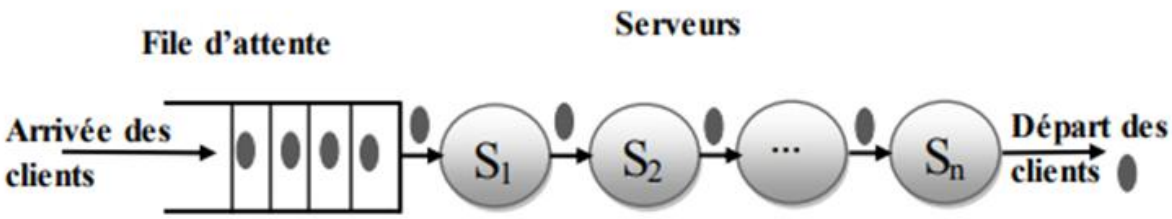

Figure 2.4 : Système de file d'attente avec plusieurs serveurs en série.

- **Discipline de service**

La discipline de service détermine l'ordre dans lequel les clients sont rangés dans la file et y sont retirés pour recevoir un service. Les disciplines les plus courantes sont :

- **FIFO (FCFS)** : *''first in, first out''*, le premier client arrivé sera le premier à quitter la file, c'est la file standard dans laquelle les clients sont servis dans leur ordre d'arrivée.
- **LIFO :** *''last in, first out ''*, cela correspond à une pile, dans laquelle le dernier client arrivé (donc posé sur la pile) sera le premier traité (retiré de la pile).
- **RANDOM** (aléatoire) : le prochain client qui sera servi est choisi aléatoirement dans la file d'attente.
- **Round-Robin** (cyclique) : tous les clients de la file d'attente entrent en service à tour de rôle, effectuant un quantum *Q* de leur temps de service et sont replacés dans la file, jusqu' à ce que leur service soit totalement accompli.
- **PS (Processor Sharing) :** c'est le cas limite de la distribution Round-Robin lorsque le quantum de temps *Q* tend vers 0 .Tous les clients sont servis en même temps, mais avec une vitesse inversement proportionnelle au nombre de clients simultanément présents. Si le taux du serveur est égal à μ et qu'à un instant donné il y a n clients à la station, tous les clients sont donc servis simultanément avec un taux μ/n.
- **Priorité:** il existe plusieurs classes de clients de priorité différente, le client avec la priorité la plus haute est servi en premier, et la discipline de service est FIFO au sein d'une même classe. Pour la discipline ''Priorité'', on distingue deux variantes, selon que le client en service est interrompu lorsqu'un client plus prioritaire arrive ou non. La discipline est une discipline de priorité non préemptive si le client en service n'est pas interrompu lorsqu'un client plus prioritaire entre dans le système. Dans le cas contraire, le client en service est interrompu lorsque qu'un client plus prioritaire arrive, afin que ce dernier puisse commencer son service aussitôt. Une fois le service du client prioritaire terminé, le client interrompu reprend son service, et on distingue deux politiques :

✓ -Priorité préemptive avec recommencement: le client interrompu reprend son service au début.

✓ Priorité préemptive avec continuation: le client interrompu reprend son service là où il avait été interrompu.

2.2.4 Classification des files d'attente

Pour la classification des files d'attente, on a recours à une notation symbolique appelée *notation de kendall* qui a la forme générale suivante *A/B/C [/D/E]*, où [11]:

- *A* : Distribution des temps des inter-arrivées ;
- *B* : Distribution des temps de service ;
- *C* : Nombre de serveurs ;
- *D* : Capacité(ou taille) de système, la valeur par défaut : ∞ ;

- *E* : Discipline de service, la valeur par défaut : FIFO.

Pour « *A* » et « *B* », la liste suivante résume les lois de probabilités les plus couramment rencontrées dans la modélisation des systèmes de file d'attente ainsi que les symboles associés.

- *M* : Désigne la loi Exponentielle (Markovienne) ;
- *D* : Correspond à une loi Déterministe ;
- *G* : Désigne la loi Générale, elle est utilisée lorsqu'aucune hypothèse particulière n'est faite sur le processus d'arrivée ou le processus de service.

Pour voir les formules mathématiques de ces lois de distribution des temps des inter-arrivées et la distribution des temps de service , voir l'annexe.

2.2.5 Les mesures de performance d'un système de file d'attente

Les mesures de performance de système de file d'attente sont un aspect très critique et important. L'analyse mathématique de système de file d'attente est effectuée uniquement pour obtenir les différentes mesures de performance qui peuvent être utilisées pour déterminer la mesure de l'efficacité d'un processus donné. [16]

Afin de faciliter la compréhension des paramètres de performance, le tableau suivant présente les symboles et la terminologie utilisés pour l'étude des systèmes de file d'attente.

symbole	Signification
λ	le taux d'arrivé
μ	le taux de service
ρ	L'intensité du trafic
W_q	Temps moyen d'attente en file
W_s	Temps moyen d'attente en système
L_q	longueur moyenne de la file d'attente
L_s	longueur moyenne de la file d'attente dans le système

Tableau 2.1 : les paramètres de performance d'un système de file d'attente.

Avant la détermination des paramètres de performance du système, nous présentons d'abord *''la loi de Little''* qui joue un rôle très important dans l'étude des systèmes des files d'attente.

❖ **La loi de Little**

La loi de Little est une relation très générale qui s'applique à une grande classe de systèmes. Elle ne concerne que le régime permanent du système. Aucune hypothèse sur les variables aléatoires qui caractérisent le système (temps d'inter-arrivées, temps de service,...) n'est nécessaire. La seule condition d'application de la loi de Little est que le système soit stable. La loi de Little s'exprime telle que dans la propriété suivante [11]:

*Nombre moyen de clients dans un système = intensité d'arrivée * temps moyen qu'un client passe dans le système.*

C'est-à-dire :

$$L_s = \lambda * W_s.$$

$$L_q = \lambda * W_q.$$

Les mesures de performances utilisées lors de l'analyse du modèle d'attente sont les suivantes [16] :

- ✓ **Nombre moyen de client dans le système de files d'attente**

Le nombre moyen de clients dans le système est égal au nombre de clients en attente plus le nombre de clients en cours de service.

- ✓ **Nombre moyen de client dans la file d'attente**

Le nombre moyen de clients en attente dans la file d'attente pour obtenir le service.

- ✓ **Temps de réponse *T***

Le temps qu'un client passe dans le système de l'instant de son arrivée à la file d'attente jusqu'à l'instant de son départ du serveur est appelé temps de réponse ou temps de séjour.

- ✓ **Temps d'attente moyen dans le système**

Le temps total passé par un client en file d'attente plus le temps de service.

- ✓ **Temps d'attente moyen dans la file**

Le temps moyen passé par un client en file d'attente pour obtenir son tour de service.

- ✓ **Le débit d'un système**

Le débit est défini comme le nombre moyen de clients quittant le système. Le débit d'un système de file d'attente est égal à son taux de départ, c'est-à-dire au nombre moyen de clients qui sont traités par unité de temps.

Autrement dit, le débit de la file est égal au taux d'arrivée! C'est un résultat intuitivement correct pour une file d'attente stable : à l'équilibre, le taux de sortie de la file est égal au taux d'arrivée.

- ✓ **Longueur de la file d'attente *L***

La longueur de la file d'attente *L* représente le nombre de clients dans la file d'attente.

- ✓ **L'intensité du trafic ρ**

L'intensité du trafic est la mesure de l'occupation moyenne d'une installation pendant une période donnée. Il est défini comme le rapport entre le temps pendant lequel une installation est occupée et le temps pendant lequel cette installation est disponible pour occupation. Autrement dit, L'intensité du système est donnée par la relation du taux d'arrivée et le taux de service.

$$\rho = \frac{\lambda}{\mu}.$$

2.2.6 Étude de la file en régime stationnaire

Si $\rho < 1$, soit $\lambda < \mu$ la file d'attente est stable, il existe un régime dit stationnaire ou permanent : le nombre de clients dans le système à l'instant *t, X(t)* converge en loi lorsque t→∞ vers une variable aléatoire discrète *N*.

Si $\rho \geq 1$, $\lambda \geq \mu$ la file d'attente est instable, il n'existe pas de régime permanent ou stationnaire. La file grossit indéfiniment. [15]

2.3 Les files d'attente avec vacance

2.3.1 Description du modèle des files d'attente avec vacance

Les files d'attente avec vacance sont des files d'attente dans lesquels le serveur prend des vacances d'une tâche primaire pour s'occuper d'une ou plusieurs tâches secondaires ou pour se reposer ou pour accomplir quelques tâches de maintenance préventive, pendant ce temps les clients attendent d'être servis. Cette classe de files d'attente peut se trouver dans plusieurs applications.

La représentation graphique d'un système de files d'attente avec vacance est donnée dans le schéma suivant [2] :

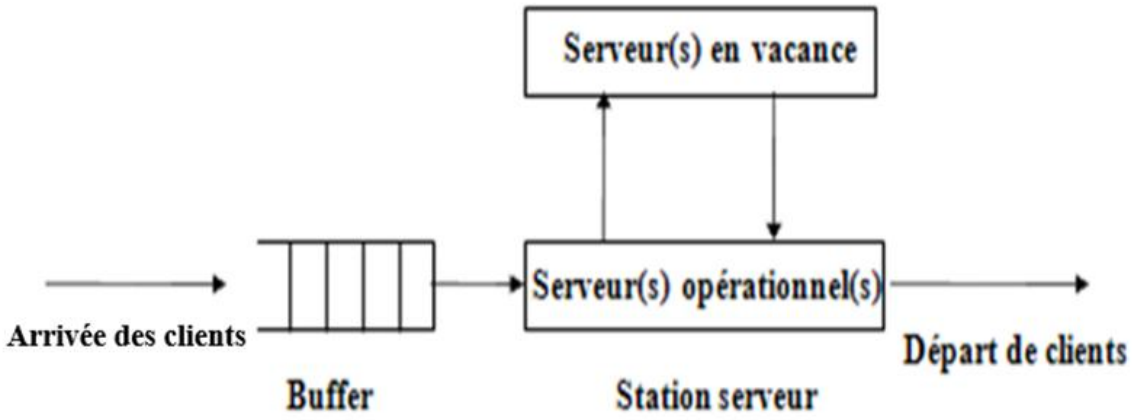

Figure 2.5 : Système de file d'attente avec vacance.

2.3.2 Classification des politiques de vacances de serveurs

Une politique de vacance est caractérisée par trois aspects : la discipline de service, les types de vacance et la durée de vacances.

2.3.2.1 Disciplines de service

Connue aussi sous le nom règle de début de vacances, cette règle détermine deux disciplines de service majeurs :[2]

- ✓ **Un service exhaustif :**

Le serveur doit servir tous les clients dans sa file d'attente et ne peut prendre de vacance qu'une fois que tous les clients sont servis ou que la file d'attente soit vide.

- ✓ **Un service non exhaustif :**

Un serveur peut prendre une vacance même si la file n'est pas vide. Dans ce dernier cas on distingue les politiques suivantes :

- ❖ **Avec un service limité:**

Dans les systèmes de vacance à service limité, quand le serveur retourne d'une vacance il sert un nombre k de clients au maximum puis débutera une autre vacance. C'est-à-dire le serveur sert jusqu'à ce que la file d'attente soit vide ou les k clients seront servis.

- ❖ **Avec un serveur avec barrière :**

Dans les systèmes de vacance à service avec barrière, quand le serveur retourne des vacances, il sert seulement les clients qui étaient dans la file d'attente à son arrivée et ne reprend une autre vacance que si tous ces clients sont servis. [11]

- ❖ **Les Services limités avec barrière :**

Si le serveur trouve '**n**' clients dans la file à son retour d'une vacance, il sert la **min (n, k)** des clients avant de reprendre une autre vacance.

- ❖ **Les services limités exhaustif** *:*

Si le serveur trouve ' **n** ' clients à la fin d'une vacance, les clients qui arrivent durant le service des 'n' clients peuvent être servis si la limite '**k**' n'est pas franchie

Il existe une autre politique de vacance non exhaustive dite la « politique de décision séquentielle de Bernoulli » .basée sur une probabilité fixe 'p' ($0 \leq p \leq 1$). A la fin de chaque service, le serveur sert le prochain client avec la probabilité 'p' et va en vacance avec la probabilité '1-p'.La politique de service de Bernoulli a deux types :

- ➢ **Bernoulli avec barrière :**

Si le serveur trouve n clients dans la file à son retour d'une vacance. Il sert seulement les clients qui étaient en attente dans la file à son arrivée avec une probabilité 'p' avant de débuter une autre vacance.

- **Bernoulli limitée :**

Si le serveur trouve ' **n** ' clients à la fin d'une vacance, les clients qui arrivent durant le service des '**n**' clients peuvent être servis avec la probabilité '**p**' si la limite '**k**' n'est pas franchie.

2.3.2.2 Types de vacances

Il existe 3 types de vacance :[2]

- ✓ **Les vacances uniques**

Dans ce cas, à la fin d'une vacance, le serveur attend jusqu'à ce qu'au moins un client soit servi avant de commencer une autre vacance. Autrement dit, il y'a au moins une période de service entre deux périodes de vacance.

- ✓ **Les vacances multiples**

Dans ce cas, si le serveur retourne d'une vacance et trouve le système vide il commence immédiatement une autre vacance ; et continue de prendre des vacances successives jusqu'à ce qu'il trouve au moins un client dans la file d'attente.

- ✓ **Les vacances hybrides**

Dans ce cas, si le serveur revient d'une vacance et trouve le système vide il attend alors une durée de temps aléatoire (distribuée exponentiellement) à la fin de la quelle, si aucun client n'est arrivé, le serveur commencera une autre vacance.

2.3.2.3 Les politiques de fin de vacance

Dans le schéma de classification constaté précédemment, les différentes disciplines de service définissent l'instant du début de vacance.

Il existe un autre schéma de classification possible qui considère l'instant où le serveur retourne de vacance. Dans ce schéma, trois politiques de vacance sont définies et dans chacune d'elle, le serveur prend une vacance, seulement quand le système est vide (service exhaustif).

- ✓ **N-vacance** : Dans cette politique le serveur retourne de la vacance quand la taille de la file d'attente est supérieure ou égale à N.
- ✓ **D-vacances :** Dans cette politique le serveur retourne d'une vacance quand la somme des temps de service des clients dépasse la valeur D.
- ✓ **T-vacance :** Dans ce cas, le serveur est non activé(en vacance) T unités de temps après la fin d'une période de service. Après cette vacance (de longueur T), le serveur est réactivé pour servir les clients seulement quand au moins un client est présent, sinon le serveur prend une autre période de vacance de longueur T.

2.3.2.4 Durée d'une vacance

Les vacances du serveur sont généralement considérées comme des variables aléatoires indépendantes et identiquement distribuées avec une certaine loi de

probabilité, selon les caractéristiques du système modélisé. [2]

2.4 Conclusion

Dans ce chapitre nous avons abordé la théorie des files d'attente classiques, et nous avons vu que les modèles de files d'attente standards ne permettent pas de décrire le comportement réel des clients et des serveurs, d'où vient la nécessité de d'utilisation des files d'attente avec vacance pour la modélisation des systèmes d'attente.

Nous avons vu que le phénomène de vacance, rend l'analyse des files d'attente complexe. Ainsi les résultats analytiques n'existent que pour quelques modèles et avec des hypothèses contraignantes sur certains paramètres, et par conséquent, les chercheurs dans ce domaine se sont orientés vers des méthodes numériques, des approximations ou des simulations.

Dans le prochain chapitre, nous nous intéressons à la simulation et ses principes et les types de simulation.

Chapitre 3

3. La simulation des systèmes de file d'attente

3.1 Introduction

Lorsque la situation est trop complexe pour être analysée par simulation mentale seule, nous nous appuyons sur la simulation informatique.

La simulation en tant que technique scientifique puissante vise dans le processus de service à évaluer le modèle approprié, ou teste les effets des changements dans le système exploité existant. Elle permet de limiter le risque et d'éviter le coût d'une série d'épreuves réelles et elle peut offrir un aperçu sur le développement d'un système trop complexe pour simuler avec de simples formules mathématiques.

Le but de ce chapitre est de fournir au lecteur une compréhension de base de la simulation, les trois types de simulation et avoir une présentation sur la simulation à événement discret ainsi que leurs approches et les méthodes utilisées pour collecter des données d'observation en régime permanent. Enfin, nous allons connaitre les travaux connexes liés à l'analyse et l'évaluation des performances des systèmes de files d'attente.

3.2 compréhension de base de la simulation

La simulation est l'imitation d'un processus ou d'un système du monde réel au fil du temps. La simulation implique la génération d'une histoire artificielle du système et l'observation de cette histoire artificielle pour tirer des inférences concernant les caractéristiques de fonctionnement du système réel représenté.

La simulation est une méthodologie de résolution de problèmes indispensable pour la résolution de nombreux problèmes du monde réel. Elle est utilisée pour décrire et analyser le comportement d'un système, poser des questions « Et qu'est-ce qui se passerait si ?» sur le système réel et aider à la conception des systèmes réels.

3.2.1 Les étapes de développement d'un modèle de simulation

Une étude de simulation approfondie et solide va passer par l'ensemble des étapes suivantes [17]:

1. **L'identification du problème :** Il faut définir le problème explicitement :
   Quelles sont les sorties désirées ?
   Y a-t-il d'importants éléments stochastiques dans la procédure ?
   Des files d'attente sont-elles impliquées ?

2. **Fixer les objectifs et le plan global du projet :** Il faut se poser certaines questions :
   Comment le problème pourrait-il être résolu sans simulation ?
   Pourquoi la simulation est-elle la meilleure solution ?
   Comment d'autres ont-ils déjà résolu ce type de problème ?
   Le problème peut-il réellement être résolu ?
3. **La formulation du problème :** Ceci a pour but d'identifier toutes les entrées dont on a besoin pour le modèle et d'y associer les sorties pour pouvoir collecter les données à l'étape suivante.
4. **Collecte et entrée des données :** La collecte des données se base sur les entrées et sorties définies précédemment. On procède comme suit :
   Observer le système de trafic à étudier et collecter au début seulement un échantillon de données.
   Faire une revue de littérature sur les simulations et études précédentes.
   Faire un plan de collecte des données.
   Adapter la taille des échantillons à la calibration et la validation.
   Etudier les données collectées pour qu'elles soient compatibles avec les besoins du modèle de simulation (calcul de centre et de déviation, analyses des distributions, régressions, conversion des unités).
5. **Codage.** Le modèle conceptuel construit à l'étape 3 est écrit sous une forme reconnaissable par ordinateur (c'est-à-dire un modèle opérationnel). Dans cette étape vous écrivez la logique du modèle en utilisant un langage informatique.
6. **Vérification :** La vérification fait référence au processus consistant à déterminer si le modèle opérationnel fonctionne comme prévu.
7. **Validation :** La validation permet de comparer les données calculées par l'ordinateur et les données mesurées sur le terrain. Si les résultats de la validation sont acceptables, le modèle de simulation est prêt pour les applications. Sinon, il faut effectuer d'autres calibrations et validations.
8. **Conception expérimentale :** Pour chaque scénario à simuler, des décisions doivent être prises concernant la durée de la simulation, le nombre de simulations nécessaires et le mode d'initialisation, selon les besoins.
9. **Production et analyse des résultats :** Toutes les simulations devraient être sauvegardées et le code devrait offrir la possibilité de nommer toutes les sorties de résultat.
10. **Documentation et rapports :** La documentation ne devrait pas seulement contenir les résultats obtenus. Elle devrait également contenir un manuel pour l'utilisateur et pour l'opérateur du système.
11. **L'implémentation :** Si le client a été impliqué tout au long de l'étude et que l'analyste de simulation a suivi rigoureusement toutes les étapes, il est probable que la mise en œuvre sera réussie.

3.2.2 Quelques avantages de la simulation

- ✓ **Faire les bons choix [17] :**

La simulation vous permet de tester chaque aspect d'un changement ou d'un ajout proposé sans engager de ressources pour leur acquisition.

- ✓ **Comprendre « Pourquoi ? » :**

Les managers veulent souvent savoir pourquoi certains phénomènes se produisent dans un système réel. Avec la simulation, vous déterminez la réponse aux questions du « pourquoi » en reconstituant la scène et en procédant à un examen microscopique du système. Vous ne pouvez pas accomplir cela avec un système réel car vous ne pouvez pas le voir ou le contrôler dans son intégralité.

- ✓ **Explorer les possibilités :**

L'un des plus grands avantages de l'utilisation de la simulation est qu'une fois que vous avez développé un modèle de simulation valide, vous pouvez explorer de nouvelles politiques, procédures d'exploitation ou méthodes sans les dépenses et les perturbations liées à l'expérimentation du système réel. Les modifications sont incorporées dans le modèle et vous observez les effets de ces changements sur l'ordinateur plutôt que sur le système réel.

3.2.3 Quelques inconvénients de la simulation

- ✓ **Les résultats de simulation peuvent être difficiles à interpréter [17]**:

Étant donné que la plupart des sorties de simulation sont essentiellement des variables aléatoires (elles sont généralement basées sur des entrées aléatoires), il peut être difficile de déterminer si une observation est le résultat d'interrelations du système ou d'un caractère aléatoire.

- ✓ **La modélisation et l'analyse de simulation peuvent être longues et coûteuses :**

Lésiner sur les ressources pour la modélisation et l'analyse peut entraîner un modèle de simulation et/ou une analyse qui ne suffit pas à la tâche

- ✓ **La simulation n'est possible que si le développeur comprend parfaitement le système.**

3.3 Les types de simulation informatique

Il existe 3 types de la simulation informatique :

3.3.1 La simulation continue

La simulation continue est une évaluation numérique d'un modèle informatique d'un système dynamique physique qui suit en permanence les réponses du système au fil du temps selon un ensemble d'équations impliquant généralement des équations différentielles.[18]

3.3.2 La simulation de Monte Carlo

La méthode de Monte-Carlo est une méthode statistique qui se base sur le concept d'expériences aléatoires, elle s'applique pour des systèmes en équilibre thermodynamique. L'état d'équilibre est unique quelque soit le nombre de répétitions de l'expérience. Cependant, le choix initial intervient dans la convergence et le temps de calcul. Le principe de la méthode est de résoudre les problèmes par la voie des grandeurs probabilistes, telle que l'espérance mathématique. Cette méthode a une large utilisation dans divers domaines scientifiques, technologiques, économiques, ayant un aspect aléatoire. [19]

3.3.3 Simulation à évènement discret

Un système d'événements discrets est un système dans lequel les changements d'état (événements) se produisent à des instances discrètes dans le temps, et les événements ne prennent aucun temps pour se produire. On suppose que rien (c'est-à-dire rien d'intéressant) ne se passe entre deux événements consécutifs, c'est-à-dire qu'aucun changement d'état n'a lieu dans le système entre les événements (contrairement aux systèmes continus où les changements d'état sont continus). Les systèmes pouvant être considérés comme des systèmes d'événements discrets peuvent être modélisés à l'aide de la simulation d'événements discrets. (D'autres systèmes peuvent être modélisés, par exemple avec des modèles de simulation en continu). [20]

Les principaux termes du vocabulaire de La simulation par événements discret sont [21] :

Terme	Description
Système	Ensemble d'entités qui interagissent au fil du temps pour atteindre un ou plusieurs objectifs.
Modèle	Représentation abstraite d'un système, contenant habituellement des relations structurelles, logiques ou mathématiques qui décrivent un système en termes d'état, d'entités et de leurs attributs, ensembles, processus, événements, activités et retards.
État du système	Ensemble de variables contenant toutes les informations nécessaires pour décrire le système à tout moment.
Entité	Tout objet ou composant du système qui nécessite une représentation explicite dans le modèle.
Attributs	Les propriétés d'une entité donnée.
Liste	Une collection d'entités associées (de façon permanente ou temporaire), commandées d'une manière logique (comme tous les clients en attente, commandées par "premier arrivé, premier servi" ou par priorité).
Événement	instantané qui modifie l'état d'un système.
Avis d'événement	Un enregistrement d'un événement qui doit se produire à l'heure actuelle ou à une date ultérieure, ainsi que toutes les données connexes nécessaires pour exécuter l'événement ; au minimum, l'enregistrement comprend le type d'événement et l'heure de l'événement.

Liste des événements	Liste des avis d'événements pour les événements futurs, ordonnée selon l'heure de l'événement ; aussi connue sous le nom de liste des événements futurs (FEL).
Activité	Une durée de temps de durée spécifiée, qui est connu quand il commence (bien qu'il puisse être défini en termes de distribution statistique).
Délai	Une durée indéterminée non précisée, qui n'est pas connue jusqu'à ce qu'elle atteigne.
Horloge	Une variable représentant le temps simulé.

Tableau 3.1 : Tableau des principaux termes de La simulation par événements discret.

3.3.3.1 Les approches de la simulation à évènement discret

- **L'approche de planification d'événements (*Event scheduling*)**

Le concept de base de l'approche de planification d'événements est d'avancer le temps jusqu'au moment où quelque chose se passe ensuite (c'est-à-dire, lorsqu'un événement se termine, le temps est avancé jusqu'à l'heure du prochain événement programmé). Un événement libère généralement une ressource. L'événement réaffecte ensuite les objets ou entités disponibles en programmant des activités auxquelles ils peuvent désormais participer. Par exemple, dans la laverie libre-service, si le linge d'un client est terminé et qu'il y a un panier disponible, le panier pourrait être attribué immédiatement au client, qui commencerait alors à décharger la machine à laver. Le temps est avancé jusqu'au prochain événement programmé (généralement la fin d'une activité) et les activités sont examinées pour voir si certaines peuvent maintenant commencer en conséquence.

- **L'approche d'analyse d'activité (*Activity scanning*)**

La deuxième structure de modélisation de simulation est l'analyse d'activité. L'analyse d'activité est également connue sous le nom d'approche en deux phases. L'analyse d'activité produit un programme de simulation composé de modules indépendants en attente d'exécution. Dans la première phase, un laps de temps fixe est avancé, ou scanné. Dans la phase deux, le système est mis à jour (si un événement se produit). L'analyse des activités est similaire à la programmation basée sur des règles (si la condition spécifiée est remplie, une règle est exécutée).

- **L'approche en trois phases (*Three-phase*)**

La troisième structure de modélisation de simulation est connue sous le nom de l'approche en trois phases. Dans la première phase, le temps est avancé jusqu'à ce qu'il y ait un changement d'état dans le système ou jusqu'à ce que quelque chose se passe ensuite. Le système est examiné pour déterminer tous les événements qui ont lieu à ce moment (c'est-à-dire toutes les activités terminées qui se produisent). La deuxième phase est la libération des

ressources dont la fin de leurs activités est prévue à ce moment-là. La troisième phase consiste à démarrer les activités, compte tenu d'une image globale de la disponibilité des ressources.

Des inexactitudes de modélisation possibles peuvent se produire avec les méthodes de l'analyse d'activité et de modélisation en trois phases, car des tranches de temps discrètes doivent être spécifiées. Si l'intervalle de temps est trop large, les détails sont perdus. Ce type de simulation deviendra moins populaire à mesure que la puissance de calcul continuera d'augmenter et que les coûts de calcul continueront de diminuer. [17]

3.4 Les Méthodes de simulation en régime permanent

Nous discuterons brièvement dans cette section de certaines des méthodes courantes utilisées pour recueillir des données d'observation de l'état stationnaire au cours d'une simulation. Le problème spécifiquement considéré est la façon dont on peut obtenir n séries de simulation indépendantes sur lesquelles les paramètres de sortie peuvent être observés et estimés afin que les résultats des séries puissent être utilisés pour obtenir une estimation de confiance appropriée. Les trois techniques communément proposées pour cela sont la méthode des sous-intervalles, la méthode régénérative et la méthode de réplication.

3.4.1 La méthode des sous-intervalles (*Subinterval Method*)

Cette méthode ne comporte en réalité qu'un seul très long cycle de simulation qui est convenablement subdivisé en une période transitoire initiale et en n lots. Chacun de ces lots est alors traité comme un cycle indépendant de l'expérience de simulation, tandis qu'aucune observation n'est faite pendant la période transitoire qui est traitée comme l'intervalle de préchauffage. Un exemple de ceci a été montré à la figure 3.1. Notez qu'on choisirait normalement des intervalles de lots de taille égale mais ce n'est pas obligatoire. On peut choisir des intervalles de lots de différentes longueurs s'il existe une logique raisonnable sur laquelle cela peut être basé.

Le choix d'un intervalle de lots important mènerait effectivement à des lots indépendants et, par conséquent, à des exécutions indépendantes de la simulation selon les besoins aux fins de l'estimation de confiance.

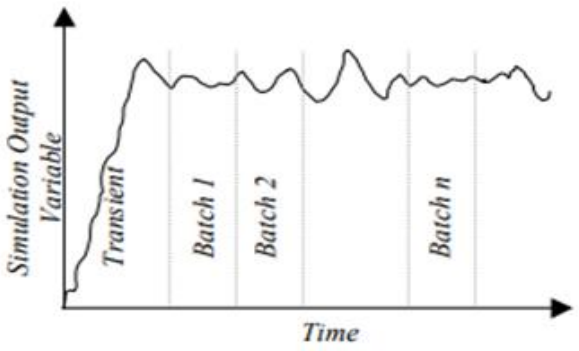

Figure 3.1 : La méthode des sous-intervalles.

3.4.2 La méthode régénérative (*Regenerative Method*)

Cette méthode vise essentiellement à réduire le problème de corrélation entre les lots. Nous utilisons toujours un long terme comme avant mais sélectionnons un état correctement identifié du système comme l'état régénératif et les instants de temps où cela se produit comme les points régénératifs. Les lots commencent et se terminent à ces points de régénération une fois l'état d'équilibre atteint. Cette méthode est illustrée à la figure 3.2.

Le choix de l'état de régénération approprié peut ne pas être aussi difficile qu'il peut d'abord apparaître. Considérez la variable N représentant le nombre dans une file d'attente, qui est simulée. Cela s'accumulera généralement à partir d'une valeur zéro, mais, si le système est stable, il finira également par revenir à une valeur zéro. Cela se produit de manière répétée lorsque l'état du système fluctue entre des valeurs entières positives et non nulles entre les points de valeur zéro. Les points zéro value sont intéressants car une fois que le système atteint cet état, son évolution future ne dépend pas de la façon dont il a réellement atteint cet état. Ce comportement du système implique que la façon dont le système évolue dans un cycle de ralenti à ralenti de ce type ne dépendra pas des cycles précédents de ce type et n'affectera pas non plus les cycles futurs de ce type. Puisque c'est le cas, nous pouvons en effet choisir ces cycles de ralenti à ralenti comme les lots indépendants suggérés dans la description précédente et illustrés à la figure 3.2. [22]

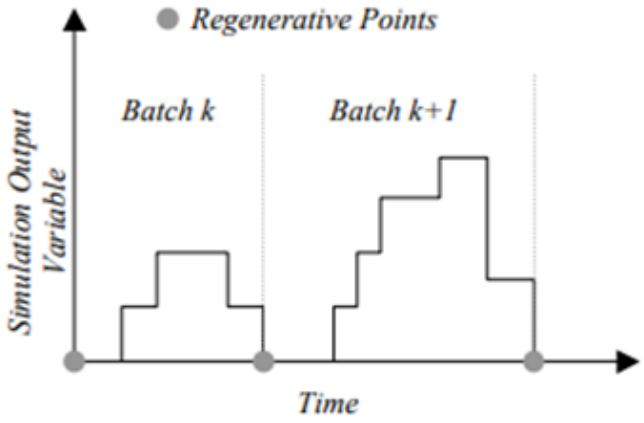

Figure 3.2 : La méthode régénérative.

3.4.3 La méthode de réplication (*Replication Method*)

La troisième méthode, la suppression, exclut la phase transitoire initiale qui est influencée par les conditions initiales. Les données sont collectées à partir de la simulation uniquement après la fin de la phase transitoire (préchauffage). Cette idée est illustrée à la figure 3.3. La difficulté avec la méthode de suppression est la détermination de la durée de la phase transitoire. Bien que des techniques statistiques élégantes aient été développées, une méthode satisfaisante consiste à tracer la sortie d'intérêt au fil du temps et à observer visuellement lorsque l'état d'équilibre est atteint. [17]

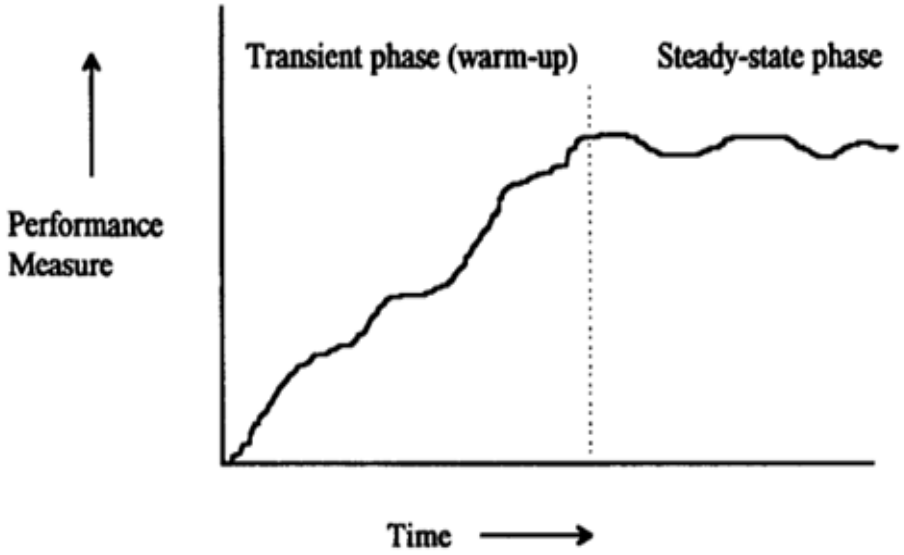

Figure 3.3 : La méthode de réplication.

3.5 Travaux connexes

L'évaluation des performances des systèmes de file d'attente (temps moyen d'attente, nombre moyen de clients, probabilité de blocage,…) se fait suivant différentes techniques. Parmi les mieux adaptées, on distingue Les techniques de simulation qui permettent d'imiter artificiellement sur ordinateur les systèmes à étudier et de prévoir leurs comportements avec plus ou moins de précision.

Dans les sections suivantes, nous allons présenter des travaux dans ces domaines pour l'évaluation et l'analyse des performances des modèles de file d'attente .

Donatien Chedom Fotso et Laure Pauline Fotso [23] proposent un simulateur multi agent permettant d'implémenter tout système d'attente A/B/c où A, B appartiennent à l'ensemble des distributions {Déterministe, Exponentielle, Erlang, Gamma} et de fournir des mesures de performances pour ces modèles. Ce simulateur est utilisé pour vérifier l'hypothèse selon laquelle dans un système bancaire, le type d'organisation de la file d'attente (file unique pour tout les serveurs ou files multiples, une par serveur) n'influence pas les mesures de performances. Le simulateur programmé en Java, les paramètres d'entrée du simulateur sont : le type d'organisation de la file du modèle à simuler, la durée de la simulation, les distributions des durées de services et des arrivées et le nombre de guichets, il utilise la méthode de la transformation inverse [24] pour générer les variables aléatoires. Le simulateur fournit en sortie les mesures de performance du modèle.

Prathamesh Mayekar, J. Venkateswaran, Manu K. Gupta et N. Hemachandra [25] analysent deux mesures des performances importantes et relativement complexes : la probabilité de queue et la fréquence de commutation pour certains systèmes de priorité dynamique utilisant la simulation pour deux classes. Ils construisent un simulateur pour priorité relative et la première date d'échéance basée sur la priorité dynamique dans SimPy, ce dernier est un cadre de simulation basé sur les processus et les événements discrets basé sur

Python standard. Simulateur est construit pour deux classes de clients et tous les résultats de simulation sont en ce qui concerne le temps de service exponentiel et les arrivées de Poisson. Pour toute analyse concernant la priorité relative simulateur dans cette section temps d'exécution est considérée comme étant 11,000 temps unités avec une période de préchauffage de 1 000 unités de temps, tandis que pour l'exécution du système la plus ancienne date d'échéance est de10 000 unités de temps avec une période de préchauffage de 1000 unités de temps.

N Sadeghi et AR Fayek et NG Seresht [26] ont adopté la simulation à événements discrets flous (FDES) qui a été proposée pour simuler des projets de construction. De plus, ils fournissent une nouvelle méthodologie pour considérer l'incertitude subjective dans l'analyse des files d'attente floues dans les modèles (FDES) de construction. Ils ont intégré la théorie de la file d'attente floue avec la méthodologie (FDES) proposé pour améliorer l'applicabilité du (FDES) dans les projets de construction. La méthodologie proposée est validée par des exemples de file d'attente qui sont présenté par un seul serveur avec des temps d'inter-arrivées et des temps de service flous et résolus mathématiquement, et ses aspects pratiques sont illustrés à l'aide d'un exemple d'opération de revêtement d'asphalte.

La simulation à événement discret floue est une intégration de la théorie des ensembles flous avec (DES) qui fournit un cadre pour considérer l'incertitude subjective dans les modèles de simulation de construction. Les cadres (FDES) actuels calculent uniquement le temps de simulation comme résultat de la simulation. Cependant, les mesures de performance de la file d'attente (par exemple, la longueur moyenne de la file d'attente et le temps d'attente) à travers d'importantes sorties de modèle de simulation pour la prise de décision, la recherche de goulots d'étranglement et l'optimisation des ressources de construction ne sont pas analysées dans les méthodologies (FDES) actuelles.

Silvia Ďutkováa , Karol Achimskýa et Dominika Hoštákováa[27] proposent un simulateur comme outil d'optimisation système de file d'attente d'un bureau de poste particulier. Pour l'analyse, il a été sélectionné le système de file d'attente du bureau de poste Bytca (une ville avec 11 000 habitants en Slovaquie). Le principal objectif est d'optimiser le coût du ce bureau. L'application de la méthode de simulation avec la théorie de la file d'attente ils ont permis d'obtenir et puis de analyser les caractéristiques de base du système de bureau de poste Bytca. Le système de file d'attente du bureau de poste est considéré comme système stochastique avec un front sans fin, les intervalles de temps entre les arrivées des clients et les heures de service sont régis par des distributions exponentielles.

Un programme de simulation a été codé selon le modèle du système de file d'attente, puis plusieurs expériences ont été menées .L'objectif était d'analyser état actuel du système et explorer ses autres possibilités. Dans les simulations, la partie initialisation du programme toujours changé. Après analyse du système actuel, le nombre de comptoirs de service a été réduit à chaque fois Le nombre minimal de compteurs a été établi en fonction de l'état de stabilisation du système .De plus il y a des files d'attente trop longues à certains intervalles de temps. C'est pourquoi le nombre de comptoirs de service a été augmentation à ces intervalles de temps. La caractéristique de la file d'attente est devenue plus pratique après cette étape .Le résultat de plusieurs simulations est l'optimisation du nombre de compteurs de service à des intervalles de temps individuels par rapport à caractéristiques du système. Une telle approche conduit souvent à des économies de coûts et aussi à l'optimisation dans d'autres domaines.

3.6 Conclusion

La simulation nous permet de tester des modèles en virtuel et de changer chaque facteur à chaque fois et de montrer tous ces effets sur le système, avant de l'essayer dans la vie réelle pour découvrir quel cas nous donne la solution optimale.

Dans ce chapitre nous nous sommes intéressés à la simulation et ses principes et les types de simulation. Nous avons mis des points sur la simulation à événement discret et son vocabulaire principal. Nous avons connu les déférentes approches de simulation à événement discret et les méthodes utilisées pour collecter des données d'observation en régime permanent ainsi que les travails liés à l'analyse et l'évaluation des performances des systèmes de files d'attente.

Dans le prochain chapitre, nous nous intéressons à la conception de l'outil de simulation que nous avons développé.

Chapitre 4

4. Conception de l'outil de simulation pour les modèles de file d'attente

4.1 Introduction

Dans la conception, Le processus de modélisation vise à obtenir une solution acceptable du système informatique. La solution finalement retenue n'est pas obtenue en une seule itération. Plusieurs étapes sont nécessaires ; ces étapes successives permettent de raffiner le niveau de détails du système à réaliser. De plus modéliser un système avant sa réalisation permet de faciliter la compréhension du fonctionnement d'un système et comprendre sa complexité.

Au cours du présent chapitre, nous procédons à la modélisation d'un modèle de file d'attente avec vacances, de plus, nous présentons l'architecture du ce modèle avec les politiques de vacance (N-vacance, T-vacance et hybride), ensuite nous présentons tous les aspects importants concernant la conception du simulateur. En fin, nous montrons les différents indices de performances et leurs définitions, Nous terminerons ce chapitre par une conclusion.

4.2 Architecture du modèle proposé

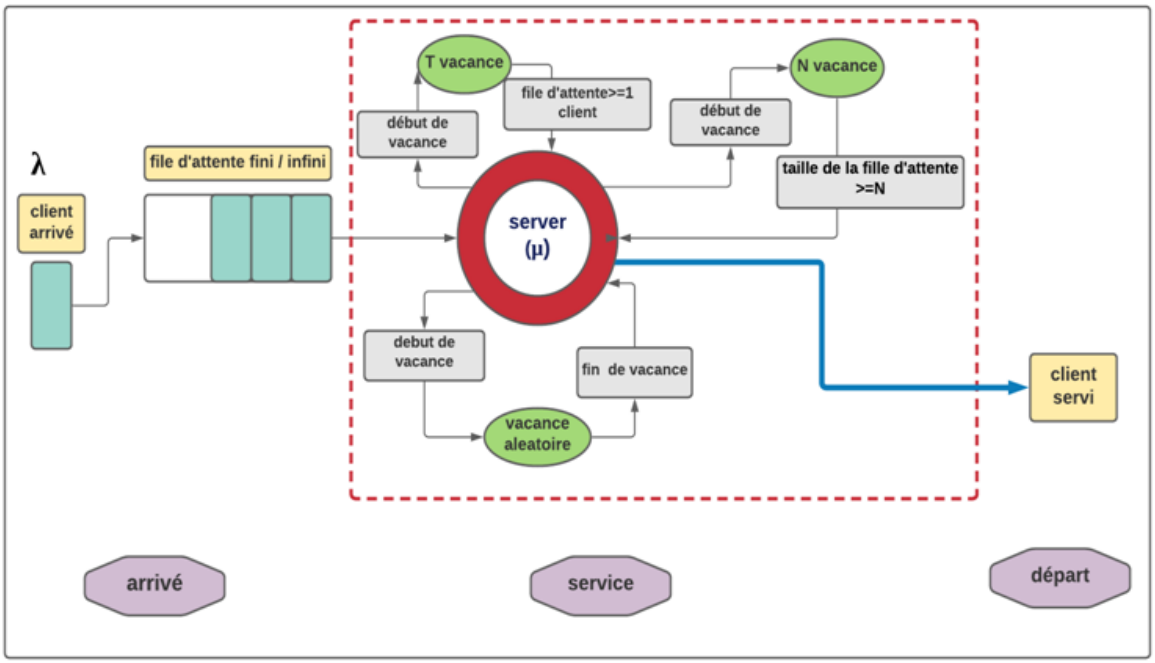

Figure 4.1 : La structure de base d'un système de file d'attente avec vacance.

Notre modèle proposé est composé de trois parties :

- **La première partie** : est **la file d'attente** des clients qui peut être de capacité soit fini ou infini, elle reçoit les clients selon l'une de ces disciplines : FIFO,LIFO , RANDOM, par priorité ... , le temps entre les arrivées est distribué de manière exponentielle ou générale ou déterministe avec la moyenne *« 1/λ »* où λ représente le nombre moyen d'arrivées par unité de temps.
- **La deuxième partie** : contient **Le serveur** qui fourni le service des clients avec une durée de service distribué de manière exponentielle ou générale ou déterministe avec une moyenne *« 1/μ »* où μ représente le nombre moyen des clients servis par unité de temps. Il possède deux états de fonctionnement soit actif (libre ou occupé) ou inactif :

  - ✓ **L'état actif :** dans ce cas si un client arrive et trouve le serveur libre entre directement dans le serveur et rendre le serveur occupé, Tout client arrivant et trouvant le serveur occupé entre directement à la file d'attente pour attendre son tour de service
  - ✓ **L'état inactif :** dans ce cas le serveur prend une durée de vacance pour se reposer ou pour accomplir quelques tâches de maintenance, pendant ce temps les clients arrivant attendent dans la file d'attente pour être servis

Le passage de l'état actif à l'état inactif est contrôlé par la règle de début de vacances, cette règle détermine deux disciplines de service : service exhaustif et service non exhaustif. Dans la première situation, le serveur prend ses vacances lorsque le système est vide. En outre, le serveur est en vacances jusqu'à ce qu'il trouve (en revenant des vacances) au moins un client dans le système (vacances multiples), ou bien il prend une seule vacance après chaque période d'activité (vacance unique) Dans la seconde situation le serveur peut prendre une vacance même si le système n'est pas vide. Parmi les disciplines de ce service , on trouve par exemple, le service limité, dans ce cas, le serveur sert un nombre k de clients au maximum, donc il sert jusqu'à ce que la file soit vide ou les k clients sont servis et le service avec barrière dont le serveur sert seulement les clients qui étaient dans la file d'attente à son arrivée, avant de pouvoir débuter une autre vacance.

Le passage de l'état inactif à l'état actif est contrôlé par les politiques de retour de vacance ou fin de vacance comme : N-vacance (la taille de la file d'attente >=N clients), T-vacance (la période T est expiré et il y a au moins un client dans la file d'attente), ou la durée d'une vacance aléatoire est expiré (une seule à la fois).

- **La troisième partie** est le départ : le client quitte le système après l'achèvement du service.

4.3 Conception du simulateur

Au sein du simulateur, Nous avons choisi d'appliquer les techniques de la simulation à événements discrets, qui est l'une des méthodes de simulation les plus utilisées, notamment dans les systèmes de files d'attente. Cette méthode, largement utilisée durant cette dernière

décennie, se base sur la modélisation de systèmes dans lesquels le changement d'état s'effectue à des instants discrets de l'axe temporel. Elle permet ainsi d'étudier le comportement d'un système à travers quelques périodes, en construisant un système (modèle) ayant la même structure que l'original (objet de l'étude) mais plus simple à manipuler. [28]

Les principaux concepts de la simulation à événement discret que nous avons utilisé sont :

- **Le system étudié** : standard.
- **Le model proposé** : une file d'attente avec vacance A/B/1 où A, B appartiennent à l'ensemble des distributions {Déterministe, Exponentielle, générale}.
- **Entités** : serveur, client.
- **Attributs** : actif, inactif.

Parmi les différentes approches de simulation à événements discrets que nous avons introduites (chapitre 3), c'est l'approche de planification d'événements *« Event Scheduling Approach »* .Le but de cette approche est d'être rapide et précise.

4.3.1 Les variables

Les variables (paramètres) nécessaires utilisées et leurs descriptions sont décrits dans le tableau 4-1 ci-dessous.

Nom de la variable	**Description**
horloge	C'est la variable globale représentant le temps simulé, l'ordonnanceur est responsable de l'avancement de ce variable
FEL	La liste triée d'événements (Futur Event List)
Batch	La liste pour stocker les statistiques
QUEUE	La file d'attente des clients
State	Représente l'état instantané du serveur : libre ; occupé ; en vacances
N	Le seuil pour quitter l'état de N vacance
T	Le seuil pour quitter l'état de T vacance
limite	Nombre de clients servis
K	La taille maximale de la file d'attente cas fini
λ	Le taux d'arrivée
μ	Le taux de service
τ	Le taux de vacance aléatoire

$\mathbf{C_h}$	Coût par unité de temps lorsqu'un client se joint à la file d'attente et attend le service
$\mathbf{C_v}$	Coût par unité de temps lorsque le serveur est en vacance
$\mathbf{C_b}$	Coût par unité de temps lorsque le serveur est occupé
$\mathbf{C_s}$	Coût par unité de temps de transition de l'état inactif à l'état occupé

Tableau 4.1 : Tableau des variables.

4.3.2 Evénements

Dans la simulation à événements discrets, il est nécessaire de s'assurer que les événements se produisent dans le bon ordre et au moment approprié, pour cela, nous avons garanti la planification des événements par une liste triée selon l'heure nommée « FEL ». Chaque élément de la liste contient l'heure à laquelle l'événement doit se produire et le type d'événement qui doit être exécuté à ce moment.

Pour mieux comprendre le fonctionnement du simulateur, dans ce qui suit. Nous allons présenter les déférents types événements que nous avons utilisés dans la simulation.

- ✓ **L'événement arrivé**

Représente l'arrivé d'un client au système, il est généré pour la première fois lors de l'initialisation comme une condition de départ pour planifier le premier client. Les autres événements d'arrivées sont générés aléatoirement avec des intervalles distribués de manière exponentielle ou générale ou déterministe avec une moyenne « $1/\lambda$ » entre eux.

- ✓ **L'événement début du service**

Représente l'entrée d''un client au serveur et le début de son traitement. Début du service est généré lorsqu'une arrivée du client si le serveur est vide ou bien lors de départ d'un client si la file d'attente n'est pas vide.

- ✓ **L'événement départ**

Représente la sortie d'un client du système, l'événement départ est généré lorsqu'un client finit sa période de traitement au serveur et quitte le système.

- ✓ **L'événement fin de vacance**

Nous avons ajouté un type d'événement spécial pour notre modèle proposé de la politique vacance-aléatoire et T-vacance, c'est l'événement fin de vacance. Cet événement représente fin de période de vacance, il est généré lorsqu'un client quitte le système et la file d'attente est vide (le serveur passe au vacance), ce type est conçu pour réveiller le serveur après avoir pris des vacances d'une durée aléatoire.

4.3.3 Les Routines

Au sein du simulateur, Le programme de simulation contient plusieurs routines telles que, initialisation, boucle principale et statistiques… etc. Ainsi, chaque événement est simulé par sa routine. Les routines se présentent comme étant des fonctions pour mettre à jour les variables d'état du système et planifient l'occurrence d'événements. Nous allons les présenter dans les sections suivantes et les expliquer avec des pseudos algorithmes.

- ✓ **Routine Arrivée**

C'est la routine exécutée par l'événement « arrivée ». En générale, Cette routine présente l'arrivé d'un nouveau client pour l'insérer dans la file et vérifier s'il y a possibilité de rendre le serveur actif. Le pseudo algorithme 4.1 ci-dessous montre la routine arrivé.

Routine Arrivée

```
0: Début
1: Si ((queue.length<K) && (type-queue == fini)) ou (type-queue == infini)
2:      Si (serveur ==0)
3:       Planifier l'événement= « début de service » : heure de l'événement=horloge ;
4 :     Sinon si (serveur == 1)
5:              Enfiler le client arrivé ;
6 :              Queue.length++ ;
7 :             Sinon Si (serveur ==2)//en vacance
8 :                     Si(type-vacance==N-vacance=)&&(queue.length==N)//le seuil à été atteint
9:                    Rendre le serveur actif ;

10:                   Planifier l'événement= « début de service » : heure de l'événement=horloge ;
11:                    Fin.
12:          Fin.
13:     Fin.
14: Fin.
15:Planifiez le prochain événement « arrivé »:heure de l'événement=horloge+période inter-arrivé ; // Les périodes sont générer aléatoirement de manière distribué exponentielle ou générale ou déterministe avec la moyenne « 1/λ » ;
16: Fin.
```

Pseudo algorithme 4.1 : la routine « Arrivée ».

✓ **Routine début de service**

C'est la routine exécutée par l'événement « début de service ». Elle sert à mettre le serveur occupé pour servir un client depuis la file et le faire planifier un événement de départ. Le 4.2 pseudo algorithme ci-dessous montre la routine début de service.

Routine début de service

0: Début

1: Définir le serveur occupé ;

2: Défiler le client du la queue ;

3: Planifiez un événement « départ » pour ce client : heure de l'événement=horloge+période de service ;

4: //les temps sont générer aléatoirement de manière distribué exponentielle ou générale ou déterministe avec la moyenne « 1/μ » ;

5: Fin.

Pseudo algorithme 4.2 : la routine « début de service ».

✓ **Routine départ d'un client**

C'est la routine exécutée par l'événement « départ ». En générale, cette routine met le serveur en état libre, si la file est vide, elle le met en vacances, de plus, elle fait planifier les événements nécessaires. Le pseudo algorithme 4.3 ci-dessous représente la routine départ d'un client.

Routine départ d'un client

0: Début

1: Mettre le serveur libre ;

2: Si (queue.length ==0) ou ((nombre de client servis == limite) && (type-vacance == aléatoire))

3: Mettre le serveur en vacance ;

4 : si (type-vacance == aléatoire) ou (type-vacance == T-vacance) alors

5 : planifier l'événement « fin de vacance » : heure de l'événement=horloge + période de vacance // pour vacance aléatoire les périodes sont générer aléatoirement de manière distribué exponentiellement avec la moyenne « 1/τ »

6 : Fin.

7: Sinon

8: Planifier l'événement « début de service » : heure de l'événement=horloge ;

9: Fin.

10: Queue.length−− ; //décrémente le nombre de clients

11: Fin.

Pseudo algorithme 4.3 : la routine « départ ».

- ✓ **Routine de fin de vacance**

C'est la routine exécutée par l'événement « fin de vacance ». Elle est conçue pour indiquer que le serveur a fini ses vacances et vérifier s'il y a possibilité d'avoir de nouvelles vacances. Le pseudo algorithme 4.4 ci-dessous représente la routine fin de vacance.

Routine fin de vacance

0 : Début

1 : Si ($x \leq 0$) alors

2 : planifier un nouvel l'événement « fin de vacance » : heure de l'événement=horloge + période de vacance // pour vacance aléatoire les périodes sont générer aléatoirement de manière distribué exponentiellement avec la moyenne « $1/\tau$ » et pour T-vacance la période donné par l'utilisateur.

3 : Sinon planifier un événement « début de service » : heure de l'événement = horloge ;

4 : Fin.

5 : Fin.

Pseudo algorithme 4.4 : la routine « fin de vacance ».

- ✓ **Routine d'initialisation**

Cette routine contient tous les paramètres nécessaires du system saisis par l'utilisateur et leurs initialisations qui reflètent l'état du système au temps zéro tell que l'initialisation de l'horloge et mettre la file vide, les paramètres liés à le coût et préciser le taux d'arriver et le taux de service, le seuil, le taux de vacance. etc.de plus, un événement de type Arrivé doit être planifié comme une condition de départ. La routine d'initialisation permet généralement de faire varier un paramètre d'une manière spécifiée.

Routine d'initialisation

0: Début ;

1: Définir horloge =0 ;

2: Définir queue = 0 ;

3: Définir le serveur en vacance ;

4: Définir la liste d'événement « FEL » à vide ;

5: Récupérer et initialiser les valeurs de : limite, K, N, T, λ, μ, τ ;

6: Récupérer et initialiser les valeurs de : C_h, C_v, C_b, C_s ;

7: Planifiez un événement « arrivée » à l'instant 0 ;

8: temps de saturation = 0 ;

9 : Fin.

Pseudo algorithme 4.5 : la routine « Initialisation ».

- ✓ **Routine de la boucle principale**

La routine de la boucle principale rassemble toutes les autres routines.au début, Elle fait un appel à la routine d'initialisation et elle termine par appeler la routine de sortie statistique. A l'intérieur de la boucle elle fait appeler une routine en fonction de type d'événement précisé dans la liste d'événement « FEL » durant la simulation avec un temps prédéfinie.

Routine de la boucle principale

0 : Pour (i =1 ; i≤10 ; i++)

1 : initialisation () ;

2: Tant que (horloge ≤1000)

3: Get_événement () ; //récupérer l'événement

4: Défiler le prochain événement de la liste d'événement « FEL » ;

5: Horloge = l'heure de l'événement ;

6: Appeler une fonction d'exécution de routine en fonction du type d'événement ;

7: Fin de tant que ;

8:Appeler la fonction d'exécution de routine statistiques ;

9 : Stocker les statistiques dans une liste « batch » ;

10 : Fin pour.

11 : Vérifier () ;

12 : libérer batch () ;

13:Fin.

Pseudo algorithme 4.6 : la routine « boucle principale ».

Pendant une simulation, nous pouvons bien sûr nous intéresser à la trace d'une simulation pour comprendre les comportements transitoires. Mais la plupart du temps, nous nous intéressons plutôt au comportement d'un système à long terme, autrement dit, à son régime stationnaire.

Il existe de nombreuses méthodes utilisées pour recueillir des données d'observation de l'état stationnaire au cours d'une simulation. Dans cette mémoire, nous présentons la méthode la plus simple. Il s'agit de la Méthode des blocs (Subinterval ou batch means).Nous avons choisi cette méthode car la plus simple et un des avantages de cette méthode est qu'un seul intervalle transitoire doit être pris en compte, actualisé et supprimé pendant le processus d'enregistrement des observations.

Considérons le temps de simulation est de 10000 unité de temps, découpée en N=10 fois de taille de l'intervalle m=1000 unité de temps. après chaque fin d'intervalle m de simulation on appelle la fonction des statistiques et on stock le résultats de statistiques (la valeur de l'indice) dans une liste appeler « batch », à la fin on fait l'appel de la méthode Vérifier () comme le montre dans le pseudo algorithme 4.7 ligne 11, cette méthode parcours la liste « batch » jusqu'à ce que la condition suivante est vérifiée :

Valeur absolue (valeur de l'indice (i) – valeur de l'indice (i-1)) < 0.000001

Si cette condition est vérifiée donc le système est en régime stationnaire.

Le pseudo algorithme 4.7 représente la méthode Vérifier () :

Vérifier ()

```
0 : i=1 ;
1 : Tant que (la liste « batch » n'est pas vide)
2 :     résultat =Math.abs (batch(i) – batch (i-1) ;
3 :     Si (résultat > 0.000001)
4 :         supprimer le premier élément de la liste « batch » ;
5 :     i++ ;
6 : Si (la liste « batch » n'est pas vide)
7 :     récupérer la valeur de premier élément de la liste « batch » ;
```

Pseudo algorithme 4.7 : la méthode « Vérifier ».

- ✓ **Routine statistique**

C'est la routine de sortie qui est exécutée à la fin de la simulation. Elle est conçue pour donner des résultats des mesures et statistiques finales de la simulation. Les résultats se sont des mesures qui seront détaillés dans la dernière section de ce chapitre.

4.3.4 Les Diagrammes UML

Les diagrammes UML structurels sont utilisés pour raisonner le comportement du système et présenter les conceptions proposées et communiquer avec les parties prenantes ainsi que pour piloter la mise en œuvre. Parmi ces diagrammes, nous allons nous focaliser sur le diagramme de classes et diagramme de cas d'utilisation qui analyse ou améliore la conception du système. Il est considéré comme le plus important de la modélisation orientée objet.

- ✓ **Diagramme de classes :**

Le diagramme de classes fait abstraction des aspects temporels ou dynamiques. Il identifie les classes et les associations de notre système. Le diagramme de classes et sa description sont illustrés respectivement dans la figure 4.2 et le tableau 2 suivants :

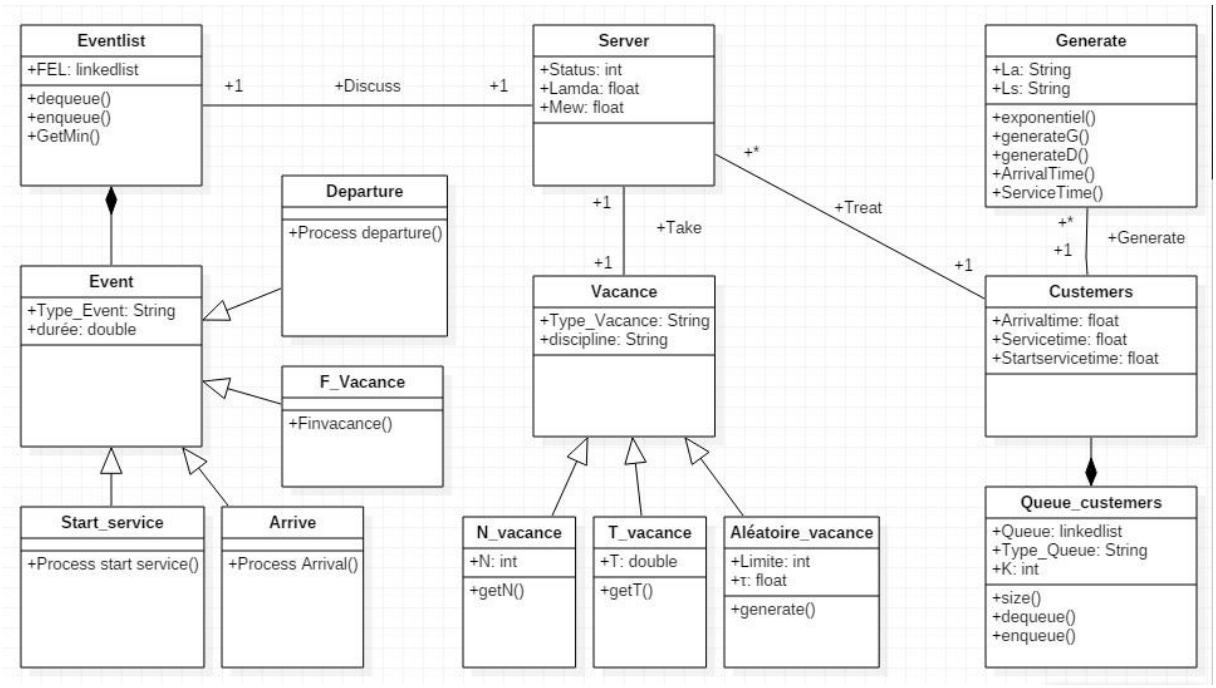

Figure 4.2 : Diagramme de classes.

Classe	**Attribut**	**Type**	**Désignation**	**Méthode**
Generate	LA	string	Loi d'arrivé .	Exponential() generateG () generateD () arrivalTime() serviceTime()
	LS	string	Loi de service.	
Custemers	Arrivaltime	float	Le temps d'arrivé de client.	

	Servicetime	float	Le temps de service de client.	
	Startservicetime	float	Le temps de début de service de client.	
Queue-custemers	K	int	La taille maximale de la file	Size() Dequeue() Enqueue()
	queue	Linked-list	La file d'attente des clients.	
	Type-queue	String	Type de la file d'attente : fini ou infini.	
server	status	int	L'état de serveur 0 : libre /1 : occupé /2 : vacance.	
	lamda	float	Taux d'arrivé.	
	mew	float	Taux de service.	
vacance	Type-vacance	string	Le type de vacance .	
	discipline	string	La discipline de vacance :exhaustif/non exhaustif.	
N-vacance	N	int	Le seuil.	Get-N()
T-vacance	T	double	période de vacance.	Get-T()
Hybride-vacance	limite	int	Le seuil.	Generate()
	τ	float	période de vacance.	
Event-list	FEL	linkedList	Liste d'événement triée selon le temps d'arrivée	Dequeue() Enqueue() GetMin()
Event	Type-event	string	Type d'événement :arrivée/start-service/departed .	
	durée	double	Le temps d'arrivée d'événement.	
Start-service			L'événement début de service	Process start service ()
Departure			L'événement départ	Process departure()
Arrive			L'événement arrivé	Process arrival()
Fin-vacance			L'événement fin de vacance	Finvacance ()

Tableau 4.2 : Tableau descriptif des classes.

- ✓ **Diagramme de cas d'utilisation :**

Comme son nom l'indique, un cas d'utilisation représente une fonctionnalité du système. Ce diagramme nous permet d'identifier toutes les possibilités d'interaction entre le système et les acteurs, ce qui veut dire toutes les fonctionnalités que le système doit fournir. La figure 4.3

suivante représente le diagramme de cas d'utilisation pour le système de file d'attente avec vacance:

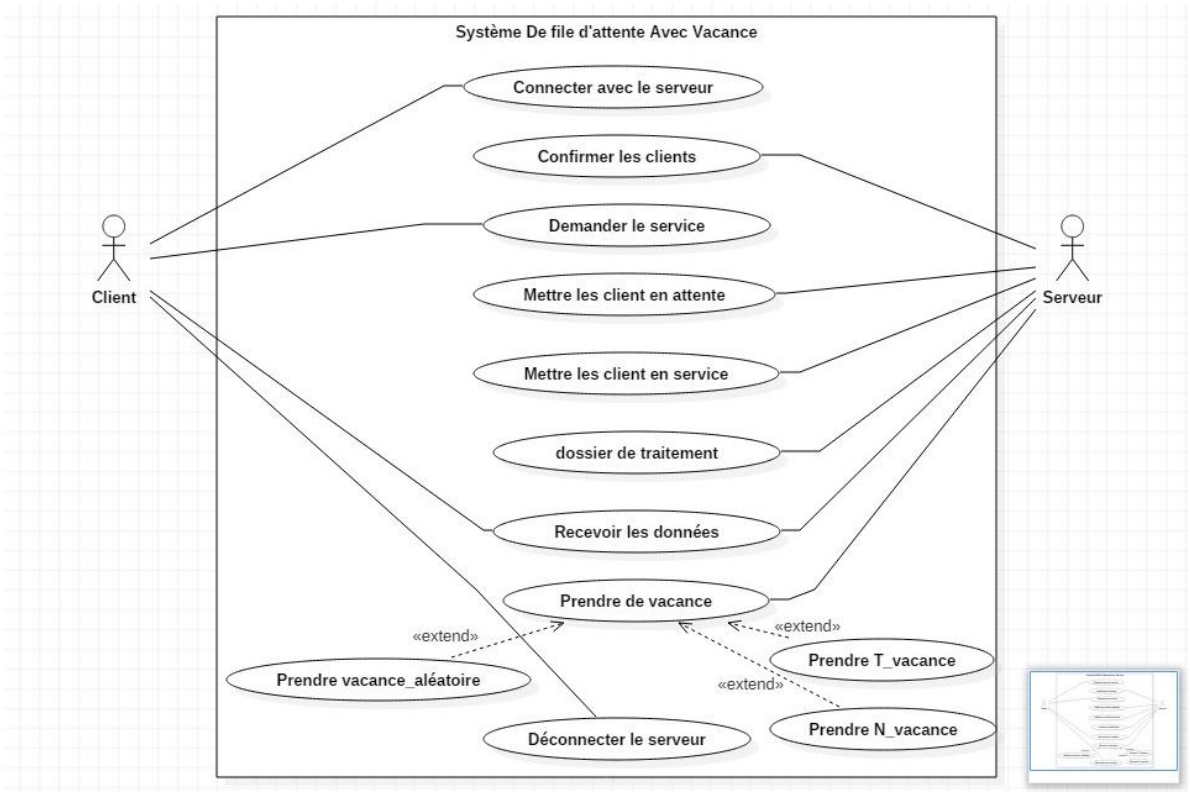

Figure 4.3 : Diagramme de cas d'utilisation.

1. **Identification des acteurs :**

Un acteur est une personne interagissant avec le système. Chaque acteur est identifié par un rôle. Ces rôles décrivent les capacités et les besoins de l'acteur. Les personnes interagissant avec le système de file d'attente avec vacance sont :

- le client,
- le serveur.

2. **Description des cas d'utilisation :**

- **Mettre les clients en attente :**

Lors de l'arrivé d'un client s'il trouve un serveur occupée ou inactif (en vacance), il se place directement dans une file d'attente qui peut être d'une capacité fini ou infini, il attend son tour selon une politique spécifier (FIFO, LIFO, priorité …).

- **Mettre les clients en service :**

Lors de l'arrivé d'un client s'il trouve un serveur libre et dans l'état actif donc il passe directement au service. Ou après le départ d'un client servi, le serveur est libre,

si la file d'attente n'est pas vide et la condition de début de vacance n'est pas encore vérifier donc le premier client de la file d'attente va être en service.

- **Prendre de vacance :**

À la fin de service d'un client le serveur vérifier la condition de début de vacance d'un serveur si la file d'attente est vide (pour un service exhaustif) ou si le nombre de clients servis est égal à K client (pour un service exhaustif limité) donc il part en vacance soit une vacance d'une durée aléatoire ou N vacance ou T vacance .

4.4 Les indices de performances

Après avoir vue les différents aspects nécessaires liée à la conception du simulateur, nous devons également connaitre les principaux indices de performance des modèles de file d'attente utilisé, Ces indices sont présentés dans les sections suivantes.

4.4.1 Débit

Il est défini comme le taux des clients qui peuvent être traitées par le serveur. Autrement, il correspond à la fraction du nombre des clients ayant été servis sur le temps total de simulation, Le débit est mesuré en clients par unité de temps, qui est généralement « seconde». La manière de calculer sa valeur est illustré dans le pseudo algorithme 4.8.

Débit

Lors de l'événement de départ

0:Nombre de client servi++ ;

À la fin de la simulation

1:Débit = nombre de client servi/temps total de simulation ;

2:Fin.

Pseudo algorithme 4.8 : calcule de débit .

4.4.2 Taux de perte

Il correspond au cas où un nouveau client ne peut être pas mettre à cause de la saturation de la file d'attente. Autrement, c'est la fraction de nombre des clients ayant été rejeté sur le nombre total des clients. Pour calculer sa valeur le pseudo algorithme 4.9 montre une suite d'instruction à suivre.

Taux de perte

Lors de l'événement d'arrivé

0:Nombre de client arrivé++ ;

1:Si (queue.length == K)//client est rejeté car la file est saturée (cas de la taille fini)

2:Alors nombre de client perdu++ ;

3:Fin.

A la fin de la simulation

4:Taux de perte = Nombre de client perdu / Nombre de client arrivé ;

5:Fin.

Pseudo algorithme 4.9 : calcul de taux de perte .

4.4.3 Probabilité de blocage

Correspond à la probabilité que la capacité de la file d'attente atteigne sa limite K (taille maximale de la file d'attente dans le cas fini) durant la simulation. En d'autres termes, Elle représente la fraction des temps auxquels le système a connu une saturation sur le temps total de la simulation. Une suite d'instruction dans le pseudo algorithme 4.10 illustrant la manière pour calculer sa valeur.

Probabilité de blocage (saturation de la queue c'est la taille de la file fini)

Lors de l'événement d'arrivé

0:Si (queue.length ==K) //la queue est saturée

1:Alors début saturation=horloge ;

2:Fin.

Lors de l'événement de départ

3:Temps de saturation+= (horloge − début saturation) ;

À la fin de simulation

4:Blocage = temps saturation / temps total de simulation ;

5:Fin.

Pseudo algorithme 4.10: calcule de probabilité de blocage .

4.4.4 Probabilité que l'unité de transmission soit en vacances « P_V »

Celle-ci correspond à la probabilité que le serveur soit à l'état de vacance. Si aucun client se trouve dans la file d'attente (service exhaustif) ou k clients sont servis (service non exhaustif) le serveur prend des vacances après un départ de client. Autrement, Elle représente la fraction des temps de vacances sur le temps total de simulation. Pour plus de détaille le pseudo algorithme 4.11 ci-dessous montre bien la manière pour calculer cette probabilité.

Probabilité de vacance «P_v »

Lors de l'événement de départ

0:Si (queue.length==0)ou((nombre de client servis == limite)&&(type de vacance=aléatoire))

1:Début vacance = horloge ;

2: Fin.

Lors de l'événement d'arrivé

3:Si (queue.length ==N)

4:Total vacance += (horloge − début vacance) ;

5 : Fin.

Lors de l'événement de Fin de vacance

6 : si (queue.length>0) alors

7 : total de vacance += (horloge − début vacance)

8 : Fin.

À la fin de simulation

9: P_V =total de vacance /temps total de simulation

10:Fin.

Pseudo algorithme 4.11 : calcul de probabilité de vacance .

4.4.5 Temps d'attente moyenne

Cette quantité est estimée comme la moyenne arithmétique de tous les temps d'attente des clients observés. Cependant, pour calculer cette estimation, nous devons mesurer pour chaque client le temps écoulé depuis son arrivée jusqu'au temps de son début de service. Autrement, Elle représente la fraction des temps d'attente pour chaque client sur le nombre total des clients servis.

Temps d'attente moyenne

Lors de l'événement début de service

0 : temps d'attente += horloge – le temps d'arrivé de client ;

Lors de l'événement de départ

1 : nombre de client servi++ ;

À la fin de la simulation

2 : moyenne d'attente = temps d'attente / nombre de client servi ;

3 : Fin.

Pseudo algorithme 4.12 : calcul de temps d'attente moyen.

4.4.6 Temps de réponse moyenne

Représente le temps moyen passé par un client depuis l'instant de son arrivé jusqu'à l'instant de son départ du serveur. Ceci est illustré dans le pseudo algorithme 4.13 ci-dessous.

Temps de réponse moyenne

Lors de l'événement de départ

0 : temps de répons += horloge –le temps d'arrivé de client ;

1 : nombre de client servi++ ;

À la fin de la simulation

2 : moyenne de réponse = temps de réponse /nombre de client servi ;

3 : Fin.

Pseudo algorithme 4.13 : calcul de temps de réponse moyen.

4.4.7 Le coût

Ils définissent une fonction de coût par unité de temps avec les éléments de coût suivants :

C_h : Coût par unité de temps lorsqu'un client se joint à la file d'attente et attend le service.

C_v: Coût par unité de temps lorsque le serveur est en vacance.

C_b: Coût par unité de temps lorsque le serveur est occupé.

C_s: Coût par unité de temps de transition de l'état inactif à l'état occupé.

Selon les définitions ci-dessus des éléments de coût, la fonction de coût prévue par unité de temps est calculée à la fin de la simulation comme suit :

$$C = P_v * C_v + (1 - P_v) * C_b + C_s / N_c + C_h * Q$$

Pour calculer le coût, nous aurons besoin de calculer la langueur moyenne de la file d'attente et le nombre de cycles suivants :

- **La longueur moyenne de la file d'attente (Q)**

Elle représente le nombre moyen de clients en attente dans la file d'attente.

La longueur moyenne de la file d'attente (Q)

Lors de la boucle principale

0 : défiler l'événement ;

1 : T[x]+= le temps d'événement – horloge ; // x : le compteur de client dans la queue

À la fin de la simulation

2 : pour tout pair<clé, valeur>S ∈ T

3 : Q=S.clé * S.valeur ;

4 : Fin.

Pseudo algorithme 4.14 : calcul de la longueur moyenne de la file.

- **Le nombre de cycles « N_c »**

Il correspond au nombre de transitions de l'état inactif à l'état occupé par unité de temps. Pour plus de détaille le pseudo algorithme 4.15 présente la manière de calculer cette variable.

Le nombre de cycles N_c

Lors de l'événement d'arrivée

0:Si (queue.length==N)

1:Nbrcycles++ ; // le nombre de cycles

2:Fin.

À la fin des périodes des vacances//pour aléatoire et T-vacance

3:Nbrcycles++ ;

4:Fin.

À la fin de la simulation

5:Nombre de cycles = nbrcycles/horloge.

6:Fin.

Pseudo algorithme 4.15: calcul de nombre de cycles.

4.5 Conclusion

Dans ce chapitre nous avons présenté une modélisation des files d'attente avec vacance, puis une conception sur les déférents aspects important liés au simulateur et à l'approche de simulation à événement discret. Ensuite, nous avons présenté les indices de performances importantes pour l'évaluation des performances d'un système de file d'attente avec vacance.

Dans le prochain chapitre, nous présenterons les divers tests des différents modèles de files d'attente avec vacance qui montent l'impact des différents paramètres des modèles sur la performance du système.

Chapitre 5

5. Implémentation et étude expérimentale

5.1 Introduction

L'objectif principal de ce chapitre est l'implémentation d'une application pour l'évaluation des performances des systèmes de files d'attentes avec vacance. Cette application implémente les algorithmes proposés dans le chapitre précédant en fonction des paramètres du système introduit par l'utilisateur. De plus, elle affiche les différents indices de performances des modèles des systèmes de files d'attentes avec vacance.

Tout d'abord, nous commençons ce chapitre par la présentation des différentes fonctionnalités de notre application. Ensuite, une étude expérimentale pour montrer l'influence de certains paramètres des systèmes comme le taux d'arrivé des clients et le taux de service, sur les différents indice de performance d'un système pour chaque politique de vacance et pour chaque modèle de file d'attente. Nous terminons par une conclusion

5.2 Présentation de l'application

5.2.1 Choix du langage Java

L'implémentation de notre application est réalisée sous le système ***Windows***, en utilisant l'outil Java de ***Netbeans***. Ce dernier est un langage de programmation largement utile, simple, solide, orienté objet, et performant. Ses principaux avantages sont les suivants :

- Il est simple et facile de modéliser un système réel.
- Java a été conçue pour que les programmes qui l'utilisent soient fiables sous différents aspects.
- Plus rapide à exécuter.
- Plus de flexibilité de programmation.
- Il vous permet de créer des projets standards et du code réutilisable.
- Il est sécurisé.

5.2.2 Fenêtre d'accueil

La fenêtre d'accueil dans la figure 5.1 est la première fenêtre qui apparue lors du lancement de l'application elle contient deux boutons :

- **Démarrer :** pour lancer la fenêtre de choix des différentes paramètres de système.
- **Quitter :** pour quitter l'application.

Figure 5.1 : La fenêtre d'accueil de l'application.

5.2.3 Fenêtre de choix

Cette fenêtre qui est illustrée dans la figure 5.2 permet à l'utilisateur de sélectionner les paramètres concernant leur système telles que la discipline service (exhaustif, non exhaustif), politique de fin de vacance (T vacance, N vacance , aléatoire), type de la file(finie, infinie), loi d'arrivée(exponentielle, général, déterministe), loi de service (exponentielle , général ,déterministe) , et le nombre de serveurs. Ensuite, l'utilisateur doit choisir l'un des paramètres afin d'étudier l'influence de la variation de ce dernier sur les indices de performances. Cette fenêtre contient deux boutons :

- **Appliquer** : pour lancer la simulation.
- **Retour** : pour revenir à la page d'accueil.

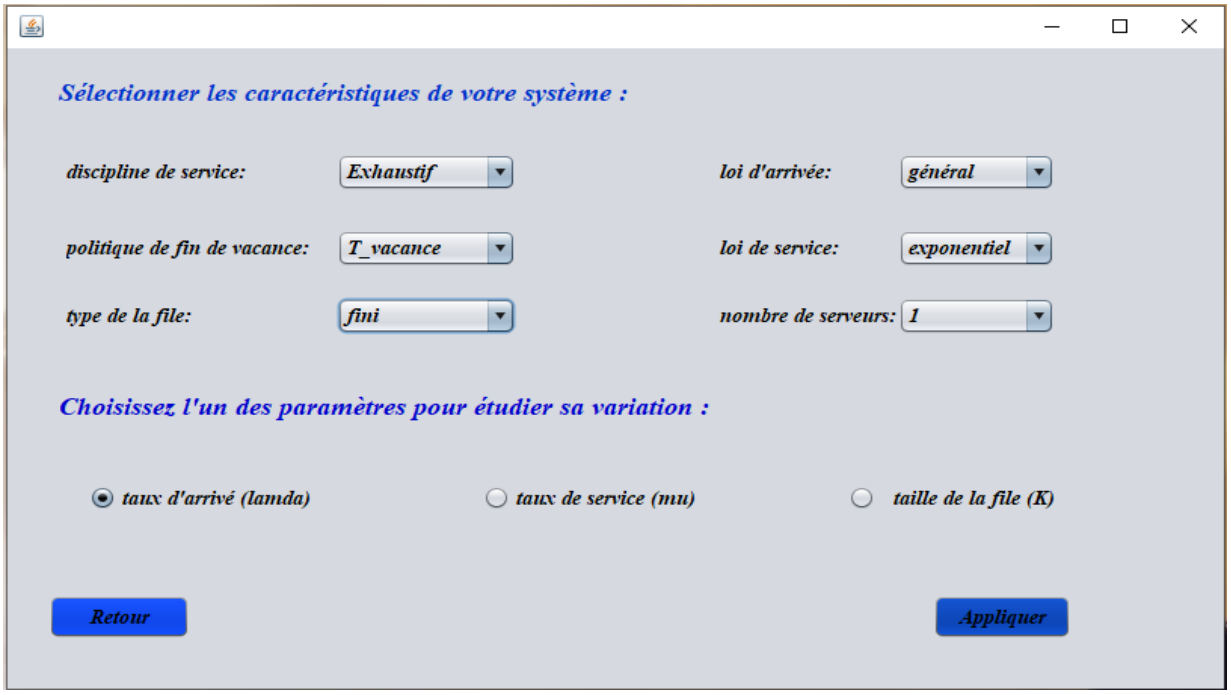

Figure 5.2 : La fenêtre de choix.

5.2.4 Fenêtre des résultats

Cette fenêtre qui est illustrée dans la figure 5.3 permet d'afficher les résultats obtenus de la simulation, elle présente les paramètres de système entrée précédemment et les graphes pour chaque indice de performance.

Cette fenêtre contient les boutons :

- **Retour** : pour revenir à la page de fenêtre de choix et lancer une nouvelle simulation.
- **Quitter** : pour quitter l'application.
- **P-vacance :** permet d'afficher le graphe de la probabilité de vacance.
- **Débit :** permet d'afficher le graphe de débit.
- **Taux de perte :** permet d'afficher le graphe du taux de perte.
- **Temps d'attente moyen :** permet d'afficher le graphe du temps d'attente moyen.
- **Temps de réponse moyen :** permet d'afficher le graphe du temps de réponse moyen.
- **P-blocage :** permet d'afficher le graphe de la probabilité de blocage du système.
- **P-server occupé :** permet d'afficher le graphe de probabilité de server occupé.
- **Le coût :** pour afficher le graphe de coût du système.

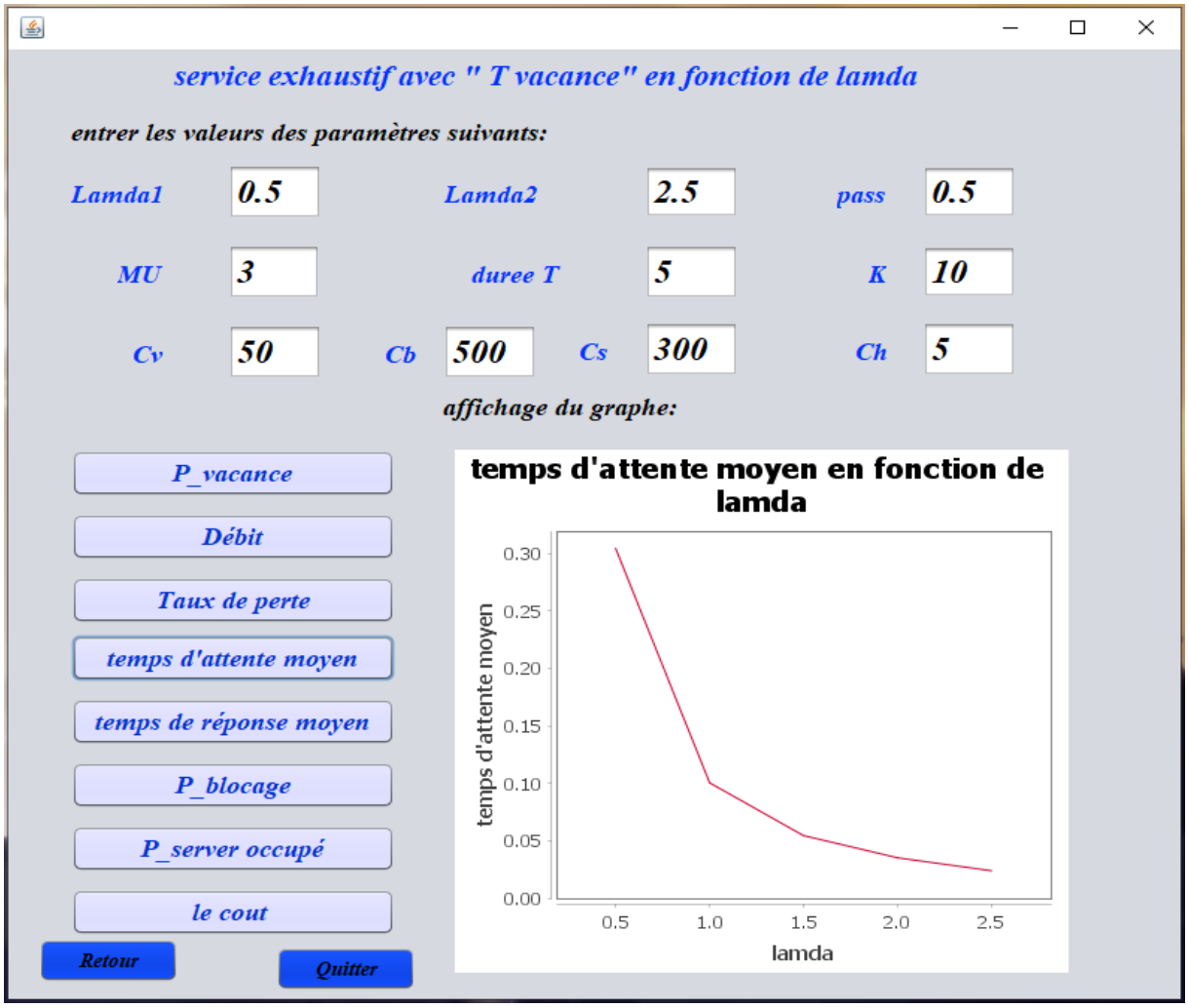

Figure 5.3 : La fenêtre des résultats.

5.3 Génération des événements suivant une loi de probabilité

Le modèle de simulation doit reproduire des événements aléatoires : arrivée d'un client dans le système, temps de service d'un client, etc. Le modèle de ces comportements consiste en variables aléatoires de loi connue. En programmation, la base pour générer une variable aléatoire qui suit une loi quelconque est l'utilisation de la fonction Random() qui permet de tirer des valeurs uniformément réparties entre 0 et 1.

Pour illustrer le principe, prenons le cas d'une variable aléatoire : X

Plusieurs méthodes existent pour générer des variables aléatoires. Nous présentons la méthode par fonction inverse qui est générale est facile à programmer. Cette méthode consiste à :

- tirer une variable μ uniformément distribuée entre 0 et 1
- prendre $X = F^{-1}(u)$ tel que $u = F(x)$

X suit alors la loi cherchée.

Preuve: il suffit de vérifier que $P[X \leq x] = F(x)$

$$P[X \leq x] = P[F^{-1}(u) \leq x] = P[u \leq F(x)] = F(x)$$

La dernière égalité s'obtient car u est uniformément distribué entre 0 et 1, ce qui donne : $P[u \leq a] = a; (0 \leq a \leq 1)$

Notre modèle est basé sur les systèmes de files d'attente A/B /c, tel que « A » représente les temps inter-arrivées sont distribués de manière exponentielle ou générale ou déterministe et « B » représente les temps de service sont distribués de manière exponentielle ou générale ou déterministe. Les principales caractéristiques des distributions sont Implémentées avec le langage JAVA dans la figure 5.4. C'est fait en 3 fonctions :

```
public static double exponential(Random rng, double mean) {

    return Math.log( rng.nextDouble())/(-1*mean);
}

public static double generateG(Random rand, double var) {

    return var + rand.nextGaussian() * segma;
}

public static double generateD(double var) {
    return var;
}
```

Figure 5.4 : Implémentation d'un générateur des variables aléatoires.

La 1ère fonction génère les variables qui suivent la loi exponentielle.

Maintenant, examinons le cas de la loi exponentielle $F(x) = 1 - e^{-\lambda x}$

Soit u une variable aléatoire uniformément distribuée entre 0 et 1

On note : $F(x) = 1 - e^{-\lambda x} = u$.

La fonction inverse de F(x) s'obtient simplement par l'expression de x en fonction de u.

$e^{-\lambda x} = 1 - u$

$x = -ln(1-u)/\lambda$

Comme u peut être généré par la fonction Math.random() en programmation (en java), on a :

x = -Math.log(1-Math.random())/λ = - Math.log (Math.random())/λ.

La 2éme fonction génère les variables qui suivent la loi générale avec la méthode nextGaussian() est utilisée pour obtenir la prochaine valeur pseudo aléatoire, gaussienne

("normalement") distribuée double avec la moyenne 0.0 et l'écart type 1.0 de la séquence de ce générateur de nombres aléatoires.

La 3^{éme} fonction pour la loi déterministe qu'elle n'a pas besoin de générateur, elle utilise les temps des arrivées et les temps de service tels quels est.

5.4 Résultats numériques

Nous allons présenter les différentes tests des 3 modèles de file d'attente (G/G/1, M/M/1, D/G/1) après la simulation, pour montrer l'influence de certains paramètres des systèmes comme le taux d'arrivé des clients et le taux de service, sur les différents indices de performance : temps d'attente moyen et le débit. Les paramètres du système utilisés dans l'étude expérimentale sont répertoriés dans le tableau 5.1

Paramètre	Valeur
Capacité de la file d'attente (K)	10
Seuil de file d'attente (N)	5
Taux moyen d'arrivée (λ)	Entre 0.5 et 2.5
Taux de service moyen (μ)	Entre 2 et 2.8
Taux de vacances moyen (τ)	2
Seuil T de T-vacance	5
C_h	5
C_v	50
C_b	500
C_s	300
Temps de simulation maximal	10000

Tableau 5.1 tableau des paramètres du système.

5.4.1 Les trois modèles proposés

- ✓ **Le 1er modèle G/G/1 :**

C'est un modèle de file d'attente qui a un taux d'arrivé de distribution G (Général) et un taux de service de distribution G (Général) et un seul serveur, dans ce modèle on a utilisé une discipline de service : **Exhaustif,** avec une politique de fin de vacance : **T-vacance.**

- ✓ **Le 2eme modèle M/M/1 :**

C'est un modèle de file d'attente qui a un taux d'arrivé de distribution exponentielle et un taux de service de distribution exponentielle et un seul serveur, dans ce modèle on a utilisé une discipline de service : **Exhaustif,** avec une politique de fin de vacance : **N-vacance.**

- ✓ **Le 3eme modèle D/G/1 :**

C'est un modèle de file d'attente qui a un taux d'arrivé déterministe et un taux de service de distribution Général et un seul serveur, dans ce modèle on a utilisé une discipline de service : **Non Exhaustif,** avec une durée de vacance **aléatoire.**

5.4.2 Effet de la variation de taux d'arrivée des clients λ

Dans ce qui suit, Nous avons montré l'impact de la variation de taux d'arrivé λ sur quelques indices de performance : le temps d'attente et le débit en implémentant les 3 modèles proposés précédemment pour les deux cas de files d'attente (finie, infinie).

✓ **Le 1er modèle G/G/1 :**

Temps d'attente moyen :

1. **File d'attente finie :**

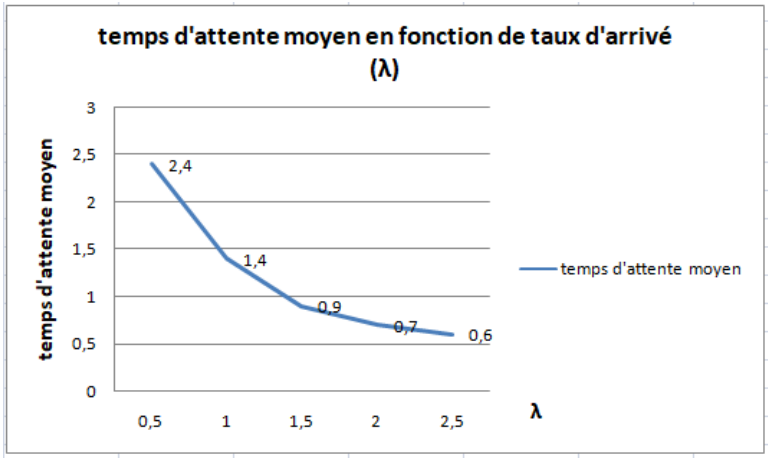

Figure 5.5 : Graphe de temps d'attente moyen dans une file d'attente finie G/G/1 en fonction de taux d'arrivé.

2. **File d'attente infinie :**

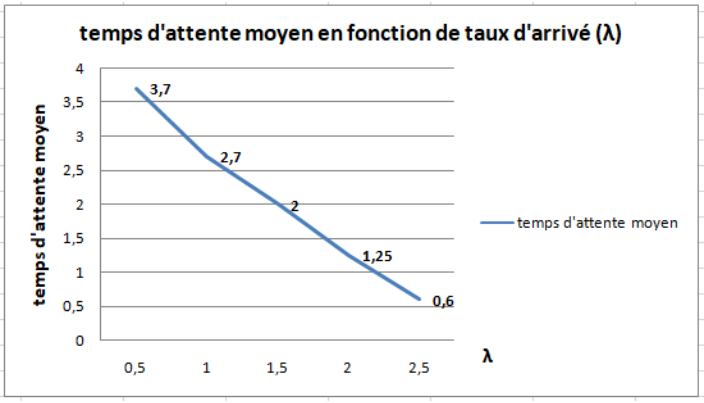

Figure 5.6 : Graphe de temps d'attente moyen dans une file d'attente infinie G/G/1 en fonction de taux d'arrivé.

.

- **Analyse :**

Nous remarquons que l'augmentation du taux d'arrivé diminue le temps d'attente des clients dans la file d'attente dans les deux cas (finie et infinie).

Le Débit :

1. **File d'attente finie :**

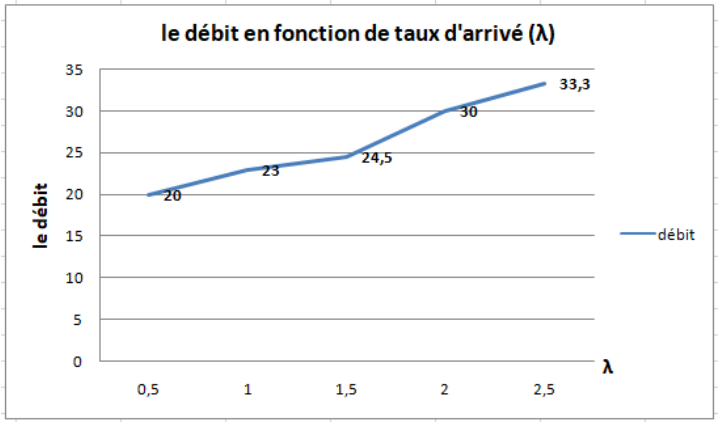

Figure 5.7 : Graphe de débit d'une file d'attente finie G/G/1 en fonction de taux d'arrivé.

2. **File d'attente infinie :**

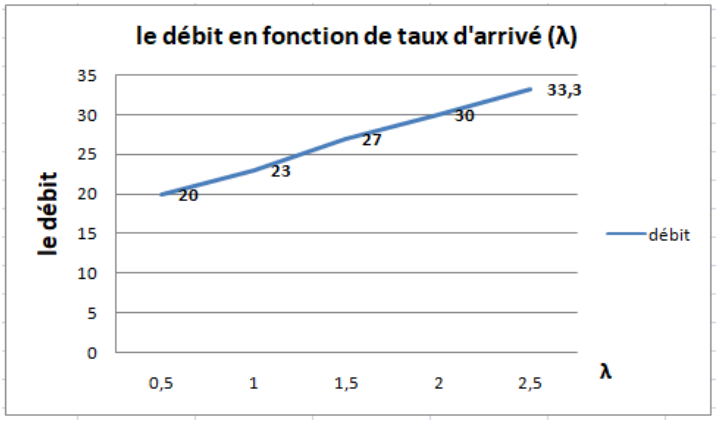

Figure 5.8 : Graphe de débit d'une file d'attente infinie G/G/1 en fonction de taux d'arrivé.

- **Analyse :**

Nous constatons que l'augmentation du taux d'arrivé influe sur le débit, il augmente dans les deux type de file d'attente (finie et infinie), car plus le taux d'arrivé augmente plus les clients de la file d'attente sont traités , ce qui implique l'augmentation de débit.

✓ **Le 2eme modèle M/M/1 :**

Temps d'attente moyen :

1. **File d'attente finie :**

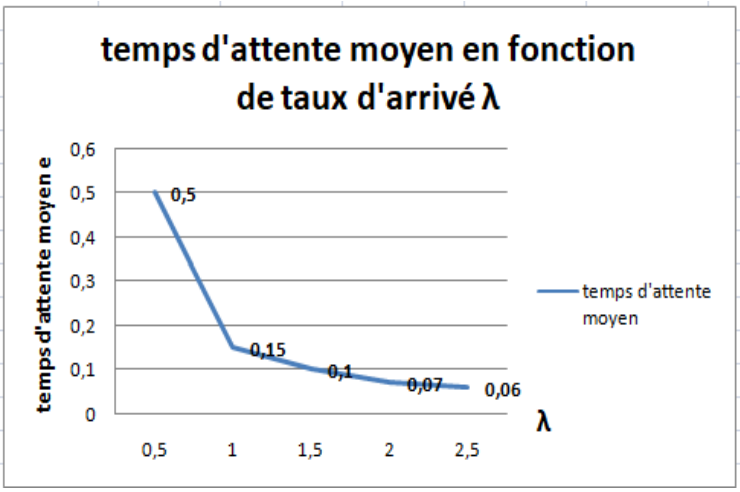

Figure 5.9 : Le temps d'attente moyen dans une file d'attente finie M/M/1 en fonction de taux d'arrivé.

2. **File d'attente infinie :**

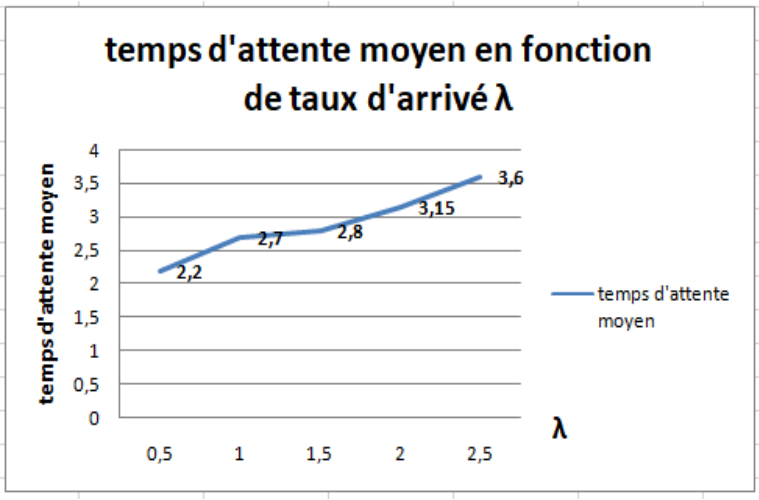

Figure 5.10 : Le temps d'attente moyen dans une file d'attente infinie M/M/1 en fonction de taux d'arrivé.

- **Analyse :**

Nous remarquons que l'augmentation de taux d'arrivé diminue le temps d'attente moyen dans une file de capacité finie, contrairement dans une file infinie on remarque que plus le taux d'arrivé augmente plus le temps d'attente augmente.

Le Débit :

1. **File d'attente finie et infinie :**

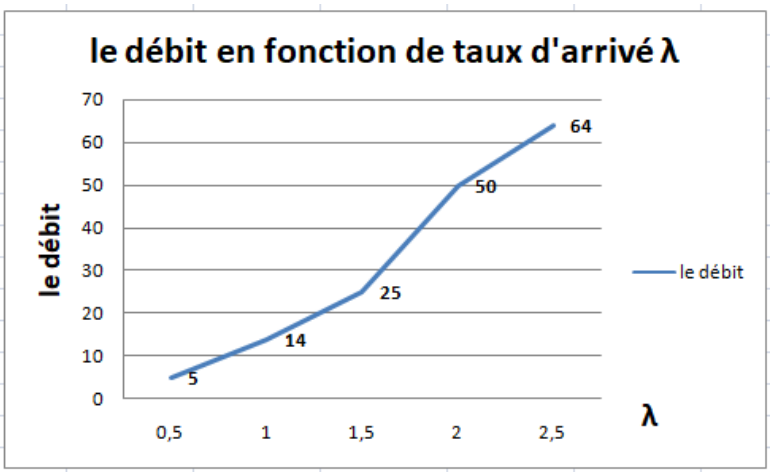

Figure 5.11 : Le débit d'une file d'attente finie et infinie M/M/1 en fonction de taux d'arrivé.

- **Analyse :**

Nous remarquons que l'augmentation de taux d'arrivé λ implique une augmentation de débit dans les deux types de file d'attente (finie et infinie).

Le débit est similaire pour les deux types de file d'attente.

✓ **Le 3eme modèle D/G/1 :**

Temps d'attente moyen :

1. **File d'attente finie :**

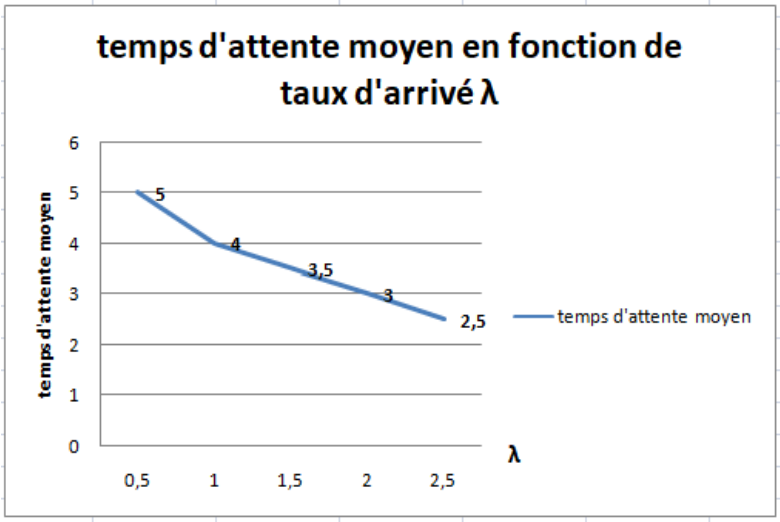

Figure 5.12 : Le temps d'attente moyen dans une file d'attente finie D/G/1 en fonction de taux d'arrivé.

2. **File d'attente infinie :**

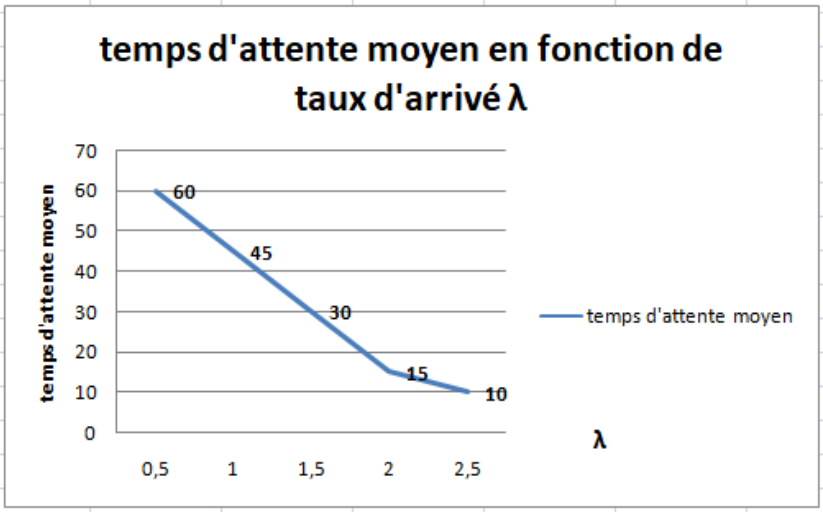

Figure 5.13 : Le temps d'attente moyen dans une file d'attente infinie D/G/1 en fonction de taux d'arrivé.

- **Analyse :**

Nous remarquons que L'augmentation du taux d'arrivé diminue le temps d'attente des clients dans la file d'attente dans les deux cas (finie et infinie), les valeurs de temps d'attente moyen dans le cas infinie sont plus élevé que les valeurs d'une file d'attente fini car cette dernière a une capacité limité contrairement dans une file infini qui reçoit un nombre infini de clients ce qu'il augmente le délai d'attente des clients dans la file.

Le Débit :

File d'attente finie et infinie :

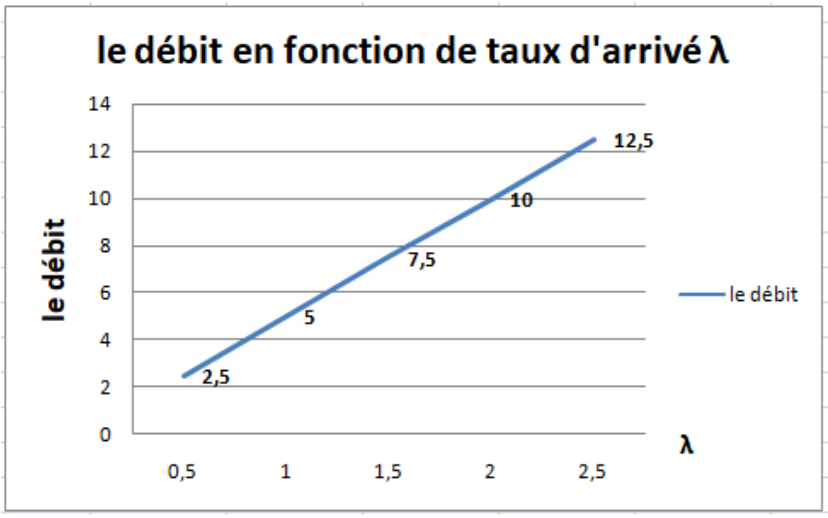

Figure 5.14 : Le débit d'une file d'attente finie et infinie D/G/1 en fonction de taux d'arrivé.

- **Analyse :**

Nous remarquons que l'augmentation du taux d'arrivé augmente le débit dans les deux cas (finie et infinie).

5.4.3 Effet de la variation de taux de service des clients µ

Dans ce qui suit, Nous avons montré l'impact de la variation de taux de service µ sur quelques indices de performance : le temps d'attente et le débit en implémentant les 3 modèles proposés précédemment pour les deux cas de files d'attente (finie, infinie).

- ✓ **Le 1er modèle G/G/1 :**

Temps d'attente moyen :

1. **File d'attente finie :**

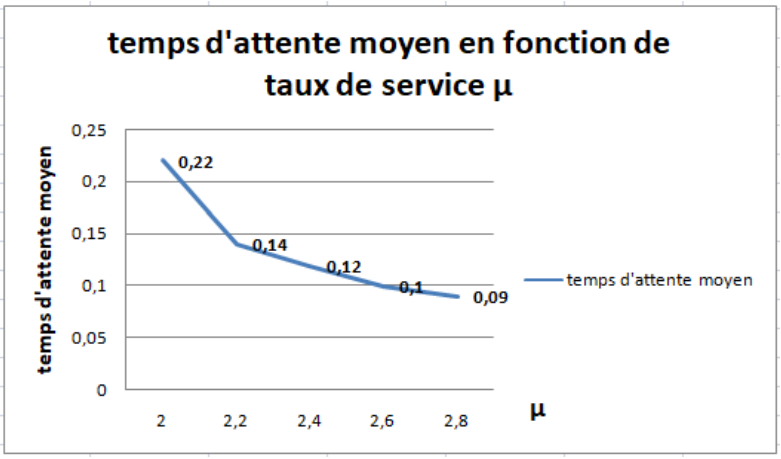

Figure 5.15 : Le temps d'attente moyen dans une file d'attente finie G/G/1 en fonction de taux de service.

2. **File d'attente infinie :**

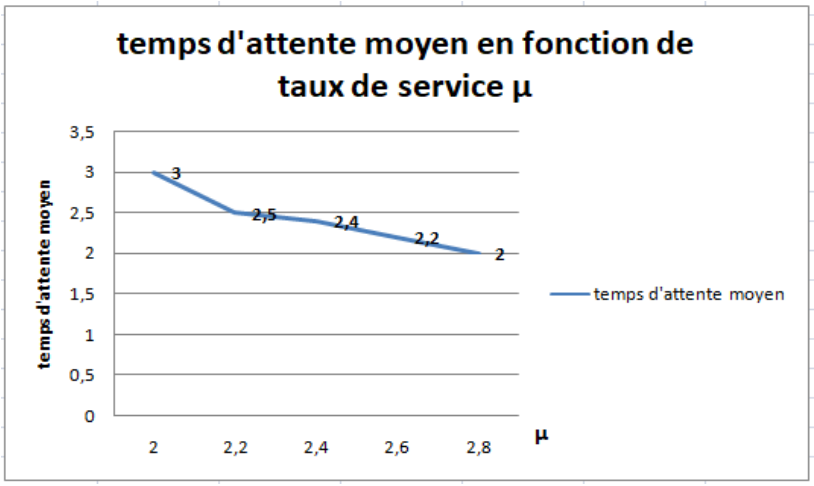

Figure 5.16 : Le temps d'attente moyen dans une file d'attente infinie G/G/1 en fonction de taux de service.

- **Analyse :**

Nous remarquons que l'augmentation du taux de service diminue le temps d'attente des clients dans la file d'attente (finie et infinie), car plus le taux de service augmente plus les clients de la file d'attente sont traités d'où la diminution du délai d'attente.

Le Débit :

1. **File d'attente finie :**

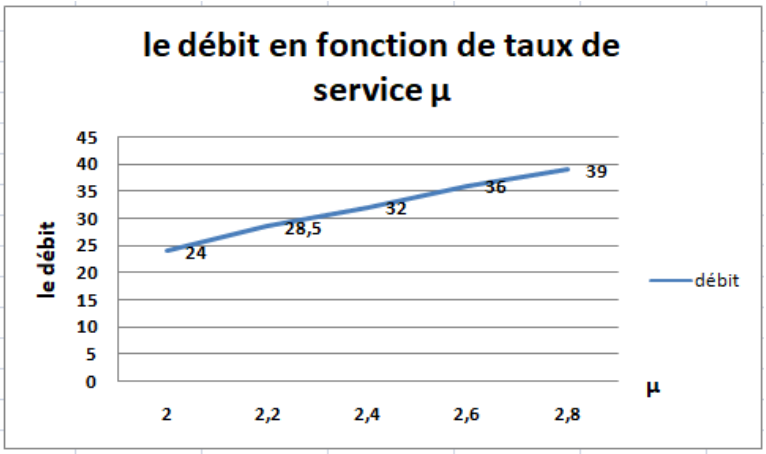

Figure 5.17 : Le débit d'une file d'attente finie G/G/1 en fonction de taux de service.

2. **File d'attente infinie :**

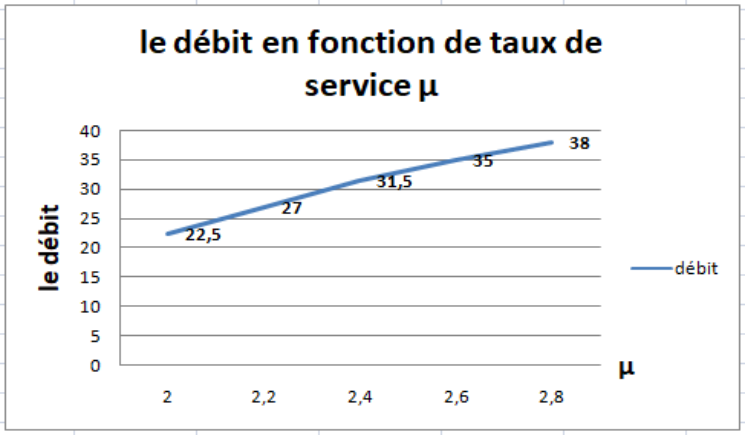

Figure 5.18 : Le débit d'une file d'attente infinie G/G/1 en fonction de taux de service.

- **Analyse :**

L'augmentation du taux de service influe sur le débit, il augmente dans les deux type de file d'attente (finie et infinie), car plus le taux d'arrivé augmente plus les clients de la file d'attente sont traités ce qui implique l'augmentation de débit.

✓ **Le 2eme modèle M/M/1 :**

Temps d'attente moyen :

1. **File d'attente finie et infinie :**

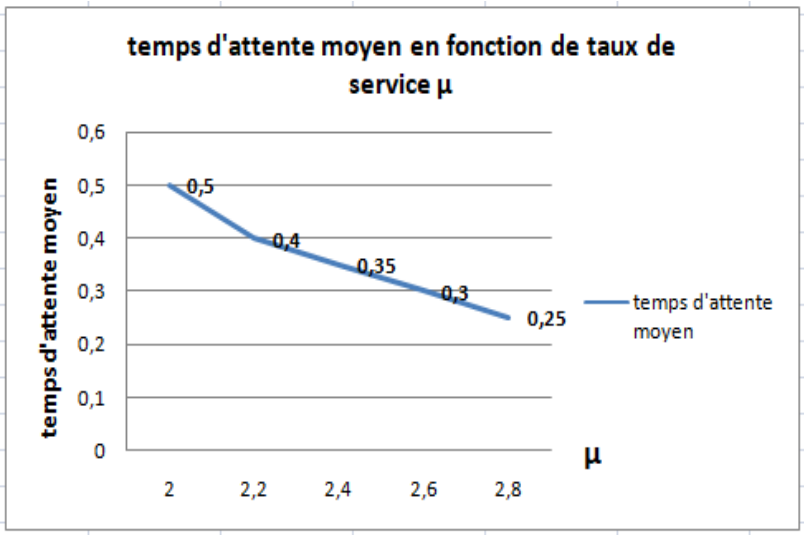

Figure 5.19 : Le temps d'attente moyen dans une file d'attente finie /infinie M/M/1 en fonction de taux de service.

- **Analyse :**

L'augmentation du taux de service implique la diminution de temps d'attente moyen pour les deux types de file d'attente (finie, infinie)

Le temps d'attente moyen est similaire pour les deux types de file d'attente.

Le Débit :

File d'attente finie et infinie :

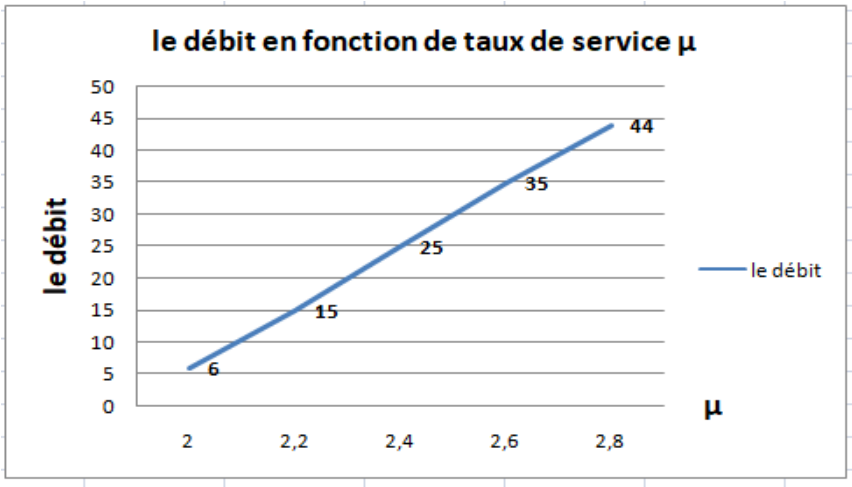

Figure 5.20 : Le débit d'une file d'attente finie /infinie M/M/1 en fonction de taux de service.

- **Analyse :**

L'augmentation du taux de service augmente le débit pour les deux types de file d'attente (finie, infinie).

Le débit est similaire pour les deux types de file d'attente.

✓ **Le 3eme modèle D/G/1 :**

Temps d'attente moyen :

1. **File d'attente finie :**

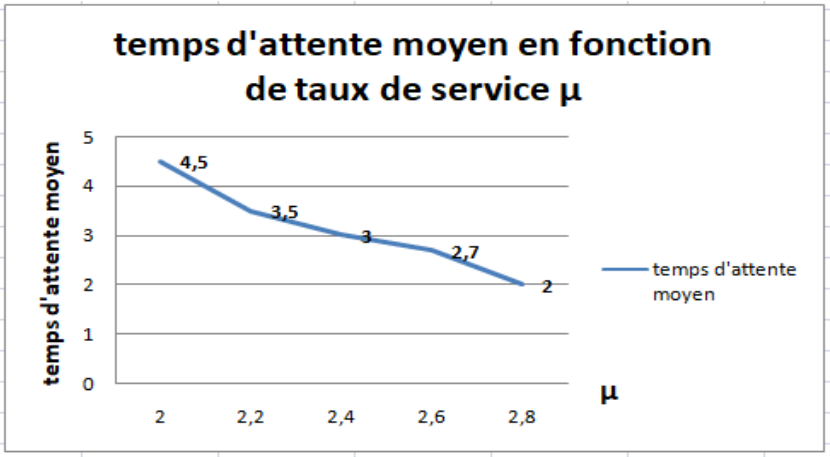

Figure 5.21 : Le temps d'attente moyen dans une file d'attente finie D/G/1 en fonction de taux de service.

2. **File d'attente infinie :**

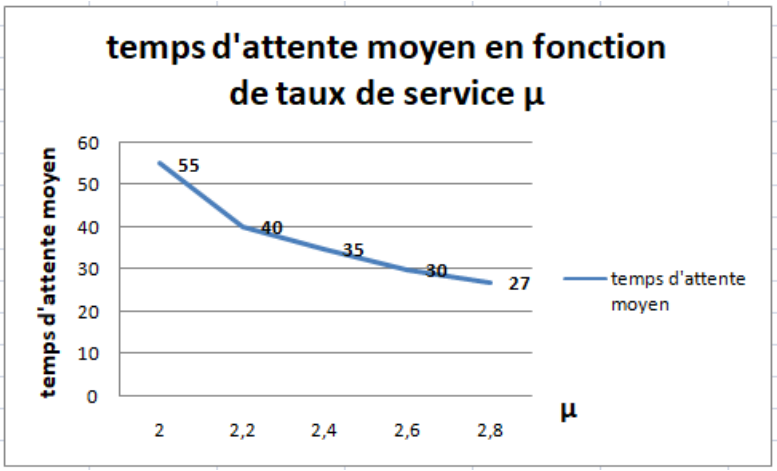

Figure 5.22 : Le temps d'attente moyen dans une file d'attente infinie D/G/1 en fonction de taux de service.

- **Analyse :**

L'augmentation du taux de service diminue le temps d'attente des clients dans la file d'attente (finie et infinie) ,car plus le taux de service augmente plus les clients de la file d'attente sont traités d'où la diminution du délai d'attente , les valeurs de temps d'attente dans une file d'attente infinie sont plus élevés que dans une file d'attente finie car cette dernière a une capacité limité contrairement dans une file infini qui reçoit un nombre infini de clients ce qu'il augmente le délai d'attente des clients dans la file.

Le Débit :

1. **File d'attente finie et infinie :**

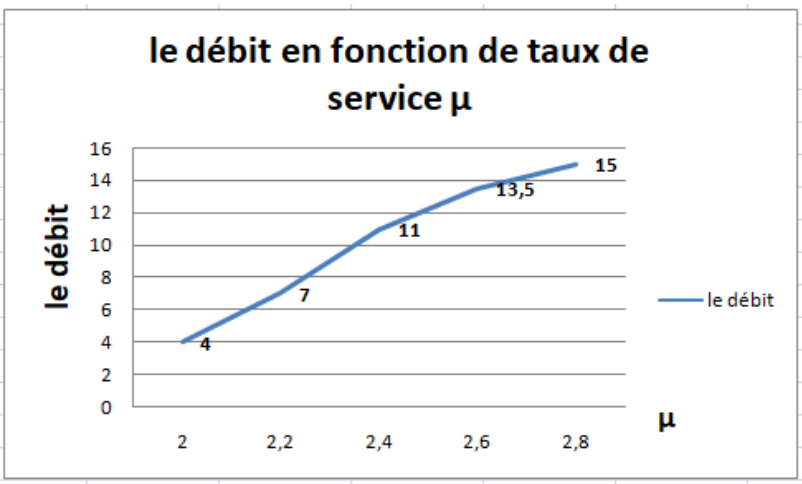

Figure 5.23 : Le débit d'une file d'attente finie/infinie D/G/1 en fonction de taux de service.

- **Analyse :**

L'augmentation de taux de service implique une augmentation de débit pour les deux types de files d'attente finie et infinie.

5.5 Résultat final

On peut dire que notre Simulateur est valable pour l'évaluation des mesures de performances de tous les systèmes de file d'attente avec vacance en fonction des paramètres de système entrés par l'utilisateur

5.6 Conclusion

Dans ce chapitre nous avons implémenté les algorithmes proposés dans le chapitre précédant en fonction des paramètres du système pour le but d'évaluer les mesures de performance de différentes systèmes de file d'attente avec vacance .

Tout d'abord nous avons présenté notre application avec les différentes fonctionnalités des fenêtres .Puis on entamé une étude expérimentale qui nous a permis de tester les 3 modèle G/G/1, M/M/1et D/G/1 avec vacance (T vacance, N vacance, aléatoire) afin de voir l'influence de la variation de certains paramètres de système sur quelques mesures de performances : le temps d'attente moyen et le débit.

Conclusion générale

Le développement d'un système complexe demande non seulement une modélisation qualitative pour vérifier sa correction logique mais aussi une validation des performances du système lors de la phase de conception.

Pour évaluer les performances d'un système, on utilise soit les méthodes analytiques telles que les files d'attente soit la simulation. Pour les solutions analytiques, le temps de résolution est rapide et les résultats sont exacts car ils sont déterminés à partir d'équations mathématiques. Toutefois, les modèles doivent être simples afin de pouvoir trouver leur forme analytique ce qui n'est pas toujours le cas, car en théorie, on pose beaucoup de contraintes ce qui rend la modélisation assez complexe, ce qui nous emmène à faire des simulations, cette méthodes donne des résultats proches de la réalité dans la plupart de temps avec moins de contraintes, mais la validation des résultats peut prendre assez de temps et d'efforts.

Dans une première partie nous avons introduire tous les notions de base utilisées dans l'étude de files d'attente avec vacance et nous avons étudie les modèles des files d'attente avec vacance.

Dans une seconde partie, nous avons inclus une approche de simulation à événement discret qui se présente comme étant une méthode de résolution des problèmes de performances, Grace à ses éléments cette approche nous a permet de bien gérer les événements du système ainsi que l'étudier à des points de discrets.

En dernière partie, nous avons développé une application sous le logiciel JAVA pour la simulation des modèles des files d'attentes avec vacance. Grâce à l'outil de simulation que nous avons développé nous avons eu la possibilité de tester les déférentes modèles de file d'attente avec vacances que nous avons proposé sur la performance des systèmes, de plus nous avons constaté que dans certains cas l'une de ces méthodes présenté meilleur résultats que l'autres et presque les mêmes résultats dans d'autres cas.

Enfin à titre de ce travail, l'approche par simulation est adoptée pour valider les résultats obtenus de l'approche analytique et rendre le système plus efficace.

Annexe

1. Quelques lois de probabilité

➢ Lois discrètes

✓ Loi uniforme discrète

Paramètres (i, j)

$$P[X = x] = 1/(j - i + 1) \text{ Pour } x=i, i+1,..., j.$$

✓ Loi binomiale

Paramètres (n, p)

$$P[X = x] = \binom{n}{x} p^x (1-p)^{n-x} \text{ Pour } x = 0, \ldots, n,$$

Où

$$\binom{n}{x} = \frac{n!}{x!(n-x)!}.$$

On a $E[X] = np$ et $VAR[X = np(1-p)]$. Dans le cas particulier $n = 1,$ nous obtenons la loi de Bernoulli.

✓ Loi multinomiale

La loi multinomiale est la généralisation directe de la loi binomiale, en considérant m résultats possibles au lieu de 2. Nous considérons alors les variables aléatoires N_i , $i = 1, \ldots, m,$ le nombre d'occurrences du i^e événement possible. Chaque événement est associe à la probabilitéP_i , $i = \{1,...,m\}$,avec les contraintes $\sum_{i=1}^{m} N_i = n$ et $\sum_{i=1}^{m} P_i = 1$.

Lorsque la variable aléatoire N_i tend vers l'infini, en vertu du théorème de la limite, nous obtenons

$$\frac{N_i - np_i}{\sqrt{np_i(1-p_{i)}}} \Rightarrow N(0,1),$$

De sorte que

$$\sum_{i=1}^{m} \frac{(N_i - np_i)^2}{np_i} \Rightarrow X^2(m-1).$$

Cette dernière remarque est à la base du test duX^2. [6]

✓ Loi géométrique

La distribution géométrique prend un paramètre p, et a comme masse de probabilité

$$P[X = x] = p(1-p)^x \quad pour \quad x = 0,1, \ldots.$$

Elle jouit de la propriété sans mémoire, à savoir $P[X = y + x/\, X \geq y] = P[X = x]$.

La fonction de répartition est :

$$F(x) = \sum_{y=0}^{\lfloor x \rfloor} P[X = x] = 1 - (1-p)^{1=\lfloor x \rfloor} \text{ pour } x \geq 0.$$

et la moyenne et la variance sont données par $E[X] = (1-p)/p$, $Var[X] = (1-p)/p^2$.

La loi géométrique peut s'interpréter comme le nombre d'expériences répétées jusqu'à on obtenir un premier succès. A chaque essai, la probabilité d'un tel succès est p, autrement dit la loi géométrique permet de caractériser un processus Bernoulli qui est arrêté dès un succès rencontré. [1]

- ✓ **Loi de poisson**

Paramètre (λ)

$$P[X = x] = \frac{\lambda^x e^{-\lambda}}{x!} \quad pour \quad x = 0,1,2, \ldots.$$

$E[X] = Var[X] = \lambda$.

Si $X_1, \ldots., X_q$ $indep.$, $X_i \sim Poisson(\lambda_i)$, alors $X = X_1 + \cdots + X_q \sim$

$Poisson(\lambda)$ ou $\lambda = \lambda_1 + \cdots + \lambda_q$

Si $\sim Binomiale\ (n,p) et\ (n,p) \to (\infty, \lambda)$, alors $X \Rightarrow Poisson(\lambda)$.

Adapté pour représenter les événements rares.

Théorème (Bornes explicites) soient $X_1, \ldots., X_n indep, X_j \sim Bernoulli(p_j)$

$X = X_1 + \cdots + X_n$ et $\lambda = p_1 + \cdots + p_n$.

Alors, pour tout x : [1]

$$\left| P[X = x] - \frac{\lambda^x e^{-\lambda}}{x!} \right| \leq \sum_{j=1}^{n} p_j^2 \left[= \lambda/n \ si\ p_j = p\ pour\ tout\ j \right].$$

➢ Lois continues univariées

- ✓ **Loi uniforme**

Uniforme sur (a, b)

$$f(x) = 1/(b-a)\ pour \quad a < x < b\ .$$

$E[X] = (b-a)/2$ et $Var\ (b-a)^2/12$.

✓ Loi triangulaire

La distribution triangulaire est uni modale, avec une fonction de densité de probabilité qui est linéaire par morceaux. La fonction de densité est croissante sur un intervalle [a ; c], décroissante sur un intervalle [c ; b], et nulle ailleurs. Elle est souvent utilisée en l'absence de données.

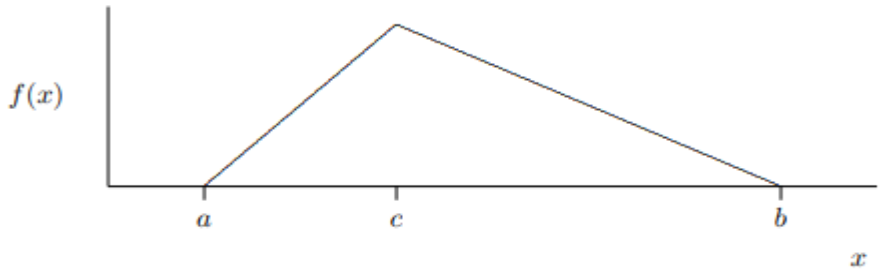

Figure A.1 : Fonction de densité de la distribution triangulaire.

Domaine	$a \leq x \leq b.$
Paramètres	a : limite inférieure.
	$b > a$: limite supérieure.
	$c, a \leq c \leq b$: mode.
Densité	$f(x) = \begin{cases} \frac{2}{b-a}\frac{x-a}{c-a} & \text{if } a \leq x \leq c, \\ \frac{2}{b-a}\frac{b-x}{b-c} & \text{if } c \leq x \leq b, \\ 0 & \text{otherwise.} \end{cases}$
Fonction de répartition	$F(x) = \begin{cases} \frac{(x-a)^2}{(b-a)(c-a)} & \text{if } a \leq x \leq c, \\ 1 - \frac{(b-x)^2}{(b-a)(b-c)} & \text{if } c \leq x \leq b, \\ 0 & \text{otherwise.} \end{cases}$
Moyenne	$\frac{a+b+c}{3}$
Variance	$\frac{a^2+b^2+c^2-ab-ac-bc}{18}$
Mode	c
Médiane	$\begin{cases} a + \sqrt{\frac{(b-a)(c-a)}{2}} & \text{if } c \geq \frac{b-a}{2} \\ b - \sqrt{\frac{(b-a)(b-c)}{2}} & \text{if } c \leq \frac{b-a}{2} \end{cases}$

Table A.1 : Distribution triangulaire.

✓ Loi normale

Paramètres : (μ, σ^2) ; $X \sim N(\mu, \sigma^2)$, de densité

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2} \; pour \; x \in \boldsymbol{\mathcal{R}}\,.$$

Une propriété intéressante est la linéarité de la distribution normale si $X \sim N(\mu, \sigma^2)$, alors $Y = \alpha X + b \sim N(\alpha\mu + b, \alpha^2\sigma^2)$.

$$\Phi(x) = P[Z \leq x] = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\delta^2/2}\, ds.$$

Nous pouvons écrire en particulier :

$$X = \mu + \sigma N(0, 1).$$

Une des principales motivations derrière la loi normale se trouve dans le théorème de la limite centrale (TLC), qui énonce que X suit approximativement une distribution normale si X est une somme de nombreux petits effets indépendants.

- ✓ **Loi lognormale**

Paramètres : (μ, σ^2)

Au lieu d'une somme, on a un produit de petits effets : $Y_n = (\mathrm{Z}_1, \mathrm{Z}_2 \ldots \mathrm{Z}_n)^{1/n}$.

On a $\ln Y_n = (\ln Z_1 + \cdots + \ln Z_n)\ /n$.

Sous les conditions du TLC :

$$\frac{ln\ Y_n - E[ln\ Y_n]}{(Var\ [ln\ Y_n])^{1/2}} \Rightarrow N(0,1)\ pour\ n \to \infty.$$

On dit que $X \sim$ lognormale si $\ln X \sim$ normale. Cette loi apparait souvent en économie et finance.

Si $ln\ X \sim N(\mu, \sigma^2), X \sim lognormale(\mu, \sigma^2)$ a la densité

$$f(x) = \frac{1}{\sqrt{2\pi\sigma x}} e^{-(\ln x - \mu)^2/(2\sigma^2)} pour\ x > 0\ .$$

On a $E[X] = e^{\mu+\sigma^2/2}$ et $Var[X] = e^{2\mu+\sigma^2}\left(e^{\sigma^2} - 1\right)$.

- ✓ **Loi de Erlang**

Une variable aléatoire de Erlang est une variable gamma où le paramètre de forme α est un entier. Elle correspond aussi à la somme de α variables exponentielles indépendamment et identiquement distribuées, de paramètre λ.

Domaine	$0 \leq x < \infty$.
Paramètres	$\alpha \in \mathbb{N}_0$: paramètre de forme ;
	$\lambda > 0$: paramètre d'échelle.
Distribution	$1 - e^{-\lambda x}\left(\sum_{i=0}^{\alpha-1}\right)\frac{(\lambda x)^i}{i!}$
Densité	$\frac{\lambda(\lambda x)^{\alpha-1}e^{-\lambda x}}{(\alpha-1)!}$
Moyenne	$\frac{\alpha}{\lambda}$
Variance	$\frac{\alpha}{\lambda^2}$

Table A.2 : Distribution de Erlang.

✓ **Loi exponentielle**

Il s'agit de la seule loi continue qui jouit de la propriété sans mémoire :

$$P[X > t + x | X > t] = \frac{P[X > t + x]}{P[X > t]} = \frac{e^{-\lambda(t+x)}}{e^{-\lambda t}} = e^{-\lambda x} = P[X > x].$$

Cette propriété simplifie beaucoup l'analyse mathématique et l'explique la grande popularité de la loi exponentielle.

Domaine	$0 \leq x < \infty$.
Paramètres	$\alpha \in \mathbb{N}_0$: paramètre de forme ;
	$\lambda > 0$: paramètre d'échelle.
Distribution	$1 - e^{-\lambda x} \left(\sum_{i=0}^{\alpha-1} \right) \frac{(\lambda x)^i}{i!}$
Densité	$\frac{\lambda(\lambda x)^{\alpha-1} e^{-\lambda x}}{(\alpha-1)!}$
Moyenne	$\frac{\alpha}{\lambda}$
Variance	$\frac{\alpha}{\lambda^2}$

Table A.3 : Distribution exponentielle.

✓ **Loi de Cauchy**

La distribution de Cauchy est unimodale et symétrique, avec des queues nettement plus épaisses que la normale. La fonction de densité de probabilité est symétrique autour de a, avec pour quartiles supérieur et inferieur $a \pm b$. Le rapport de deux variables aléatoires normales, indépendantes, et de moyennes nulles, suit une distribution de Cauchy.

Domaine	$-\infty < x < \infty$.
Paramètres	a : paramètre de position, la médiane.
	b : paramètre d'échelle.
Distribution	$\frac{1}{2} + \frac{1}{\pi} \arctan\left(\frac{x-a}{b}\right)$
Densité	$\frac{1}{\pi b \left(1 + \left(\frac{x-a}{b}\right)^2\right)}$
Moments	n'existent pas.
Médiane	a
Mode	a

Table A.4 : Distribution de Cauchy.

La loi de Cauchy est de plus une des lois pour lesquelles la loi des grands nombres ne s'applique pas. Si $X_1, \ldots, X_n$ sont indépendamment et identiquement distribuées. et suivent une distribution de Cauchy, la moyenne empirique

$$\overline{X_n} = \frac{1}{n} \sum_{i=1}^{n} x_i.$$

Ne converge pas vers une quantité déterministe quand n tend vers l'infini; elle est elle-même distribuée selon une loi de Cauchy.

✓ **Loi de Weibull**

Paramètres : (α, λ)

$$f_{(x)} = \alpha\lambda^{\alpha}(x-\delta)^{\alpha-1}exp[-(\lambda(x-\delta))^{\alpha}] pour\ x > \delta,$$

$$F_{(x)} = 1 - exp[-(\lambda(x-\delta))^{\alpha}].$$

Paramètres : $\delta \in R$ (localisation), $\alpha > 0$ (forme), $\lambda > 0$ (échelle).

Souvent utilise (avec $\delta = 0$) pour modéliser les durées de vie.

Pour $\alpha = 1$: loi exponentielle.

Pour $\alpha > 1$: taux de panne croissant, $CV > 1$.

Pour $\alpha < 1$: taux de panne décroissant, $CV < 1$.

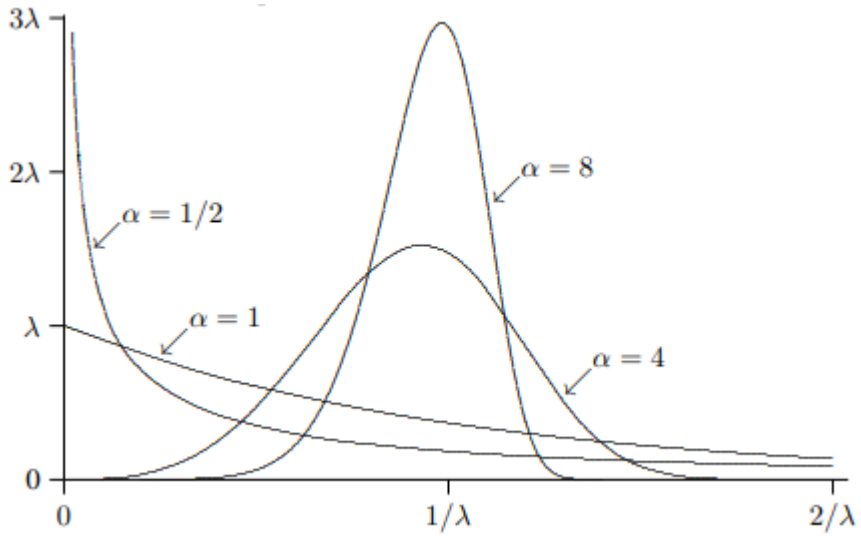

Figure A.2 : Densité de la distribution Weibull.

Motivation : le minimum le plusieurs variable aléatoire indépendamment et identiquement distribuées. Suit approx. Une loi de Weibull.

Interprétation : la panne d'un système survient lors de l'occurrence du premier événement parmi plusieurs événements indépendants.

- ✓ **Loi de Gumbel**

La loi de Gumbel est aussi appelée loi de la valeur extrême . $Y \sim$ Weibull (α, λ) Ssii $X = ln\ Y \sim$ Gumbel (α, λ).

La loi de Gumbel est populaire en théorie des choix discrets ; la différence de deux Gumbel indépendantes suit une distribution logistique.

- ✓ **Loi beta**

La distribution beta de paramètres de forme $\alpha > 0$ et $\beta > 0$, sur l'intervalle $(0, 1)$, a la densité

$$f_{(x)} = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1}(1-x)^{\beta-1} \quad pour \;\; 0 < x < 1.$$

Elle est parfois utilisée pour modéliser une proportion aléatoire. Sa moyenne et sa variance sont $E[X] = \alpha/(\alpha+\beta)$ et $Var[X] = \alpha\beta/[(\alpha+\beta)^2(\alpha+\beta+1)]$.Certaines densités de beta sont illustrées sur la Figure 3 pour le cas symétrique $(\alpha = \beta)$ et la Figure 4 pour le cas non-symétrique.Pour le cas symétrique, toute la densité se concentre à $1/2$ (asymptotiquement) lorsque $\alpha \to \infty$ et se divise en deux entre 0 et 1 lorsque $\alpha \to 0$.

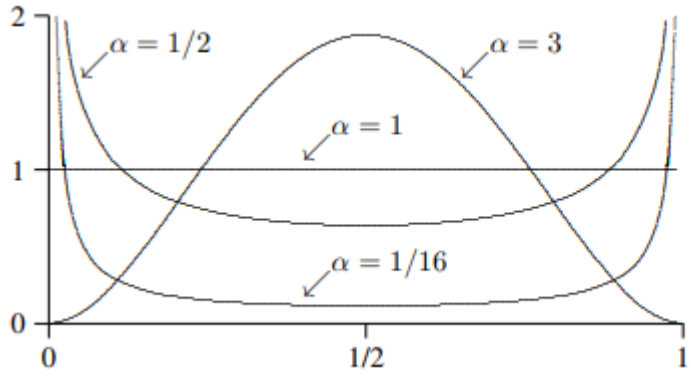

Figure A.3 : La densité bêta symétrique avec les paramètres $\alpha = \beta =$ $1/16$, $1/2$, 1 et 3.

On a $X \sim Beta(\alpha, \beta)$ si et seulement si $1 - X \sim Beta(\beta, \alpha)$, ou encore si et seulement si $X/(1-X)$ $Pearson\text{-}6(\alpha, \beta, 1)$. Les distributions $Beta\ (1, 1)$ et $U\ (0, 1)$ sont les mêmes. Le $Beta\ (1, 2)$ et le $Beta\ (2, 1)$ sont des cas particuliers de la distribution triangulaire. Une variable bêta aléatoire X sur $(0, 1)$ peut être transformée en une variable bêta aléatoire Y avec la même forme de distribution, mais sur un intervalle arbitraire(a, b), via la transformation linéaire $Y = a + (b-a)X$.

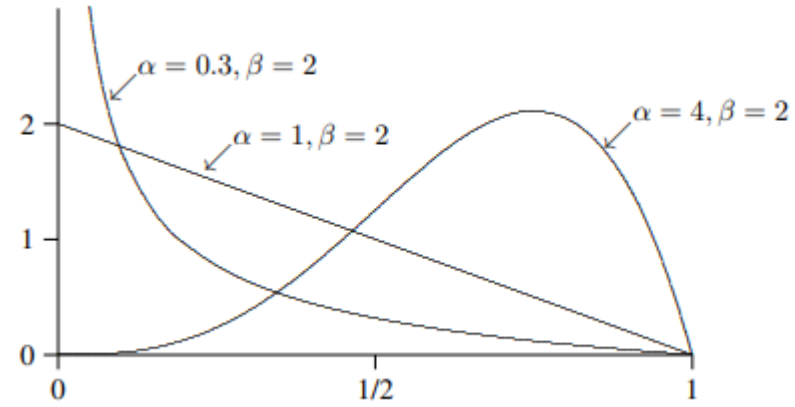

Figure A.4 : La densité bêta avec les paramètres $\beta = 2$ et $\alpha =$ $0{,}3$, 1 et 4.

Nous avons la relation suivante entre les fonctions de distribution binomiale et bêta : Si Y Binomiale (n, p) ,alors $P[Y \geq y] = P[X \leq p]$ où $X \backsim Beta(y, n - y + 1)$. Ceci est parfois utilisé pour estimer la fonction de distribution binomiale.

- ✓ **Loi du chi-deux**

Paramètre : (n)

$$f_{(x)} = \frac{x^{(n/2)-1} e^{-x/2}}{2^{n/2}\, \Gamma(n/2)} \; pour \; x > 0.$$

Cas particulier de la loi gamma $E[X] = n \; et \; Var[X] = 2n.$

Si $X_1, \ldots, X_n$ independemment et identiquement distribuées. $N\ (0,\ 1)$, alors $X = X_1^2 + \cdots + X_n^2 \sim$ chi-deux(n).

Utilisé dans les tests de chi-deux et pour calculer un intervalle de confiance pour la variance. [6]

➢ Lois tronquées

Pour tronquer $f(x)$ sur un intervalle (a, b) :

$$\check{f}(x) = \begin{cases} f(x)/K & pour \; a; x; b, \\ 0 & ailleurs\,, \end{cases}$$

Où

$$K = \int_a^b f(u)\, du.$$

Pour générer des variable aléatoire selon [1]

$$\check{f}: U \sim Uniforme\big(f(a), f(b)\big) et \; X = F^{-1}(U).$$

➢ Lois décalées

$X = X_0 + a$ où X_0 a une loi standard.

Exemples : exponentielle, gamma, lognormale, ...

Estimer le paramètre a à partir des données est parfois difficile. [6]

➢ Mélanges de lois

$$f_{(x)} = \sum_{j=1}^{\infty} w_j f_j(x),$$

où chaque f_j est une densité et les poids w_j somment à 1.

Population divisée en sous-ensembles, chaque sous-ensemble a sa propre distribution.

Mélange non dénombrable :

$$f(x) = \int_{\theta} f_{\theta}(x) w(\theta)\, d\theta,$$

où $\theta \subseteq \mathcal{R}^d$, et w une densité sur θ.

Pour générer *X,* générer d'abord Y selon les poids, puis X sous la densité f_y. [6]

➢ Lois multivariées

Un vecteur $X = (X_1, \dots, X_d)^T$ a une loi multivariée de fonction de répartition F

si pour tout $x = (x_1, \dots, x_d)^T \in \mathcal{R}^d$, on a

$$F(x) = P[X \leq x] = P[X_1 \leq x_1, \dots, X_d \leq x_d].$$

On définira également les lois marginales en considérant la fonction de répartition associée à chaque composante du vecteur aléatoire :

$$F_j(x) = P\left[X_j \leq x\right]$$

$$= P\left[X_1 \leq +\infty, \dots, X_{j-1} \leq +\infty, X_j \leq x_j, X_{j+1} \leq x_{j+1}, \dots, X_d \leq +\infty\right],$$

Pour *j=1,...d.*

Deux variables *X* et *Y* sont jointement continues s'il existe une fonction non négative *f(x, y),* appelée fonction de densité jointes de *X* et *Y,* telle que pour tous ensembles réels *A* et *B,*

$$[X \in A \cap Y \in B] = \int_B \int_A f(x, y) dx dy\,.$$

Le concept s'étend directement à plus de deux variables. [6]

➢ Fonctions de dépendance (copules)

Soit$X = (X_1, \dots, X_d)$ variable aléatoire continue, où $P[X_j \leq x] = F_j(x)$.Alors $U = (U_1, \dots, U_d) = (F_1(X_1), \dots, F_d(X_d))$ est un vecteur aléatoire dont les lois marginales sont *U (0, 1).* Ce vecteur U a une certaine fonction de répartition, disons$C$: $P[U_1 \leq u_1, \dots, U_d \leq u_d] = C(u_1, \dots, u_d)$.

En principe, on peut générer U selon C et poser$X_j = F_j^{-1}(U_j)$. La fonction C spécifie la dépendance entre lesU_j, indépendamment des marginales F_j . Une telle fonction de répartition

C, dont les lois marginales sont toutes *U (0, 1)*, s'appelle une copule ou fonction de dépendance pour *U*. [6]

➢ Lois empiriques et quasi-empiriques

Supposons que nous disposons des observations $x_1, \dots, x_n$, que nous renotons $x_{(1)}, \dots, x_{(n)}$ Après les avoir triées par ordre croissant.La fonction de répartition empirique est définie en partant de *0* et en augmentant sa valeur de *1/n* chaque observation rencontrée :

$$\widehat{F_n}(x) = \frac{1}{n}\sum_{i=1}^{n} \mathrm{I}[x_i \leq x].$$

Schématiquement, nous pouvons la représenter comme sur la Figure 6.

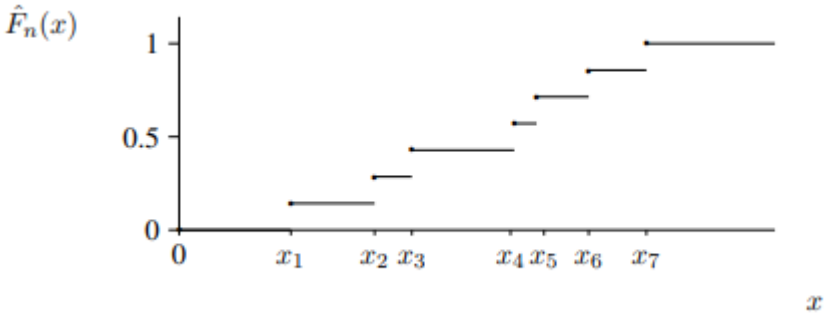

Figure A.5 : Fonction de répartition empirique.

Générer des valeurs selon $\widehat{F_n}$ correspond a tirer des valeurs au hasard dans l'échantillon, avec remplacement. [6]

Bibliographie

[1] H.,kobayashi,modelling and analyses ;an introduction to system performance evaluation methodology , the systems programming series , Addison Wesley,1987.

[2] Naishuo, T., & Zhang, Z. G. Vacation QueueingModels:Theory and Applications ,2006.

[3] Doshi, B. T. Queueing systems with vacations—a survey. Queueing systems, 1(1), 29-66, 1986.

[4] Reviewer-Frankel, D. brief review: Queueing Analysis: A Foundation of Performance Evaluation. Volume 1: Vacation and Priority Systems, Part 1 by H. Takagi (North-Holland, 1991). ACM SIGMETRICS Performance Evaluation Review, 19(2), 13, 1991.

[5]Ke, J. C., Wu, C. H., & Zhang, Z. G. Recent developments in vacation queueing models: a short survey. International Journal of Operations Research, 7(4), 3-8 , 2010.

[6] Averill M. Law. Simulation Modeling & Analysis. McGraw-Hill, Boston, MA, USA, 4th edition, 2007.

[7] Rehab F. Khalaf.''On Some Queueing Systems with Server Vacations, Extended Vacations, Breakdowns, Delayed Repairs and Stand-bys''A thesis submitted for the degree of Doctor of Philosophy, Brunel University,2012.

[8] QUITTARD P., -Processus stochastiques et file d"attente, OPU, Algérie, 1983.

[9] Jain.J , Mohanty.S , & Böhm.W .A-Course-on-Queueing-Models.

[10] Stewart, W. J. Probability, Markov chains, queues, and simulation: the mathematical basis of performance modeling. Princeton university press ,2009.

[11] : Lakatos, L., Szeidl, L., & Telek, M. Introduction to queueing systems with telecommunication applications. Springer ,2019.

[12] A. Bartoli. Le management dans les organisations publiques, Ed. Dunod, 1997.

[13]Newell, G. F. 1982. Applications of Queueing Theory. Second edition, Chapman and Hall, London (First edition : 1971).

[14] FOTSO, Donatien CHEDOM et FOTSO, Laure Pauline, " Étude et simulation du phénomène d'attente dans un système bancaire".

[15] BAYNAT B., -Théorie des files d'attente : Des chaines de Markov aux réseaux a forme produit, coll. Réseaux et télécommunications, 2000.

[16]KUMAR, Binay, VIJ, Ashu, et KUMAR, Pankaj, "Some Basic Concepts in Queuing Theory," International Advanced Research Journal in Science, Engineering and Technology, vol. 2, pp. 56- 60, 2015.

[17] Handbook of Simulation, ed. Jerry Banks, pp. 3–30. Copyright © 1998. Adapted by permission of John Wiley & Sons, Inc.

[18] CHOI, Byoung Kyu et KANG, Donghun, "Modeling and simulation of discrete event systems," John Wiley & Sons, 2013.

[19] C. R. A. Catlow, R. A. van Santen, and B. Smit, Computer Modelling of Microporous Materials, 2004.

[20]Discrete Event Simulation System Version 3.2 User Manual by András Varga Last updated: March 29, 2005.

[21] Banks, Carson, Nelson & Nicol Discrete-Event System Simulation .

[22]K., Sanjay Bose An Introduction to Queueing Systems « Chapter 7. Simulation Techniques for Queues and Queueing Networks »Pages 257-281.

[23]Donatien Chedom ,Fotso & Laure Pauline, Fotso « Étude et simulation du phénomène d'attente dans un système bancaire » ,Université de Yaoundé I ,Cameroun.

[24]LAW,, KELTON, « Simulation Modeling and Analysis ».

[25]Prathamesh Mayekar, J. Venkateswaran, Manu K. Gupta and N. Hemachandra « Simulation and analysis of some dynamic priority schemes in two-class queues » Industrial Engineering and Operations Research, IIT Bombay September 22, 2014.

[26]SADEGHI, N., FAYEK, A. Robinson, et SERESHT, N. Gerami, "Queue performance measures in construction simulation models containing subjective uncertainty," Automation in Construction, vol. 60, pp. 1-11, 2015.

[27]Silvia Ďutkováa , Karol Achimskýa &Dominika Hoštákováa «Simulation of Queuing System of Post Office » 13th International Scientific Conference on Sustainable, Modern and Safe Transport (TRANSCOM 2019), High Tatras, Novy Smokovec – Grand Hotel Bellevue, Slovak Republic, May 29-31, 2019.

[28]P. J. Erard et P. Deguenon, "Simulation par événements discrets". Presses Polytechniques et universitaires romandes ,1996.