**RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE**

**MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR**

**ET DE LA RECHERCHE SCIENTIFIQUE**


Université SAAD DAHLEB BLIDA

Faculté des Sciences

Département d'Informatique


## Mémoire de fin d'études

En vue d'obtenir le diplôme de Master en Informatique

**Spécialité :** Systèmes Informatique et Réseaux

**Thème**

# Évaluation des performances d'un mécanisme d'économie d'énergie dans un réseau de capteurs sans fil


**Devant le jury :**

Mme Boutoumi Bachira Promotrice.

M. Cherif Zahar Amine President.

Mlle. Djeddar Affrah Examinatrice.

**Présenté par :**

M. Messous Ali.

M. Mameche Mohamed.


**Année universitaire 2020/2021**

# Remerciement

Nous remercions Allah qui nous a aidé et donné le courage et la force de terminer ce travail malgré tous les obstacles et les conditions sanitaires à cause de COVID19.

Avant de commencer cette présentation, nous tenons à exprimer notre sincère gratitude à tous ceux qui nous ont aidés à mener à bien ce projet.

Nous remercions Madame Boutoumi Bachira, notre promotrice, pour son aide, sa générosité et son temps avec nous.

Nous remercions également tous nos professeurs pour leur disponibilité, leur soutien et leur contribution inestimable et tout particulièrement M. Ould khoua notre responsable spécialisé.

Un grand merci à toutes nos familles et à nos proches et à tous nos collègues.

Un grand merci à tous ceux qui ont contribué de près ou de loin à la réalisation de ce travail.

# Résumé

De nos jours, les réseaux de capteurs sans fils RCSFs ont trouvé leur chemin vers le centre de la zone d'intérêt des chercheurs dans le monde entier, grâce à leurs caractéristiques imposantes, telles que leur déploiement et évolutivité, leur petite taille, faible coût, et le nombre des sources d'informations et calculs limités. Ces derniers influencent d'une manière directe la durée de vie d'un RCSF, d'où se voit que la consommation d'énergie est un facteur critique dans la conception d'un RCSF.

Ce facteur représente, depuis quelques années, le défi majeur dans le domaine des RCSFs. En plus de la conservation d'énergie, l'évaluation des mesures de performances d'un nœud de capteur seront l'objectif principal de notre travail. On procèdera donc notre travail avec les files d'attentes avec plusieurs ordres de priorité en incorporant la politique de vacance N, et on utilisera l'approche de simulation à événement discret. De plus, on réalisera un outil de simulation pour calculer et comparer les différentes mesures de performances entre les buffers avec et sans priorité, et les systèmes avec et sans préemption.

**Mots Clés**
Réseaux de capteurs sans fil, File d'attente avec vacances, File d'attente avec priorité, Économie d'énergie, Simulation à temps discret, Mesures de performances, Évaluation des performances.

# Abstract

Nowadays, WSN wireless sensor networks have found their way to the center of the area of interest of researchers worldwide, thanks to their imposing features, such as deployment and scalability, small size, low cost, and the number of limited sources of information and calculations. These directly influence the lifespan of a WSN; hence power consumption is a critical factor in the design of a SCN.

This factor has been the major challenge in the field of WSN in recent years. In addition to energy conservation, evaluating the performance metrics of a sensor node will be the main focus of our work. We will therefore proceed with our work with queues with several orders of priority by incorporating the vacancy policy N, and we will use the discrete event simulation approach. In addition, a simulation tool will be produced to calculate and compare the different performance measurements between the buffers with and without priority, and the systems with and without preemption.



**Keywords**
Wireless sensor networks, Queue with vacations, Queue with priority, Energy saving, Discrete time simulation, Performance Metrics, Performance evaluation.

# ملخص

في الوقت الحاضر، وجدت شبكات الاستشعار اللاسلكية طريقها إلى مركز مجال اهتمام الباحثين في جميع أنحاء العالم ، وذلك بفضل ميزاتها ألمهيبة مثل النشر وقابلية التوسع ، والحجم الصغير ، والتكلفة المنخفضة ، وعدد مصادر المعلومات والحسابات ألمحدودة. تؤثر هذه بشكل مباشر على عمر ، وبالتالي فإن استهلاك الطاقة هو عامل حاسم في تصميم شبكات الاستشعار اللاسلكية.

كان هذا العامل هو التحدي الرئيسي في مجال شبكات الاستشعار اللاسلكية في السنوات الأخيرة. بالإضافة إلى الحفاظ على الطاقة ، سيكون تقييم مقاييس أداء عقدة الاستشعار هو المحور الرئيسي لعملنا. لذلك سنواصل عملنا مع قوائم الانتظار مع العديد من أوامر الأولوية من خلال دمج سياسة الوظائف الشاغرة N ، وسوف نستخدم نهج محاكاة الحدث المنفصل. بالإضافة إلى ذلك ، سيتم إنتاج أداة محاكاة لحساب ومقارنة قياسات الأداء المختلفة بين المخازن المؤقتة ذات الأولوية وبدون الأولوية ، والأنظمة التي بها إجراءات استباقية وبدونها.



**الكلمات الدلالية:**
شبكات الاستشعار اللاسلكية ، قائمة الانتظار مع الإجازات ، قائمة الانتظار مع الأولوية ، توفير الطاقة ، محاكاة الوقت المنفصلة، تقييم الأداء، مقاييس الأداء

# Table des matières

# Table des figures

## Liste des Tableaux



# Liste des abréviations

**RCSF :** Réseaux de capteurs sans fil.
**WSN:** Wireless sensor network.
**QoS:** Quality of service.
**FCFS:** First come first served.
**DES :** Discrète évent simulation.
**FEL :** Future Event List.
**POO :** Programmation orienté objet.

# Introduction Générale

Vers la fin des années 1990, les nœuds de capteurs sans fils ont émergé. Ces derniers sont devenus un élément incontournable grâce à leurs caractéristiques spéciales telles que leur petite taille, leur mobilité, faible coût, leur déploiement et leur faible consommation d'énergie.

Un nœud de capteur se compose généralement de quatre unités : unité d'acquisition, unité de traitement, unité de communication et une source d'énergie et équipé de deux éléments supplémentaires : un mobilisateur et un système de localisation.

Un nœud de capteur sans fil, est équipé d'une batterie relativement faible vu les emplacements des nœuds dans des environnements hostiles et dangereux.

Un réseau de capteur sans fil RCSF est un ensemble d'un grand nombre de capteur qui communique entre eux. Ces réseaux de capteurs sans fil sont de nos jours, au centre des recherches et d'implémentation dans des différents domaines (militaire, médical, environnemental …etc.).

Ces RCSFs présentent une variété des défis qui attirent l'intention des chercheurs. L'un des défis majeurs qui rencontrent les RCSFs et la consommation d'énergie d'un nœud, ce qui a invoqué plusieurs recherches et travaux comme illustré dans [2] et [3].

Il existe une variété de méthodes pour modéliser un nœud de capteur sans fils. Dans ce travail, on s'intéresse à utiliser un système de files d'attente avec vacances pour modéliser la problématique posée. Dans ce système, les serveurs passent à un état de vacance une fois la condition de vacances est acquise. Le serveur peut également réaliser des tâches supplémentaires quand il est à l'état inactif.

Il existe plusieurs travaux qui ont été élaboré par des chercheurs (illustré dans [52] [4] [6] [3] [33]).

Pour réaliser notre système, on utilise une approche de simulation à événements discrets. La simulation se présente comme un outil important pour bien étudier les caractéristiques du système et ses performances.

Dans ce mémoire, nous allons nous concentrer à évaluer les différentes mesures de performance d'un RCSF en utilisant la simulation à événements discrets avec ses différentes approches et caractéristiques.

Le présent document est organisé comme suit :

**Chapitre 01 :** on présente les RCSFs : leurs caractéristiques, défis, types et les domaines d'application des RCSFs.

**Chapitre 02 :** nous allons entamer les files d'attentes classiques et celles avec vacances, avec/sans priorité, leurs caractéristiques, les différentes politiques de vacances et les différents processus d'arrivé et de distribution.

**Chapitre 03 :** on définira les principes généraux de la simulation ainsi que celle de la simulation à événements discrets, et on finira par présenter quelques travaux connexes.

**Chapitre 04 :** concerne la conception de l'outil de simulation qu'on a développé ainsi que les différentes mesures de performances d'un RCSF.

**Chapitre 05 :** une démonstration de l'outil développé et les résultats obtenus de la simulation.

Et on termine ce mémoire par une conclusion générale.

# 1. Les réseaux de capteurs sans fil

## 1.1. Introduction

Dès leur création dans les années 1990, les réseaux de capteurs sans fils (RCSF) ont connu un grand succès au sein des communautés scientifiques et industrielles. Grâce à plusieurs avantages (moins coûteux, déployables, évolutifs...etc.) les RCSFs ont pu s'instaurer comme un acteur non contournable dans les architectures réseaux actuelles.

Néanmoins, on est toujours à la recherche de nouveaux défis pour améliorer l'emploie des RCSFs dans les différents domaines d'utilisations des réseaux.

Dans ce chapitre nous allons traiter tous les aspects nécessaires d'un RCSFs. Nous montrons la définition et l'architecture interne d'un nœud de capteur, les réseaux de capteurs sans fil, Les types des RCSF, l'architecture des RCSF, la pile protocolaire, la consommation d'énergie d'un nœud capteur, les facteurs principaux de la conception des RCSFs, et ses différents d'applications dans notre vie moderne.

## 1.2. Définition d'un nœud de capteur sans fil

Un nœud capteur sans fil (mote ou sensor node) est un nœud qui constitue la base d'un RCSF, qui est constitué pour une seule tâche principale de détecter les événements et effectuer un traitement des données, puis la transmission de ces données.[39]

La figure 1.1 ci-dessous représente trois différents types des nœuds de capteurs


*Figure 1.1: Trois différents types des nœuds de capteurs [56]*

## 1.3. Architecture interne d'un capteur

Un nœud de capteur contient quatre unités de base :

### 1.3.A. Unité de captage ou acquisition (*Sensing Unit*)

L'unité de captage a pour objectif l'acquisition des données à partir de l'environnement physique et se compose de deux unités principales : le capteur et un convertisseur de signal analogique en signal numérique (CAN). Le capteur fournit des signaux analogiques, basés sur le phénomène observé, au convertisseur Analogique/Numérique. Ce dernier transforme ces signaux en un signal numérique compréhensible par l'unité de traitement.

### 1.3.B. Unité de traitement (*Processing Unit*)

Elle contient un micro-processeur associé à une petite unité de stockage. Elle fonctionne à l'aide d'un système d'exploitation spécialement conçu pour les micro-capteurs (TinyOs par exemple). Elle exécute les protocoles de communications qui permettent de faire « collaborer » le nœud avec les autres nœuds du réseau.

### 1.3.C. Unité de communication (*Transceiver Unit*)

Aussi appelé unité de transmission, elle se compose d'un émetteur/récepteur pour la communication sans fil entre les différents nœuds du réseau. En principe cette unité possède quatre modes de fonctionnement : émission, réception, oisif et sommeil.

### 1.3.D. Unité de contrôle d'énergie (*Power Unit*)

Pour ce type de réseau sans fil l'alimentation est un composant crucial. Il y a essentiellement deux aspects :

- Stocker l'énergie et la fournir sous la forme requise, habituellement à l'aide d'une batterie.
- Tenter de reconstituer l'énergie consommée par un réapprovisionnement grâce à une source externe au nœud capteur, telles les cellules solaires.

Selon le domaine d'application, il peut aussi contenir des modules supplémentaires tels qu'un système de localisation (GPS), ou bien un système générateur d'énergie (cellule solaire). Quelques micro-capteurs, plus volumineux, sont dotés d'un système mobilisateur chargé de les déplacer en cas de nécessité.

La figure 1.2 ce dessous représente les composants de base d'un nœud de capteur.

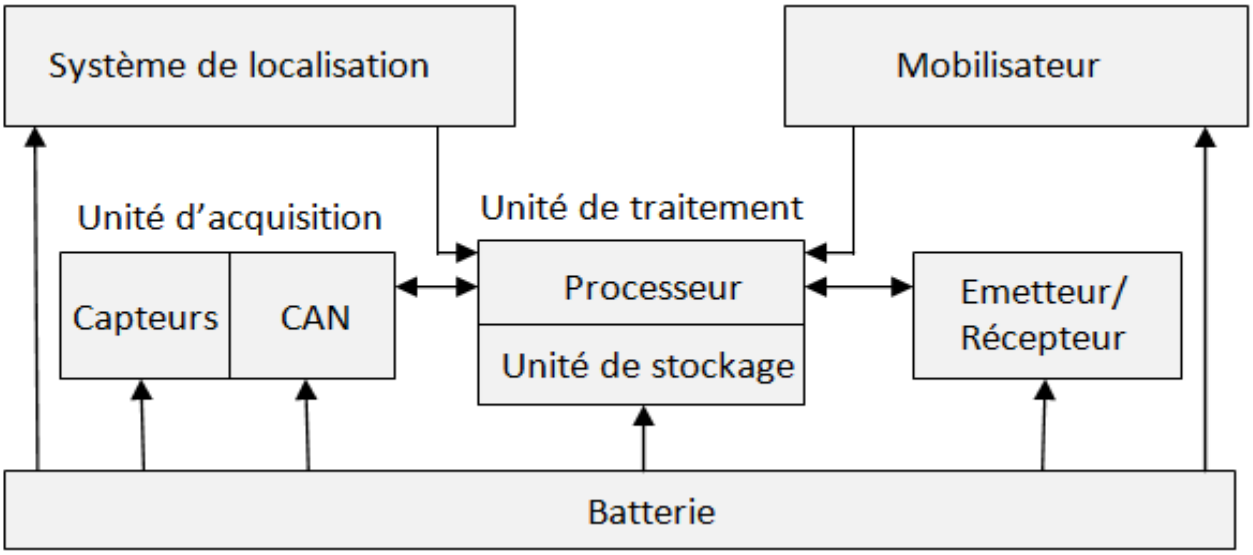



*Figure 1.2: Les composants de base d'un nœud de capteur*

## 1.4. Définition d'un réseau de capteur sans fil

Un Réseau de Capteurs Sans Fil (RCSF) est un ensemble des nœuds, variant de quelques dizaines d'éléments à plusieurs milliers, communiquant sans fil et capable de récolter et transmettre des données environnementales et de réagir en cas de besoin.

Un RCSF est composé d'un nœud de capteurs récepteur ou station de base (Sink Node) et un grand ensemble des nœuds de capteurs déployés sur une grande zone géographique (Sensor Field).[26]

Les données sont transférées des nœuds vers le puits (Sink node).

La figure 1.3 représente l'architecture d'un RCSF

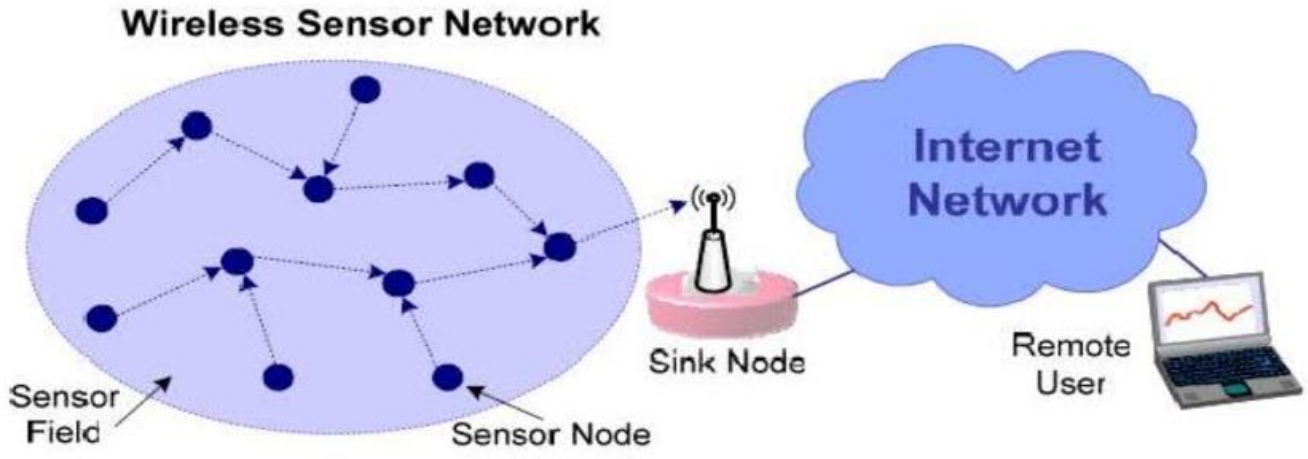



*Figure 1.3: L'architecture d'un RCSF [57]*

## 1.5. Caractéristique des RCSF

Les principales caractéristiques d'un RCSF comprennent

1. Contraintes de consommation électrique pour les nœuds utilisant des batteries ou la récupération d'énergie. Des exemples de fournisseurs sont (ReVibe Energy) [44] et (Perpetuum) [45]
2. Capacité à faire face aux défaillances de nœuds (résilience)
3. Une certaine mobilité des nœuds (pour les nœuds très mobiles)
4. Hétérogénéité des nœuds
5. Homogénéité des nœuds
6. Évolutivité à grande échelle de déploiement
7. Capacité à résister à des conditions environnementales difficiles et hostiles
8. Facilité d'utilisation
9. Optimisation inter-couches [46][47][48]

Ainsi, l'inter-couches peut être utilisée pour effectuer la modulation optimale afin d'améliorer les performances de transmission, telles que le débit de données, l'efficacité énergétique, la qualité de service (QoS), etc. [47] Les nœuds capteurs peuvent être imaginés comme de petits ordinateurs extrêmement basiques en termes d'interfaces et de composants.

Ils se composent généralement d'une unité de traitement avec une puissance de calcul et une mémoire limitée, des capteurs ou MEMS (y compris des circuits de conditionnement spécifiques), un dispositif de communication (généralement des émetteurs-récepteurs radio ou alternativement optiques) et une source d'alimentation généralement sous la forme d'une batterie.

D'autres inclusions possibles sont des modules de récupération d'énergie,[49] des ASIC secondaires et éventuellement une interface de communication secondaire (par exemple RS-232 ou USB).

Les stations de base sont un ou plusieurs composants du RCSF avec beaucoup plus de ressources de calcul, d'énergie et de communication. Ils agissent comme une passerelle entre les nœuds de capteur et l'utilisateur final car ils transmettent généralement les données du RCSF à un serveur. D'autres composants spéciaux dans les réseaux basés sur le routage sont les routeurs, conçus pour calculer, calculer et distribuer les tables de routage.[50]

## 1.6. Les types des RCSF

En fonction de l'environnement, les types de réseaux sont décidés pour que ceux-ci puissent être déployés sous l'eau, sous terre, sur terre, etc. Les différents types des RCSFs incluent :

### 1.6.A. RCSF Terrestre

Les RCSF terrestres sont capables de communiquer efficacement avec les stations de base et se composent de centaines à des milliers de nœuds de capteurs sans fil déployés de manière non structurée (ad hoc) ou structurée (préplanifiée). Dans un mode non structuré, les nœuds capteurs sont répartis de manière aléatoire dans la zone cible qui tombe d'un plan fixe. Le mode préplanifié ou structuré prend en compte le placement optimal, le placement de grille et les modèles de placement 2D et 3D. [24]

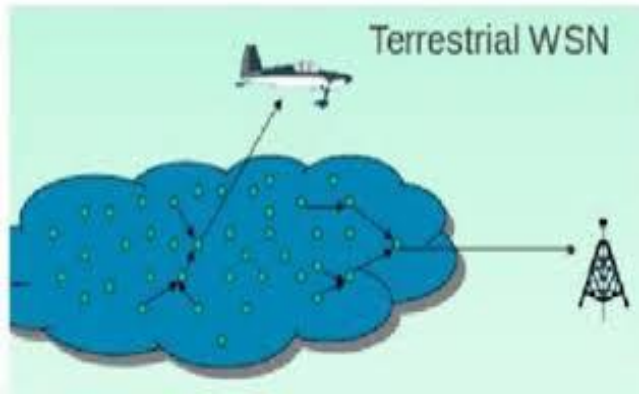



*Figure 1.4: RCSF terrestre [24]*

### 1.6.B. RCSF Sous-terrain

Les réseaux de capteurs sans fil souterrains sont plus chers que les RCSFs terrestres en termes de déploiement, de maintenance et de considérations de coût d'équipement et de planification minutieuse. Les réseaux RCSFs se composent de plusieurs nœuds de capteurs qui sont cachés dans le sol pour surveiller les conditions souterraines. Pour relayer les informations des nœuds capteurs à la station de base, des nœuds récepteurs supplémentaires sont situés au-dessus du sol.[24]

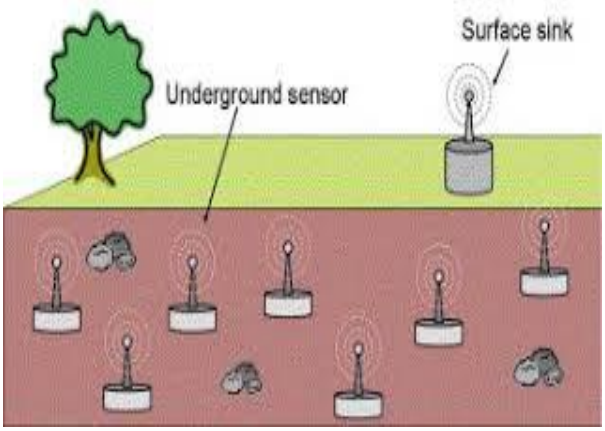



*Figure 1.5: RCSF Sous-terrain [58]*

Les réseaux de capteurs sans fil souterrains déployés dans le sol sont difficiles à recharger. Les nœuds de batterie de capteurs équipés d'une batterie limitée sont difficiles à recharger. En plus de cela, l'environnement souterrain rend la communication sans fil un défi en raison du niveau élevé d'atténuation et de perte de signal.

### 1.6.C. RCSF Sous-marin

Plus de 70% de la terre est occupée par de l'eau. Ces réseaux se composent de plusieurs nœuds de capteurs et de véhicules déployés sous l'eau. Des véhicules sous-marins autonomes sont utilisés pour collecter des données à partir de ces nœuds de capteurs. Un défi de la communication sous-marine est un long délai de propagation et des défaillances de la bande passante et des capteurs. [24]

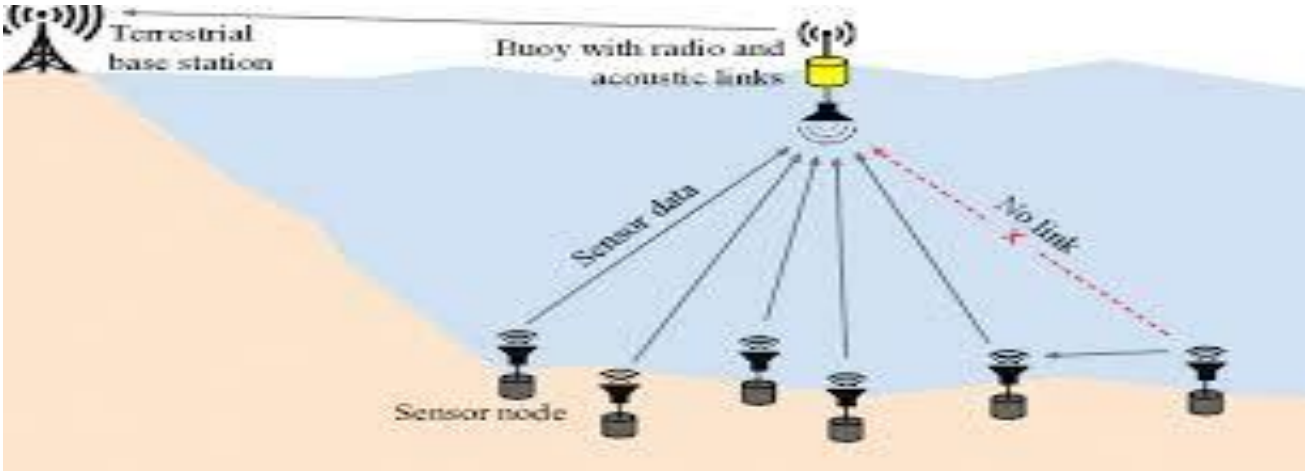



*Figure 1.6: RCSF Sous-marin [59]*

### 1.6.D. RCSF Multimédia

Des réseaux de capteurs sans fil multimédias ont été proposés pour permettre le suivi et la surveillance d'événements sous forme multimédia, tels que l'imagerie, la vidéo et l'audio. Ces réseaux sont constitués de nœuds de capteurs à faible coût équipés de microphones et de caméras. Ces nœuds sont interconnectés les uns aux autres via une connexion sans fil pour la compression de données, la récupération de données et la corrélation.[24]


*Figure 1.7: RCSF Multimédia [60]*

### 1.6.E. RCSF Mobile

Les nœuds mobiles peuvent s'organiser dans le réseau en addition de pouvoir détecter, calculer et communiquer. Les RCSFs mobiles rencontrent plusieurs défis tels-que le déploiement, la gestion de mobilité, localisation avec mobilité etc. [24]
Les réseaux de capteurs sans fil mobiles sont beaucoup plus polyvalents que les réseaux de capteurs statiques. Les avantages de MRCSFs par rapport aux réseaux de capteurs sans fil statiques incluent une couverture meilleure et améliorée, une meilleure efficacité énergétique, une capacité de canal supérieure, etc.

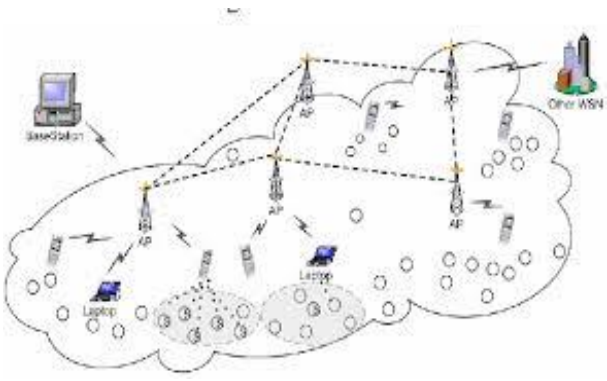



*Figure 1.8: RCSF Mobile [61]*

# 1.7. La pile protocolaire

La pile protocolaire utilisée par la station de base ainsi que tous les autres capteurs du réseau est illustrée par la figure 1.9. La pile protocolaire référencée au modèle de couche OSI, comprend la couche application, la couche transport, la couche réseau, la couche liaison de données et la couche physique, de plus, elle rajoute des plans de gestion qui sont : le plan de gestion de l'énergie, le plan de gestion de la mobilité et le plan de gestion des tâches.[39]

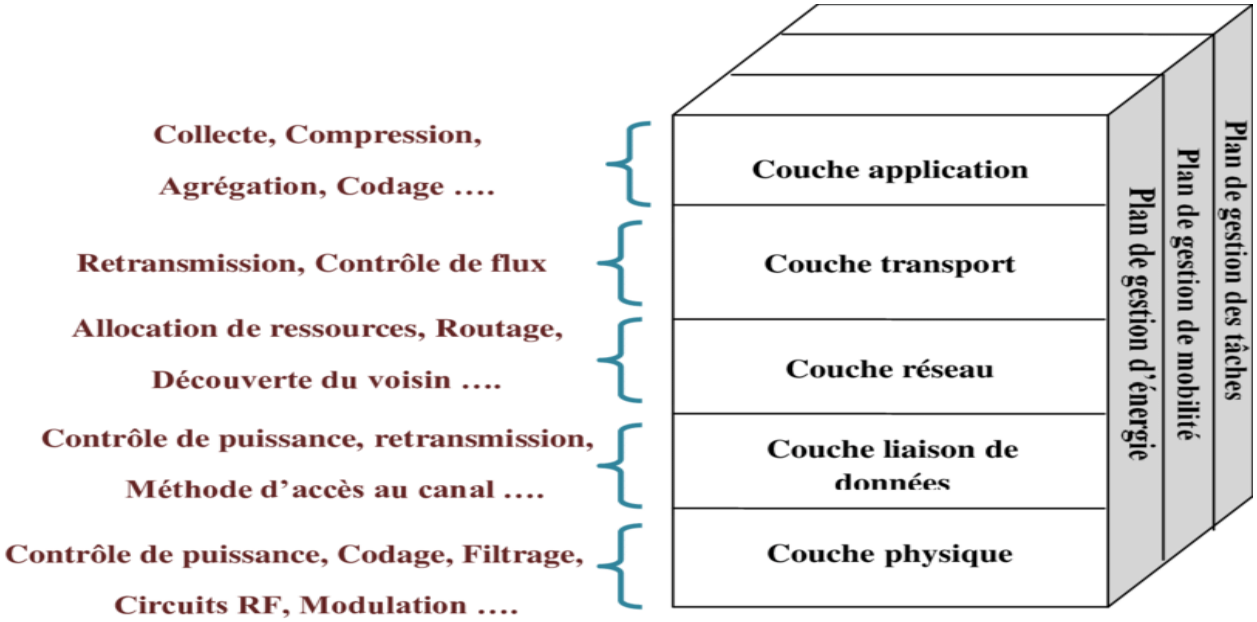



*Figure 1.9: La pile protocolaire*

## 1.7.A. La couche physique

La couche physique est responsable de la sélection de fréquence, modulation simple et robuste, détection de signal, cryptage, transmission et réception des données. Cette couche répond également aux besoins d'une technique de modulation pour affecter la puissance requise.[25]

## 1.7.B. La couche liaison de données

Cette couche est responsable de la création de l'infrastructure réseau et du partage et de l'accès aux supports de communication parmi les nœuds de capteurs. Multiplexage de flux de données, données la détection de trame, le contrôle des erreurs sont également des tâches de liaison de données.[25]

## 1.7.C. La couche réseau

Les nœuds de capteur sont dispersés de manière dense dans un champ et pour cela des protocoles de routage sans fil multi-sauts sans fil spéciaux entre les nœuds de capteur et le nœud récepteur sont nécessaires. Les protocoles de routage ad hoc traditionnels ne peuvent pas être utilisés pour le capteur réseau. [25]

### 1.7.D. La couche transport

Cette couche est requise lors de l'accès au système à travers l'Internet. TCP avec son mécanisme de fenêtre de transmission convient aux caractéristiques de l'environnement du réseau de capteurs. Le fractionnement TCP est nécessaire pour interagir avec d'autres réseaux tels qu'Internet. Les connexions TCP sont créées entre le récepteur et le nœud de capteur et la communication entre l'utilisateur et le nœud de récepteur se fait par UDP ou TCP via Internet ou par satellite.[25]

### 1.7.E. La couche application

Cette couche est la plus proche des utilisateurs et permet d'assurer une interface avec les applications. Et existe trois possible protocoles de la couche application : SMP – TADAP – SQDDP [25]

### 1.7.F. Plan de gestion de gestion d'énergie

Elle est responsable de la façon dont un nœud de capteur utilise sa puissance. Un nœud de capteur peut désactiver son message de réception de ses voisins pour obtenir des messages en double. Lorsque le niveau de puissance d'un capteur est faible, le nœud diffuse l'état de faible puissance à ses voisins et arrête participer au routage. La puissance redondante est réservée pour la tâche de détection.[25]

### 1.7.G. Plan de gestion de la mobilité

Elle détecte et conserve des enregistrements le mouvement des nœuds de capteurs. Ce résultat retourne à l'utilisateur et les nœuds capteurs peuvent entretenir leur capteur.[25]

### 1.7.H. Plan de gestion des tâches

Elle planifie et équilibre les tâches de détection données à une région spécifique. Basé sur le niveau de puissance, pas tous les capteurs des nœuds dans une région sont nécessaires pour effectuer la tâche de détection à le même temps. Ce plan de gestion est nécessaire car -les nœuds capteurs peuvent fonctionner ensemble de manière économe en énergie, acheminer les données dans un réseau de capteurs mobiles et partager des ressources entre les nœuds de capteur.[25]

## 1.8. La sous-couche MAC

L'objectif principal de la couche MAC est d'assurer une transmission de données fiable sur la liaison que la couche physique a déjà déterminée. Et fait partie de la couche liaison de données spécifiée dans la pile de protocoles de communication et est représentée dans figure 1.10 [27]. En outre, la couche MAC détermine la façon dont l'accès est contrôlé dans le canal de communication, une fonction fondamentale dans le cas de RCSF de diffusion (Broadcast) où le support physique est partagé par un grand nombre de capteurs. Plus précisément, dans n'importe quel réseau de diffusion, la question importante est de savoir comment déterminer quel nœud utilise le canal sans fil à quel moment et sur quelle fréquence. Donc, Une régulation de la transmission des messages est nécessaire pour parvenir à une allocation de canal efficace entre les nœuds.

La couche MAC et ses protocoles associés qui définissent les règles de communication entre le nœud émetteur et le nœud de réception, se réfèrent principalement aux mécanismes qui contrôlent la synchronisation des intervalles de fréquence pour l'envoi d'un message (paquet) à travers le canal et à l'écouter.
Protocoles MAC efficaces utiliser la radio judicieusement pour conserver son énergie. Ainsi, le protocole MAC aide à atteindre d'importants objectifs de conception des RCSFs en spécifiant comment les nœuds utilisent la radio, partagent le canal, évitent les collisions en corrélation et en diffusion environnements, répondre au demandeur en temps opportun et survivre plus longtemps.
Par conséquent, concevoir de nouvelles solutions pour Les protocoles MAC pour les RCSFs ont été et resteront un point focal pour de nombreux chercheurs.[27]

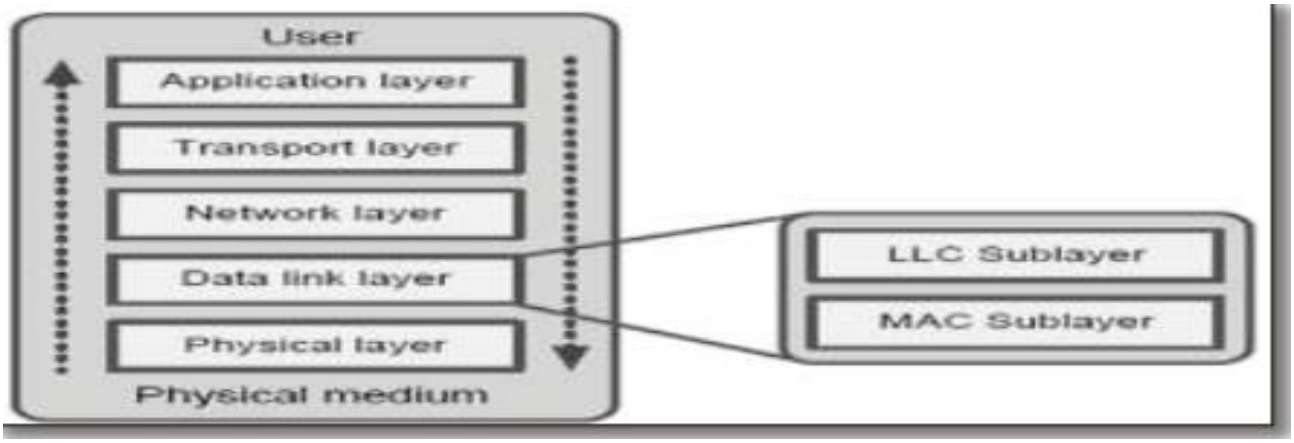



*Figure 1.10: Représentation de la sous-couche MAC [27]*

Table 1 : Classification des certains MAC protocoles dans RCSF [23]

| RCSF MAC Protocole | Organisation |
|---|---|
| TRAMA | Frames |
| BuzzBuzz | Random |
| DW-MAC | Slots |
| X-MAC | Random |
| B-MAC | Random |
| WiseMAC | Random |
| PW-MAC | Hybrid |

## 1.9. Le gaspillage d’énergie dans les RCSF

Dans les RCSFs, l'énergie est gaspillée par les capteurs lors de la détection, traiter, transmettre ou recevoir des données pour remplir la mission requis par l'application. L'acquisition des données se fait par sous-système de détection. Le système de communication est une source gourmande de gaspillage d'énergie. Une grande quantité d'énergie est gaspillé dans des états inutiles depuis le point d'application de vue tels que :[8]

### 1.9.A. Sur- émetteur (*over-emitting*)

Lorsque le nœud de destination n'est pas prêt, l'énergie peut être gaspillée et perdue et cela est dû à la continuité de la transmission du message.

### 1.9.B. L'écoute (*overhearing*)

Lorsqu'un nœud reçoit des paquets qui ne sont pas destinés à lui être envoyés, cela nous cause le problème de perdre de l'énergie et de gaspiller et ce cas est appelé l'écoute ou bien (overhearing).

### 1.9.C. L'écoute inactive (*idle listening*)

Afin de recevoir un trafic, il reste à l'écoute d'un canal inactif alors qu'il ne devrait pas, et cela peut provoquer une gaspillage d'énergie, ce cas est appelé écoute inactive (idle listening).

### 1.9.D. Collision

Cela peut arriver lorsqu'un certain nœud et au même moment reçoit deux ou plus d'un seul paquet, et cela peut nous exposer au problème de collision.

### 1.9.E. La surcharge des paquets de contrôle (*control-packet overhead*)

Dans tous les systèmes normaux et afin d'effectuer la transmission de données, nous devons définir un nombre minimum de paquets de contrôle, pour éviter un tel problème, et ce cas se produit uniquement à la suite d'une surcharge de paquets de contrôle.

## 1.10. Mécanismes d'économie d'énergie

Plusieurs approches doivent être exploitées, même simultanément, pour réduire la consommation d'énergie dans les réseaux de capteurs sans fil.

### 1.10.A. Duty cycling

Il se concentre principalement sur le sous-système de mise en réseau. Le plus efficace l'opération d'économie d'énergie met l'émetteur-récepteur radio en mode veille (faible consommation) chaque fois que la communication n'est pas requise. Idéalement, la radio doit être éteinte dès qu'il n'y a plus de données à envoyer / recevoir et doit être repris dès qu'un nouveau paquet de données devient prêt. De cette façon, les nœuds alternent entre les périodes active et de sommeil en fonction activité du réseau. Le cycle de service est défini comme la fraction des nœuds de temps qui sont actifs pendant leur vie.[10]

### 1.10.B. Data-driven

La transmission de données inutiles et redondantes prend beaucoup de temps et d'énergie. Donc cette approche présente la nécessité de ne pas transmettre des données redondantes au puits.[9]

### 1.10.C. Sleep/Wake-up

Les protocoles de Sleep /wake-up peuvent être classés en trois catégories [9] :

- **Les protocoles à la demande**

Un nœud de capteur ne doit être réveillé qu'à la demande.
Pour informer un nœud qui est en veille qu'il y a des voisins qui souhaitent communiquer avec lui, les protocoles de veille / réveil utilisent certaines radios à faible puissance et vitesse. Pour recevoir les données, un nœud relais est obligé d'attendre que le nœud se réveille.

- **Les rendez-vous planifient**

Tous les nœuds doivent être réveillés en même temps. Les nœuds capteurs se réveillent et restent actifs pendant une courte période pour communiquer avec ses plus proches, puis s'endorment jusqu'à la prochaine réunion.

- **Schéma asynchrone**

Afin de réduire la contention et la charge de travail, nous devons permettre aux nœuds capteurs de gérer leurs horaires de veille/réveil. Nous donnons à tous les nœuds la possibilité de sélectionner l'heure appropriée pour se réveiller pour une communication.
Mais à condition que les nœuds n'aient pas à être synchronisés entre eux, de sorte que chaque nœud RCSF définisse son propre planning indépendamment des autres.

## 1.11. Les protocoles de routage dans les RCSF

Le routage dans les réseaux de capteurs sans fil diffère du routage conventionnel dans les réseaux fixes en différentes manières. Il n'y a pas d'infrastructure, les liaisons sans fil ne sont pas fiables, les nœuds de capteurs peuvent échouer et les protocoles de routage doivent répondre à des exigences strictes en matière d'économie d'énergie

### 1.11.A. Classification des protocoles de routage

Il existe de nombreuses façons de classer les protocoles de routage de WSN. La classification de base des protocoles de routage est illustrée à la figure 11.[11]

- **Centré sur les nœuds** *(Node centric)*

Dans les protocoles centrés sur le nœud, le nœud de destination est spécifié avec des identifiants numériques et ce type de communication n'est pas attendu dans les réseaux de capteurs sans fil. Par exemple. Low energy adaptive clustering hierarchy (LEACH).[11]

- **Centré sur les données** (*Data-centric*)

Dans la plupart des réseaux de capteurs sans fil, les données ou informations détectées sont bien plus précieuses que le nœud lui-même. Par conséquent, les techniques de routage centrées sur les données se concentrent principalement sur la transmission d'informations spécifiées par certains attributs plutôt que sur la collecte de données à partir de certains nœuds. Dans le routage centré sur les données, le nœud récepteur interroge des régions spécifiques pour collecter les données de certaines caractéristiques donc un schéma de dénomination basé sur des attributs est nécessaire pour décrire les caractéristiques des données.[11]

- **Initié par la destination** (*Destination-initiated*)

Les protocoles sont appelés protocoles initiés par la destination lorsque la génération de configuration de chemin commence depuis le nœud de destination. Des exemples sont la diffusion dirigée (DD) & LEACH.[11]

- **Initié par la source** (*Source-initiated)*

Dans ces types de protocoles, le nœud source annonce quand il a des données à partager, puis le route est généré du côté source à la destination. Des exemples sont : SPIN.[11]

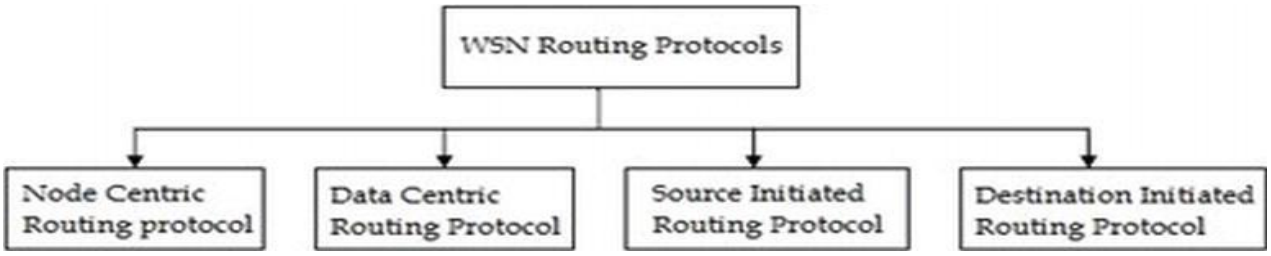



*Figure 1.11: Classification des protocoles de routage [11]*

# 1.12. Catégories des protocoles de routage

Des nombreux algorithmes de routage ont été développés pour les réseaux sans fil en général. Tous les principaux protocoles de routage proposés pour les RCSFs peuvent être catégorisé comme suit [11] :

## 1.12.A. Protocoles Basés sur l'emplacement (*Location-based)*

Également appelés protocoles de routage géographique, reposent sur les informations de localisation des nœuds au lieu des informations de connectivité topologique pour prendre des décisions de routage. Dans le routage basé sur l'emplacement unicast, les paquets sont envoyés directement à une seule destination, qui est identifiée par son emplacement.
Autrement dit, un expéditeur doit être conscient non seulement de son propre emplacement, mais également de l'emplacement de la destination. Dans le routage basé sur l'emplacement de Broadcast ou de multicast approches, le même paquet doit être diffusé vers de multiples destinations.

Les protocoles de multicast tirent parti des emplacements de destination connus pour minimiser la consommation de ressources en réduisant les liens redondants.

## 1.12.B. Protocoles centrés sur les données (*Data Centric*)

Dans les protocoles centrés sur les données, lorsque les capteurs sources envoient des données au puits, les capteurs intermédiaires peuvent effectuer une certaine forme d'agrégation sur les données provenant de plusieurs sources des capteurs et les renvoyer au puits.
Nous pouvons générer des économies d'énergie en raison de la réduction de la transmission requise pour envoyer les données des sources au puits.

### 1.12.C. Protocoles Hiérarchique (*Hirarchical*)

L'agrégation est une méthode écoénergétique qui peut être utilisée par les capteurs pour signaler leurs données détectées au puits. Nous avons un réseau composé de nombreux groupes de capteurs, pour chaque groupe de ces capteurs, nous avons un nœud clé ou un nœud de base (puits) dont son but principal est de gérer tous l'activité dans son groupe comme les données transmises.

### 1.12.D. Protocoles Basé sur mobile (*Mobile-based*)

Un réseau avec un puits fixe souffre de nombreux problèmes, l'un d'entre eux est le "problème de puits d'énergie", où les capteurs situés autour du puits fixe sont fortement utilisés pour transmettre des données au puits au nom de tous les autres capteurs. À cause de cela, ceux dont les capteurs sont chargés à proximité du nœud récepteur déchargent leur batterie plus rapidement, déconnectant ainsi le réseau.

### 1.12.E. Protocoles basés sur des trajets multiples (*Multipath-based*)

- routage mono-chemin : pour chaque capteur source, il envoie les données au nœud récepteur par le chemin le plus court trouvé.
- routage multi-chemin (multiple) : pour chaque capteur source, il trouve les n premiers chemins les plus courts vers le nœud de base et répartit les données de manière égale entre ces chemins courts.

### 1.12.F. Protocoles basés sur la QoS (*QoS-based*)

Nous devons prendre en considération les conditions de la qualité de service (QoS) telles que le temps nécessaire pour transmettre un paquet ou un groupe de paquets de l'extrémité de transmission à l'extrémité de réception, et la capacité de continuer à fonctionner malgré les pannes, et enfin la fiabilité, si jamais nous voulons minimiser la consommation d'énergie dans un nœud de capteur sans fil.

Dans ce tableau ci-dessous on présentera des exemples sur chaque type de protocoles

*Table 2 : Les Protocoles de routage dans les RCSFs [11]*

| **Catégorie** | **Protocoles** |
|---|---|
| Location-based Protocols | MECN, SMECN, GAF, GEAR, Span, TBF, BVGF, GeRaF |
| Data-centric Protocols | SPIN, Directed Diffusion, Rumor Routing, COUGAR, ACQUIRE, EAD, Information-Directed Routing, Gradient Based Routing, Energy-aware Routing, Information-Directed Routing, Quorum-Based Information Dissemination, Home Agent Based Information Dissemination |
| Hierarchical Protocols | LEACH, PEGASIS, HEED, TEEN, APTEEN |
| Mobility-based Protocols | SEAD, TTDD, Joint Mobility and Routing, Data MULES, Dynamic Proxy Tree-Base Data Dissemination |
| Multipath-based Protocols | Sensor-Disjoint Multipath, Braided Multipath, N-to-1 Multipath Discovery |
| QoS-based protocols | SAR, SPEED, Energy-aware routing |

## 1.13. Facteurs principaux de conception

La conception et la réalisation des réseaux de capteurs sans fil sont influencées par plusieurs paramètres, les plus importants sont présentés comme suit :

### 1.13.A. Durée de vie

C'est la caractéristique la plus fondamentale d'un réseau de capteurs. Elle dépend du type d'application et donc de la durée et de l'échantillonnage des mesures. Les contraintes liées au changement (ou rechargement) des batteries sont dépendantes des déploiements et du coût de maintenance des nœuds. Il est donc essentiel d'avoir une durée de vie du réseau la plus longue possible.

### 1.13.B. Étendu de réseau

La plupart des réseaux de capteurs sont composés de quelques dizaines de nœuds, mais certaines applications peuvent exiger l'utilisation de réseaux de capteurs composés de centaines ou de milliers de nœuds. La zone que doit couvrir le réseau est également importante dans son dimensionnement.

### 1.13.C. Faible coût

Les réseaux de capteurs peuvent contenir un nombre important de nœuds. Il est donc nécessaire d'avoir un coût unitaire par nœud le plus faible possible, pour obtenir un coût raisonnable du réseau global.

### 1.13.D. Scalabilité

Dans le cas d'un nœud endommagé, le réseau doit être capable de prendre en considération cette modification tout en assurant une qualité de service égale. La redondance des capteurs peut être un moyen d'assurer cette fonction. La notion de scalabilité est alors utilisée pour dire que l'architecture et les protocoles de communications du réseau doivent s'adapter et prendre en compte l'ajout ou la perte de nœuds dans le réseau.

### 1.13.E. Faible consommation

La durée de vie la plus longue possible traduit l'exigence la plus importante de la plupart des applications. Par conséquent, pour atteindre cette autonomie, il est crucial de minimiser la consommation moyenne des capteurs. Une des alternatives explorées aujourd'hui par les chercheurs consiste à extraire l'énergie de l'environnement (énergie solaire, vibrations mécaniques, bruit acoustique…). Ces techniques peuvent grandement améliorer la durée de vie, mais comme la production d'énergie est très faible, une consommation d'énergie réduite des capteurs reste de la plus haute importance.

### 1.13.F. Faible complexité matérielle et logicielle

Les fonctionnalités mises en œuvre par la partie matérielle se doivent d'être aussi simples que possible, car l'augmentation de la complexité de cette dernière peut conduire à une augmentation de la consommation d'énergie. La complexité de la partie logicielle doit également être faible sous peine d'augmenter les consommations liées aux accès mémoire.

### 1.13.G. Autoconfiguration

Un réseau de capteurs doit pouvoir configurer tous ses paramètres indépendamment de son environnement d'installation. Selon le nombre de nœuds, et selon leur déploiement, une configuration manuelle n'est pas du tout envisageable. Le réseau doit par exemple être capable d'identifier les positions des nœuds, ce qui lui permettra d'identifier et de tolérer d'éventuelles pannes (problème de batterie) ou bien encore d'intégrer de nouveaux nœuds.
Donc pour concevoir un RCSF il est essentiel de tenir compte des paramètres cités ci-dessus, en effet ces facteurs servent comme directives pour le développement des algorithmes et protocoles utilisés dans les RCSF.

## 1.14. Les différentes applications des RCFS

Le coût et la taille diminuée des capteurs et l'évolution des supports de communication sans fils, ont élargi le domaine d'application des RCSFs, parmi elles, nous citons :

### 1.14.A. Application médicale

L'implémentation des micro-capteurs sous la peau peut faciliter la surveillance des organismes vivant. Ils peuvent émettre des images ou des vidéos en temps réel d'une partie du corps sans recours à la chirurgie pendant environ 24h. On peut également surveiller la progression d'une maladie ou la reconstruction d'un muscle [12]

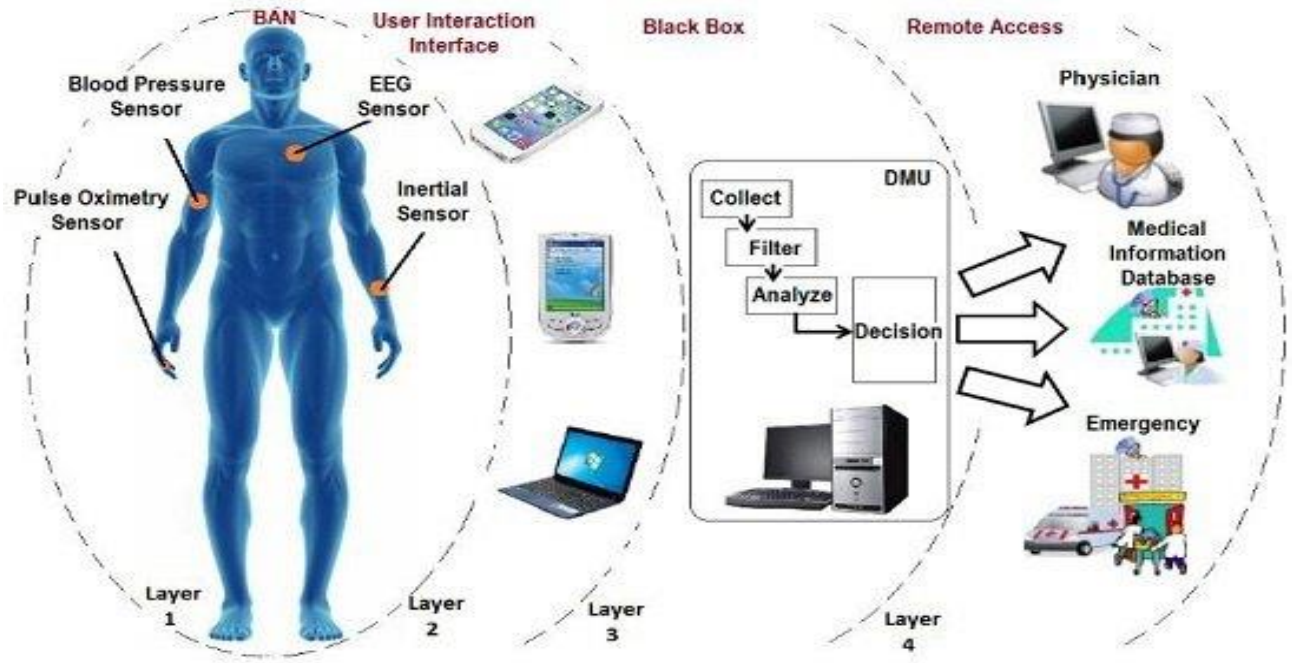



*Figure 1.12: Application médicales [62]*

### 1.14.B. Application militaire

Les RCSFs peuvent être un facteur majeur dans les stratégies militaires, où ils permettent par exemple de surveiller les mouvements des ennemies, surveiller les frontières, analyses les terrains avant d'envoyer des troupes, détecter les attaques nucléaires et chimiques et cibler les bases d'ennemies. Les réseaux de capteurs, vu leurs caractéristiques (tolérance aux pannes, moins coûteux, déployables évolutifs, …etc.) représentent une technologie très appréciable dans ce domaine, qui est devenu éventuellement, un moteur initial dans le développement des RCSFs.[12]

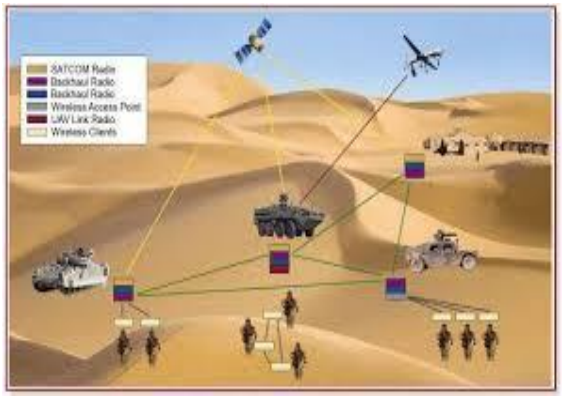



*Figure 1.13: Application militaires [63]*

### 1.14.C. Application environnementale

Les RCSFs permettent une étude plus profonde de l'environnement, y compris le signal des catastrophes naturelles à venir, le suivi des mouvements des animaux et des insectes, détection biochimique et biologique, surveillance podologique et atmosphérique, surveillance de contamination et de pollution et le contrôle d'agriculture. Le déploiement de plusieurs micro-capteurs dans l'environnement a minimisé plusieurs défis dans ce domaine.[12]

### 1.14.D. Application de sécurité

Les RCSFs de mouvement peuvent fournir un système d'alarme adéquat qui aura comme but la détection des intrusions ou des cambriolages. De plus, ils peuvent aider la surveillance des routes et des voies ferrées pour éviter des accidents. Ces RCSFs peuvent diminuer les failles de sécurité et créer des systèmes de sécurité à un coût diminué. [12]

## 1.15. Solution proposée

Pour modéliser notre système, on opte à utiliser un système de file d'attente avec vacance et avec priorité pour la modélisation du buffer de nœud de capteur sans fils, ce qui répond aux besoins de conception de notre système et nous permettra de manipuler et gérer les événements occurrents dans ce dernier.

## 1.16. Conclusion

Donc à la fin de ce chapitre, nous pouvons dire que nous avons illustré toutes les caractéristiques et l'architecture nécessaires à la fois du nœud capteur et du nœud capteur sans fil, et nous savons que l'un des problèmes majeurs d'un nœud de capteur est la consommation d'énergie en raison de la faible capacité de la batterie

Dans le prochain chapitre, nous allons voir tous les aspects et caractéristiques et mesures de tous les types de la théorie des files d'attente, et pour cela peut nous aider à construire un modèle de simulation afin d'évaluer ses mesures

# 2. *Les files d'attente avec vacance*

# 2.1. Introduction

Les files d'attente sont une partie essentielle de notre vie quotidienne. Ils se produisent chaque fois qu'il y a concurrence pour des ressources limitées. Par exemple, nous faisons la queue devant les cabinets médicaux et les caisses des supermarchés ; les avions s'alignent le long des pistes de l'aéroport ; et des demandes pour satisfaire la file d'attente de défauts de page du système d'exploitation sur un disque dur d'ordinateur. [13]
Ainsi dans ce chapitre, nous présenterons des modèles de files d'attente standard et de files d'attente avec priorité et de files d'attente à vide et leurs caractéristiques et nous introduirons les concepts de probabilité et de chaîne de Markov et le processus stochastique et les paramètres de performance d'un système de files d'attente.

# 2.2. Les files d’attente

La file d’attente est une méthode de recherche mathématique relevant du la probabilité [15], qui permet de modéliser un système admettant un phénomène d’attente, de calculer ses mesures et la détermination de ses caractéristiques et l’études des solutions optimales pour nous aider a prises des décisions [14].

## 2.2.A. Démonstration d'un modèle de file d'attente

Un système de file d'attente est illustré dans la figure 2.1. Les clients arrivent à un taux moyen de λ. Ils sont servis sans délai s'il y a des serveurs disponibles ; sinon, ils sont faits attendre dans la file d'attente jusqu'à ce que ce soit leur tour d'être servi. Une fois servis, ils sont supposés partir le système [16].

Nous serons intéressés à déterminer des quantités telles que le nombre moyen de clients dans le système, le temps moyen qu'un client passe dans le système, le temps moyen passé à attendre dans la file d'attente, etc.

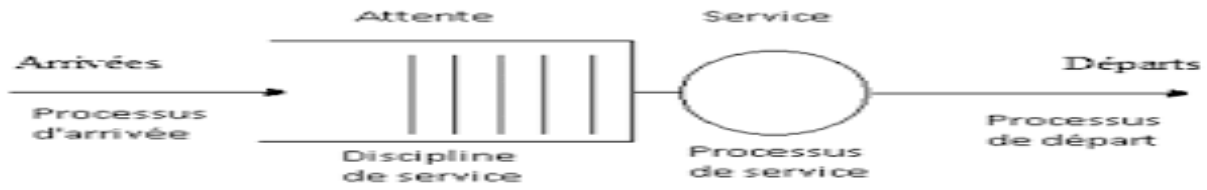



*Figure 2.1: un système de file d'attente*

## 2.2.B. Notation de Kendall

Les mêmes modèles de file d'attente peuvent apparaître dans des champs d'application complètement différents.
Pour éviter le développement parallèle des mêmes modèles dans différents domaines, en 1953, Kendall a proposé une classification et une notation standard des systèmes de file d'attente de base.

La version actuelle de cet ensemble de notations est composée de six éléments - **A / B / c / d / e / x –** où : [18]
**A** est le type de processus d'arrivée.
**B** est le type de processus de service.
**c** est le nombre de serveurs.
**d** est la capacité du système, le nombre maximum de clients dans le système.
**e** est la population de l'ensemble des clients (si elle est finie, l'intensité d'arrivée diminue avec un nombre croissant de clients dans le système).
**x** définit la discipline de service (les disciplines de service les plus courantes par exemple:

- FCFS : premier arrive, premier servi (FIFO).
- LCFS : dernier entré, premier sorti (LIFO).
- PR : priorité.

Dans les systèmes de file d'attente de base A et B, utilisez l'une des options suivantes :
**M** : sans mémoire (*memoryless*), se réfère au temps d'inter-arrivée ou de service distribué de manière exponentielle.
**Er** : ordre r Erlang distribué entre les arrivées ou le temps de service.
**Hr** : ordre r distribué de manière hyper-exponentielle entre les arrivées ou le temps de service.
**D** : temps déterministe entre les arrivées ou le service.
**G** ou **GI** : entre les arrivées aléatoires ou le temps de service avec toute distribution générale.

### 2.2.C. Caractéristiques des files d'attente

La description mathématique des systèmes de mise en file d'attente nécessite une description des éléments suivants [18] :

- **Processus d'arrivée**

La description stochastique des arrivées des clients, où les clients peuvent avoir une signification abstraite ou physique en fonction du système.
Les arrivées de clients peuvent dépendre des propriétés du système actuel, par exemple, nombre de clients dans le système. Dans le cas des modèles de mise en file d'attente de base (où les heures inter-arrivées sont indépendantes), le processus d'arrivée est caractérisé par la distribution du temps entre les arrivées.

- **Processus de service**

La description stochastique du service client. Tout comme l'arrivée du client, le service client peut également dépendre des propriétés du système, et dans les modèles de mise en file d'attente de base, les temps de service sont Indépendant et distribué à l'identique, variables aléatoires.

- **Structure du système**

Les ressources du système de file d'attente, généralement le nombre de serveurs et la taille de la salle d'attente.

- **Discipline du service**

Un ensemble de règles qui détermine l'ordre de service et le service mode des clients. Les commandes les plus courantes sont FIFO (premier entré, premier sorti), LIFO (dernier entré, premier sorti). Les ressources de service peuvent également être utilisées pour servir tous les clients en parallèle. Cette discipline est appelée partage de processeur (PS). L'ordre de service joue un rôle important lorsqu'il est différents types de clients arrivent à un système. Dans ce cas, priorité (avec et sans préemptions) peuvent être utilisées pour fournir un service plus rapide à un type de client.

- **Paramètres de performance**

Pour construire un modèle correctement détaillé d'un système, il faut tenir compte des paramètres de performance qui doivent être calculés. Les paramètres de performance les plus courants sont l'utilisation du système, la moyenne et distribution du temps d'attente, probabilité de perte (la probabilité qu'un client rejetées par le système), etc.

### 2.2.D. Paramètres de performance du système de file d'attente

Lorsque nous analysons un système de file d'attente, nous le faisons dans le but d'obtenir les valeurs de certaines propriétés du système. Par exemple, nous pouvons souhaiter trouver [13]:

- le nombre de clients dans le système.
- le temps d'attente d'un client.
- la durée d'une période d'occupation ou d'inactivité.
- la charge de travail actuelle (en unités de temps).

Ce sont des mesures de l'efficacité. Ce sont toutes des variables aléatoires et, alors que nous pourrions souhaitent connaître leurs descriptions probabilistes complètes, la plupart du temps nous devons se contenter de ne pouvoir déduire que leurs premiers instants (moyenne, variance, etc.). En outre, les modèles mathématiques des systèmes de file d'attente fournissent des réponses quantitatives à des questions spécifiques, telles que comme : « De combien la congestion sera-t-elle réduite si la variation du temps de service est réduite de 10% ? 50% ? » ou « Que se passe-t-il si nous remplaçons deux serveurs lents par un seul serveur rapide ?», et ainsi de suite. Aussi un l'analyste de la file d'attente peut souhaiter concevoir un système optimal qui implique la nécessité d'équilibrer l'attente temps avec temps d'inactivité selon une structure de coût inhérente ; par exemple, pour trouver le nombre optimal de serveurs, par exemple. Nous considérons maintenant les mesures d'efficacité les plus couramment obtenues.

- **Nombre de clients**

Nous laisserons $N$ la variable aléatoire qui décrit le nombre de clients dans le système à régime permanent. La probabilité qu'en régime permanent le nombre de clients présents dans le système soit n est désigné par $Pn$

$$p_n = \text{Prob}\{N = n\},$$

Et le nombre moyen dans le système à l'état d'équilibre est :

$$L = E[N] = \sum_{i=0}^{\infty} n p_n.$$

Dans le système de file d'attente, les clients peuvent être présents dans la file d'attente en attendant leur tour de recevoir service ou ils peuvent recevoir un service. Nous laisserons *Nq* la variable aléatoire qui décrit le nombre de clients en attente dans la file d'attente et nous désignerons sa moyenne par $Lq = E[Nq]$

- **Utilisation du système** *(utilization)*

Dans un système de file d'attente avec un seul serveur ($c = 1$), l'utilisation $U$ est définie comme la fraction de temps pendant lequel le serveur est occupé. Si le taux auquel les clients arrivent et y sont admis mise en file d'attente est $\lambda$ et si $\mu$ est le taux auquel ces clients sont servis, alors l'utilisation est égale à $\lambda / \mu$. Sur une période de temps $T$, ce système de file d'attente, en régime permanent, reçoit une moyenne de clients $\lambda T$, qui sont servis en moyenne $\lambda T / \mu$ secondes. Dans de nombreux systèmes de mise en file d'attente, la lettre $\rho$ est définie comme $\rho = \lambda / \mu$ et est par conséquent identifiée à l'utilisation.

Cependant, $\lambda$ est généralement défini comme le taux d'arrivée dans le système et cela peut ou non être le taux auquel les clients entrent réellement dans l'installation de file d'attente. Ainsi, ce n'est pas toujours le cas que $\lambda / \mu$ définit correctement l'utilisation. Certains clients peuvent se voir refuser l'admission (on dit qu'ils sont Clients « perdus ») de sorte que le taux d'arrivée effectif dans l'installation de file d'attente soit inférieur à $\lambda$ et donc l'utilisation est inférieure à $\rho = \lambda / \mu$. Cependant, sauf indication contraire, nous supposons que tous les clients qui arrivent à une installation de file d'attente sont admis.

Dans le cas de système de file d'attente avec plusieurs serveurs (c>1), l'utilisation est définie comme la fraction moyenne de serveurs actifs, qui correspond simplement à la vitesse à laquelle le travail entre dans le système divisé par la vitesse maximale (capacité) à laquelle le système peut effectuer ce travail, c.-à-d. $U = \lambda / (c\mu)$. Dans les systèmes multi serveurs, il est habituel de définir $\rho$ comme $\rho = \lambda / (c\mu)$ avec la même mise en garde qu'avant de concerner l'identification de $\rho$ comme utilisation.

- **Débit du système** *(Throughput)*

Il est noté $\rho$, le débit d'un système de file d'attente est égal le taux de départ, d'autre façons, c'est-à-dire le nombre moyen de clients servis dans le temps.

- **Intensité du trafic** *(Traffic intensity)*

Nous définissons l'intensité du trafic comme la vitesse à laquelle le travail entre dans le système, elle est donc le produit du taux d'arrivée moyen des clients et du temps moyen de service, c'est-à-dire $\lambda \bar{x} = \lambda / \mu$, où $\bar{x} = 1 / \mu$ et $\mu$ est le taux de service moyen. Notez que dans les systèmes à serveur unique, l'intensité du trafic est égale à l'utilisation. Pour plusieurs serveurs, l'intensité du trafic est égale à $cU$.

- **Temps de réponse (***Response time)*

Lorsqu'un client vient au système, il passe du temps jusqu'à son départ, ce temps dans la théorie des files d'attente, nous le connaissons sous le nom de temps de réponse ou bien le temps de séjour. [19].

- **Temps d’attente (***Waiting time)*

Lorsqu’un client vient au système, il passe du temps en attente à cause de l'occupation du serveur par un autre client, ce temps dans la théorie des files d'attente, nous le connaissons sous le nom de temps d’attente. [19].

- **Nombre moyen de clients en attente**

C’est le nombre des clients qu’on en attente dans la file d’attente dans un moment de temps [19].

## 2.3. Les files d’attente avec priorité

### 2.3.A. Description

Dans un système de file d'attente dans lequel les clients sont distingués par classe, il est habituel d'attribuer des priorités en fonction de l'importance perçue du client. La classe de clients la plus importante se voit attribuer la priorité 1 ; les classes de clients de moindre importance se voient attribuer les priorités 2, 3, . . .. Lorsque le système contient des clients de classes différentes, ceux de priorité j sont servis avant ceux de priorité j + 1, j = 1, 2, . . .. Les clients de chaque classe sont servis dans l'ordre premier arrivé, premier servi. La question est de savoir comment un client en service doit être traité lorsqu'un client de priorité plus élevée arrive. Cela donne lieu à deux politiques d'ordonnancement [13].

### 2.3.B. Les politiques d'ordonnancement

***Priorité préemptive :***

Dans ce cas, un client de priorité inférieure en service est éjecté du service au moment où un client de priorité supérieure arrive. Le client interrompu est autorisé à reprendre du service une fois que la file d'attente ne contient aucun client ayant une priorité plus élevée. L'interruption du service peut signifier que tous les progrès accomplis vers la satisfaction du besoin de service du client éjecté ont été perdus, de sorte qu'il devient nécessaire de recommencer ce service depuis le début. C'est ce qu'on appelle un redémarrage anticipé. Heureusement, dans de nombreux cas, le travail effectué sur le client éjecté jusqu'au point d'interruption n'est pas perdu de sorte que lorsque ce client est à nouveau mis en service, le processus de service peut reprendre là où il s'était arrêté. C'est ce qu'on appelle préempter-reprendre [13].

***Priorité non préemptive :***

Le service d'un client de faible priorité commencera lorsqu'il n'y aura aucun client de priorité plus élevée. Cependant, une fois le service initié sur un client de faible priorité, le serveur a l'obligation de servir ce client jusqu'à la fin, même si un ou plusieurs clients de priorité plus élevée arrivent au cours de ce service [13].

# 2.4. Les files d’attente avec vacances

## 2.4.A. Description

Les chercheurs dans le domaine des files d’attente ont défini les vacances comme un état que le serveur n'est pas disponible pour servis les clients arrives. Les arrivées pendant les vacances ne peuvent entrer en service qu'après le serveur revient de vacances. Il existe de nombreuses situations qui conduisent à des vacances de serveur, c'est-à-dire des pannes de machine, une maintenance des systèmes et des serveurs cycliques (où le serveur sert plus d'une file d'attente dans le système ou sert plus d'un système). [20]

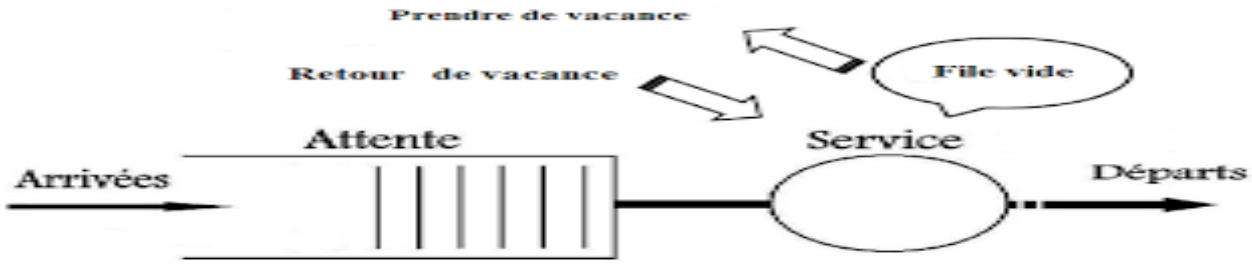



*Figure 2.2: un système de file d'attente avec vacances*

## 2.4.B. Les types de file d’attente avec vacances

- **Le modèle de vacances unique**

Après la fin de chaque période d'activité, nous ne pouvons avoir qu'un seul poste de vacances, si le serveur revient de cette vacance, il ne repart pas en vacances même si le système est vide. [20]

- **Le modèle de vacances multiples**

Ce type de vacances peut résulter de cas comme la maintenance des systèmes informatiques et de communication où les processeurs d'un ordinateur et des systèmes de communication effectuent des tests et une maintenance considérable en plus de leurs fonctions principales (traitement des appels téléphoniques, réception et transmission de données, etc.). Les travaux de maintenance nécessaires sont divisés en courtes segments. Chaque fois que les clients sont absents, le transformateur effectue un segment du travail de maintenance. Lorsque le système est inactif, le serveur prend des vacances (fonctionne sur un segment de maintenance). Au retour de vacances, le serveur ne démarre le service que s'il trouve un certain nombre ou plus de clients en attente, si le nombre en attente est inférieur alors il part en vacances (segment de maintenance).[20]

- **Le modèle de vacances à service limité**

Dans lequel le serveur prend des vacances lorsqu'il devient inactif ou après avoir servi m clients consécutifs, ou après le temps T. [20]

### 2.4.C. Discipline de service

Existe des conditions spécifiées pour que le serveur démarre des vacances telles que : [20]

- **Service fermé :** dès que le serveur revient des vacances, il place une porte derrière le dernier client en attente. Il commence alors à ne servir que les clients qui se trouvent à l'intérieur de la porte, en fonction de certaines règles sur le nombre ou la durée de service.
- **Service exhaustif :** le serveur ne part pas en vacances jusqu'il servit tous les clients dans le système.
- **Service limité :** le serveur ne part en vacances seulement si :
  - (i) le système est complètement vide.
  - (ii) le système atteint une limite maximale des clients traiter.

### 2.4.D. Politiques de fin de vacances

Il existe une autre classification possible qui considère la fin des vacances. Cette classification détermine quand le serveur est réveillé pour servir. [21]

1. **La politique N :**
   Cette politique s’intéresse a le nombre de clients dans la file d’attente, le serveur ne retourne pas de vacance s’il n’a pas atteint un certain seuil N.
2. **La politique T :**
   Cette politique s’intéresse par le temps dans la file d’attente, le serveur ne retourne pas de vacances s’il n’a pas achevé un certain T de temps en vacances.
3. **La politique D :**
   Cette politique s’intéresse par le temps des services des clients dans la file d’attente, le serveur ne retourne pas de vacances si la somme de temps de service de nos clients ne dépasse pas une certaine valeur D.

### 2.4.E. La durée d’une vacance

La durée d’une vacance est le temps quand le serveur prend une vacance jusqu'à le temps de sa réveillant.

## 2.5. La théorie de probabilité

La Théorie des probabilités est l'étude mathématique des phénomènes caractérisés par le hasard et l'incertitude. Les objets centraux de la théorie des probabilités sont les variables aléatoires, les processus stochastiques, et les évènements : ils traduisent de manière abstraite des évènements non déterministes ou des quantités mesurées qui peuvent parfois évoluer dans le temps d'une manière apparemment aléatoire. En tant que fondement mathématique des statistiques, la théorie des probabilités est essentielle à la plupart des activités humaines qui nécessitent une analyse quantitative d'un grand nombre de mesures. Les méthodes de la théorie des probabilités s'appliquent également à la description de systèmes complexes dont on ne connaît qu'en partie l'état, comme en mécanique statistique. Une grande découverte de la

physique du vingtième siècle fut la nature probabiliste de phénomènes physiques à une échelle microscopique, décrite par la mécanique quantique.

### 2.5.A. Loi exponentielle

La fonction de distribution cumulative pour une variable aléatoire exponentielle, X, avec le paramètre λ > 0, est donnée par :

$$F(x) = \begin{cases} 1-e^{-\lambda x}, & x \geq 0, \\ 0 & sinon, \end{cases}$$

The corresponding probability density function is obtained simply by taking the derivative of $F(x)$ with respect to $x$:

$$f(x) = \begin{cases} \lambda\, e^{-\lambda x}, & x \geq 0, \\ 0 & sinon, \end{cases}$$

L'une des propriétés les plus importantes de la distribution exponentielle est qu'elle possède la propriété sans mémoire. Cela signifie que l'histoire passée d'une variable aléatoire à distribution exponentielle ne joue aucun rôle dans la prédiction de son avenir. Par exemple, soit X la variable aléatoire qui dénote la durée qu'un client passe en service (appelée temps de service) dans une installation et supposons que X est distribué de façon exponentielle. Ensuite, la probabilité que le client en service termine à un moment futur t est indépendante de la durée pendant laquelle ce client a déjà été en service. De même, si le temps entre les arrivées (le temps entre les arrivées) des patients au cabinet d'un médecin est distribué de façon exponentielle, alors la probabilité qu'une arrivée se produise au temps t est indépendante de la durée qui s'est écoulée depuis l'arrivée précédente.[13]

### 2.5.B. Variable aléatoire

En théorie des probabilités, une variable aléatoire est une variable dont la valeur est déterminée après un tirage aléatoire. Par exemple, la valeur d'un dé entre 1 et 6, le côté de la pièce dans un pile ou face, etc. C’est une application définie sur l'ensemble des éventualités, c'est-à-dire l'ensemble des résultats possibles d'une expérience aléatoire. Ce furent les jeux de hasard qui amenèrent à concevoir les variables aléatoires, en associant à une éventualité un gain. Cette association éventualité-gain a donné lieu par la suite à la conception d'une fonction de portée plus générale. Le développement des variables aléatoires est associé à la théorie de la mesure.

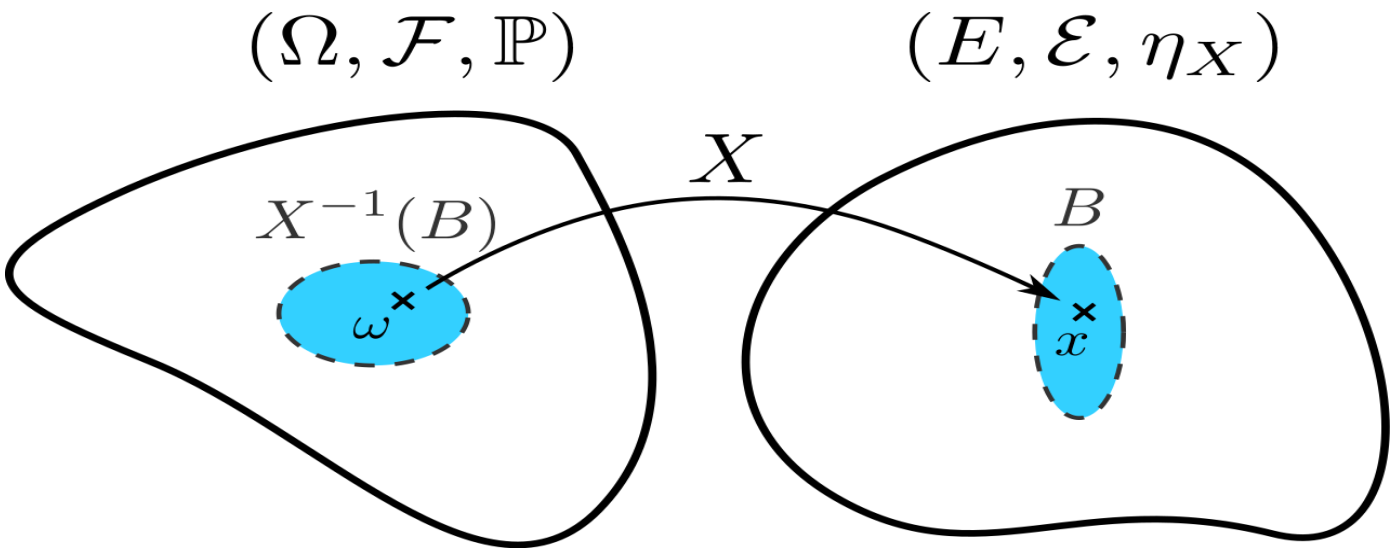



*Figure 2.3: les variables aléatoires*

Une variable aléatoire est une fonction mesurable d'un espace inconnu (l'univers Ω) vers un espace d'état E, connu. À chaque élément mesurable B de E, on associe son image réciproque dans Ω. La mesure $P_x$ (loi de probabilité de X) sur E se déduit donc de la mesure P sur Ω.

### 2.5.C. Les fonctions quantiles

Soit X une variable aléatoire de fonction de distribution F, et soit p ∈ (0,1), Une valeur de x telle que F ($\overline{X}$) = (X < x) <= p et F(x) = ℙ (X <= x) >= p est appelé un quantile d'ordre p pour la distribution. En gros, un quantile d'ordre p est une valeur où le graphe de la fonction de distribution cumulée croise (ou saute) p. Par exemple, dans l'image ci-dessous, a est l'unique quantile d'ordre p et b est l'unique quantile d'ordre q. D'autre part, les quantiles d'ordre r forment l'intervalle [c,d], et de plus, d est un quantile pour tous les ordres dans l'intervalle [r,s ].

Notez qu'il existe une sorte de relation inverse entre les quantiles et les valeurs de distribution cumulative, mais la relation est plus compliquée que celle d'une fonction et de sa fonction inverse ordinaire, car la fonction de distribution n'est pas individuelle en général. À de nombreuses fins, il est utile de sélectionner un quantile spécifique pour chaque commande ; pour cela, il faut définir un inverse généralisé de la fonction de distribution.

## 2.6. Les chaînes de Markov

En mathématiques, une chaîne de Markov est un processus de Markov à temps discret, ou à temps continu et à espace d'états discret. Un processus de Markov est un processus stochastique possédant la propriété de Markov : l'information utile pour la prédiction du futur est entièrement contenue dans l'état présent du processus et n'est pas dépendante des états antérieurs (le système n'a pas de « mémoire »). Les processus de Markov portent le nom de leur inventeur, Andreï Markov.

### 2.6.A. Chaînes de Markov à temps discret

Pour une chaîne de Markov à temps discret, nous observons son état à un ensemble de temps discret, mais infini. Les transitions d'un état à un autre ne peut avoir lieu, ou ne pas avoir lieu, à ces instants temporels, quelque peu abstraits, des instants temporels qui sont pour la plupart considérés comme séparés d'une unité de temps. Par conséquent nous pouvons représenter, sans perte de généralité, l'ensemble d'indices discrets T du processus stochastique sous-jacent par l'ensemble des nombres naturels {0, 1, 2, . . .}. Les observations successives définissent les variables aléatoires X0, X1, . . . , Xn, . . . aux pas de temps 0, 1, . . . , n, . . . , respectivement. Formellement, une chaîne de Markov à temps discret {Xn, n = 0, 1, 2, . . .} est un processus stochastique qui satisfait la relation suivante, appelé la propriété de Markov :[13]

Pour tous les nombres naturels n et tous les états xn,

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

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

Ainsi, le fait que le système soit dans l'état x0 au pas de temps 0, dans l'état x1 au pas de temps 1, et ainsi de suite, jusqu'au fait qu'il soit dans l'état xn-1 au pas de temps n-1 est totalement indifférent. L'état dans lequel se trouve le système au pas de temps n+1 ne dépend que de l'endroit où il se trouve au pas de temps n. Le fait que la chaîne de Markov soit dans l'état xn au pas de temps n est la somme de toutes les informations concernant le l’histoire de la chaîne pertinente pour son évolution future.[13]

Pour simplifier la notation, plutôt que d'utiliser xi pour représenter les états d'une chaîne de Markov, nous utiliserons désormais des lettres simples, telles que i, j et k. Les probabilités conditionnelles Prob{Xn+1 = xn+1|Xn = xn}, maintenant écrites comme Prob{Xn+1 = j|Xn = i}, sont appelées la transition à une étape probabilités, ou simplement les probabilités de transition, de la chaîne de Markov. Ils donnent la probabilité conditionnelle de passer de l'état xn = i à l'état xn+1 = j lorsque le paramètre temporel passe de n à n + 1. Ils sont notés :[13]

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

### 2.6.B. Chaînes de Markov à temps continu

En théorie des probabilités, un processus de Markov à temps continu, ou chaîne de Markov à temps continu est une variante à temps continu du processus de Markov. Plus précisément, c'est un modèle mathématique à valeur dans un ensemble dénombrable, les états, dans lequel le temps passé dans chacun des états est une variable aléatoire réelle positive, suivant une loi exponentielle.

Cet objet est utilisé pour modéliser l'évolution de certains systèmes, comme les files d'attente.

### 2.6.C. Processus Stochastique

Un processus stochastique $X=(X_t)_{t\in T}$ est une famille de variable aléatoire $X_t$ indexée par un ensemble T En général T= R ou $R_+$et on considère que le processus est indexé par le temps t. Si T est un ensemble fini, le processus est un vecteur aléatoire. Si T = N alors le processus est une suite de variables aléatoires. Plus généralement quand T ∈ Z, le processus est dit discret. Pour T inclus dans $R^d$, on parle de champ aléatoire (drap quand d = 2).

Un processus dépend de deux paramètres : Xt(ω) dépend de t (en général le temps) et de l'aléatoire ω ∈ **Ω** :
- Pour t ∈ T fixé, $\omega \in \Omega \rightarrow X_t(\omega)$ est une variable aléatoire sur l'espace de probabilité (**Ω**; F; P) ;
- Pour ω ∈ T fixé , $t \in \Omega \rightarrow X_t(\omega)$ est une fonction a valeurs réelles, appelée trajectoire du processus. C'est un enjeu que de savoir si un processus admet des trajectoires mesurables, continues, dérivables ou encore plus régulières.
Dans la suite, sauf mention contraire, on prendra T = R+ ou [0; 1]. [55]

## 2.7. Conclusion

Après tout ce dont nous avons parlé, la théorie des files d'attente est vraiment importante dans notre vie quotidienne. Nous avons donc illustré tous ses aspects et caractéristiques et toutes ses mesures importantes et certaines théories de probabilité telles que la loi exponentielle et les variables aléatoires.
Dans le prochain chapitre, on passe à présenter la simulation avec ses approches et caractéristiques.

# 3. *La simulation des systèmes de file d'attente*

## 3.1. Introduction

La simulation informatique ou numérique désigne l'exécution d'un programme informatique sur un ordinateur ou réseau en vue de simuler un phénomène physique réel et complexe (par exemple : chute d'un corps sur un support mou, résistance d'une plateforme pétrolière à la houle, fatigue d'un matériau sous sollicitation vibratoire, usure d'un roulement à billes…). Un système à événements discrets est un système à états discrets et événementiel dans lequel les changements d'état dépendent entièrement de l'occurrence d'événements discrets au fil du temps.

Donc, on va étudier tous les aspects de la simulation et la simulation à événement discret avec tous ces approches et caractéristiques et on va entamer quelques travaux connexes liés à ce domaine.

## 3.2. Pour Quoi la Simulation ?

La modélisation et la simulation sont au cœur de notre réflexion. Lorsque la situation est trop complexe pour être analysée par la seule simulation mentale, nous utilisons un ordinateur pour simuler la situation. Considérons les situations suivantes [22] :

1. Trouver des règles de répartition optimales dans une fabrique moderne de semi-conducteurs de 300 mm
2. Évaluer des conceptions alternatives pour les hôpitaux, les bureaux de poste, les centres d'appels, etc.
3. Planification d'un réseau sans fil pour une entreprise de télécommunications
4. Concevoir ou moderniser le système de circulation urbaine d'une grande ville
5. Évaluer les politiques anti-pollution pour contrôler les pollutions des systèmes fluviaux

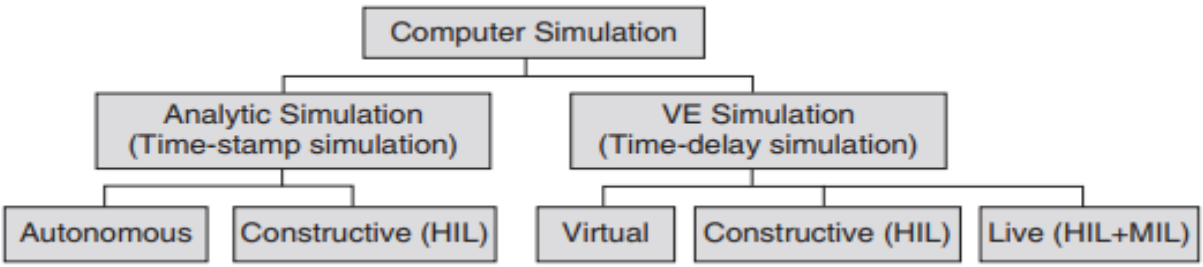



*Figure 3.1: Classification des Simulations ordinateur*

Pour les situations réelles ci-dessus, la simulation peut être le seul moyen de résoudre les problèmes. En pratique, la simulation peut être nécessaire car il n'est pas possible d'expérimenter avec le système réel ; votre budget ne vous permet pas d'acquérir un prototype coûteux ; un vrai test est risqué ; votre client le veut « hier » ; votre équipe souhaite tester plusieurs solutions et les comparer ; vous aimeriez garder un moyen de reproduire ses performances plus tard.

## 3.3. Domaines d’application

Les applications de la simulation sont vastes. Les présentations récentes à la Winter Simulation Conférence (WSC) peuvent être divisées en systèmes de fabrication, systèmes publics et systèmes de service. WSC est un excellent moyen d'en savoir plus sur les dernières applications de simulation et la théorie. Il existe également de nombreux tutoriels aux niveaux débutant et avancé.[28]

### 3.3.A. Applications des fabrications et de manutention

- Minimiser les délais de synchronisation des pièces préfabriquées avant assemblage

- Evaluation des stratégies de routage AGV

- Modélisation et analyse flexibles de systèmes AS/RS-AGV à grande échelle

- Conception et analyse de systèmes de manutention à grande échelle

- Analyse des flux de matières des usines d'assemblage automobile

- Analyse des effets des niveaux d'en-cours sur la satisfaction client

- Évaluation du coût de la qualité

### 3.3.B. Applications de systèmes publics

- ***Systèmes de santé***

  Dépistage des anévrismes de l'aorte abdominale

  Développement de lymphocyte chez les patients immunodéprimés

  Dynamique de l'asthme et amélioration médicale

- ***Systèmes militaires***

  Utilisation de l'équipement de soutien de l'armée de l'air

  Analyse des équipements de manutention pour le pré positionnement des navires

- ***Ressources naturelles***

  Analyse de la pollution diffuse

- ***Services publics***

  Analyse du système d'ambulance d'urgence

## 3.4. Les étapes d'une étude de simulation

Ce sont quelques-uns des étapes de simulation et pour une description plus détaillée, voir [53]

1- La simulation doit être conduite tout au long du cycle de vie d'une étude de simulation
2- Le résultat du modèle de simulation ne doit pas être considéré comme une variable binaire lorsque le modèle est absolument correct ou absolument incorrect
3- Un modèle de simulation est construit par rapport aux objectifs de l'étude et sa crédibilité est jugée par rapport à ces objectifs
4- Le modèle de simulation nécessite une indépendance pour éviter les biais du développeur
5- Le modèle de simulation est difficile et demande de la créativité et de la perspicacité
6- La crédibilité du modèle de simulation ne peut être revendiquée que pour les conditions prescrites pour lesquelles le modèle est testé
7- Le test complet du modèle de simulation n'est pas possible
8- Le model simulation doit être planifié et documenté
9- Prévenir les erreurs
10- Les erreurs doivent être détectées le plus tôt possible dans le cycle de vie d'une étude de simulation
11- Problème de réponse multiple doit être reconnu et résolu correctement
12- Tester avec succès chaque sous-modèle n'implique pas la crédibilité globale du modèle
13- Double problème de validation doit être reconnu et résolu correctement
14- La validité du modèle de simulation ne garantit pas la crédibilité et l'acceptabilité des résultats de simulation
15- La précision du problème formulé affecte grandement l'acceptabilité et la lisibilité des résultats de simulation

### 3.4.A. Les objectifs de simulation

1) Présenter une abstraction simplifiée des éléments essentiels d'une situation ;
2) Rendre explicites les relations essentielles et les interactions fondamentales dans une situation ;
3) Faire avancer la variable temporelle à une vitesse accélérée afin que les implications résultant d'une action entreprise dans une situation dynamique puissent être clairement ressenties ;
4) Placer le participant dans une situation de pression, afin qu'il ressente l'impact direct de la prise de décision ;
5) Offrir une opportunité de participer au processus d'enseignement-apprentissage basé sur une approche d'auto-apprentissage.[54]

### 3.4.B. Les types de simulation informatique

- **La simulation par événements discrets**

Elle est présentée dans le prochaine titre (3.5).

- **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. Soit Q(t) et X(t) les vecteurs d'état du système et de trajectoire d'entrée, respectivement. Ensuite, une simulation linéaire continue est une évaluation numérique de la fonction de transition d'état linéaire $dQ(t)/dt = AQ(t) + BX(t)$, où A et B sont des matrices de coefficients. [22]

- **La simulation de Monte Carlo**

Il s'agit d'une classe d'algorithmes de calcul qui reposent sur un échantillonnage aléatoire répété pour calculer l'intégration numérique de fonctions issues de l'ingénierie et de la science qui sont impossibles à évaluer avec des méthodes analytiques directes. Ces dernières années, la simulation Monte Carlo a également été utilisée comme technique pour comprendre l'impact du risque et de l'incertitude dans les modèles financiers, de gestion de projet et autres modèles de prévision.[22]

## 3.5. La simulation a événement discret

La simulation d'événements discrets (DES) est une technique permettant d'évaluer des systèmes représentés sous la forme d'une séquence d'événements.
Dans la modélisation basée sur les états, la dynamique du système est décrite par une fonction de transition d'état interne ($\delta int : Q \rightarrow Q$) et une fonction de transition d'état externe ($\delta ext : Q \times X \rightarrow Q$), où Q est un ensemble d'états du système. Et X est un ensemble d'événements d'entrée.
Ainsi, la simulation à événements discrets peut être considérée comme une évaluation informatique des fonctions de transition.[22]

- **Système :** Une collection d'objets qui interagissent dans le temps. Il existe deux types de systèmes :
  **Discret :** à des moments séparés de notre temps de simulation, les valeurs sont représentées par des variables et cela change encore et encore instantanément
  **Continu :** les variables changent au fur et à mesure de son état de manière continue. [30]
- **Horloge :** la valeur actuelle du temps dans notre modèle de simulation.
- **Un événement :** changement d'état d'un système.
- **List d'événement :** un enregistrement décrivant quand un événement doit être exécuté.
- **Activité :** une paire d'événements, l'un commence et l'autre termine une opération, transformant l'état d'une entité.
- **Processus :** Un ensemble d'événements classés dans le temps. [31]
- **Modèle :** pour décrire la relation entre nos objets dans n'importe quel système, nous l'avons représenté mathématiquement.

- **Entité :** n'importe lequel de nos objets dans n'importe quel système doit toujours être représenté dans le modèle de ce système.
- **Attribut :** descripteur d'une entité.
- **Liste chaînée :** ensemble d'enregistrements enchaînés dans un ordre spécifié selon les valeurs assumées par l'attribut de chaque entité.

## 3.6. Les approches de la simulation a événement discret

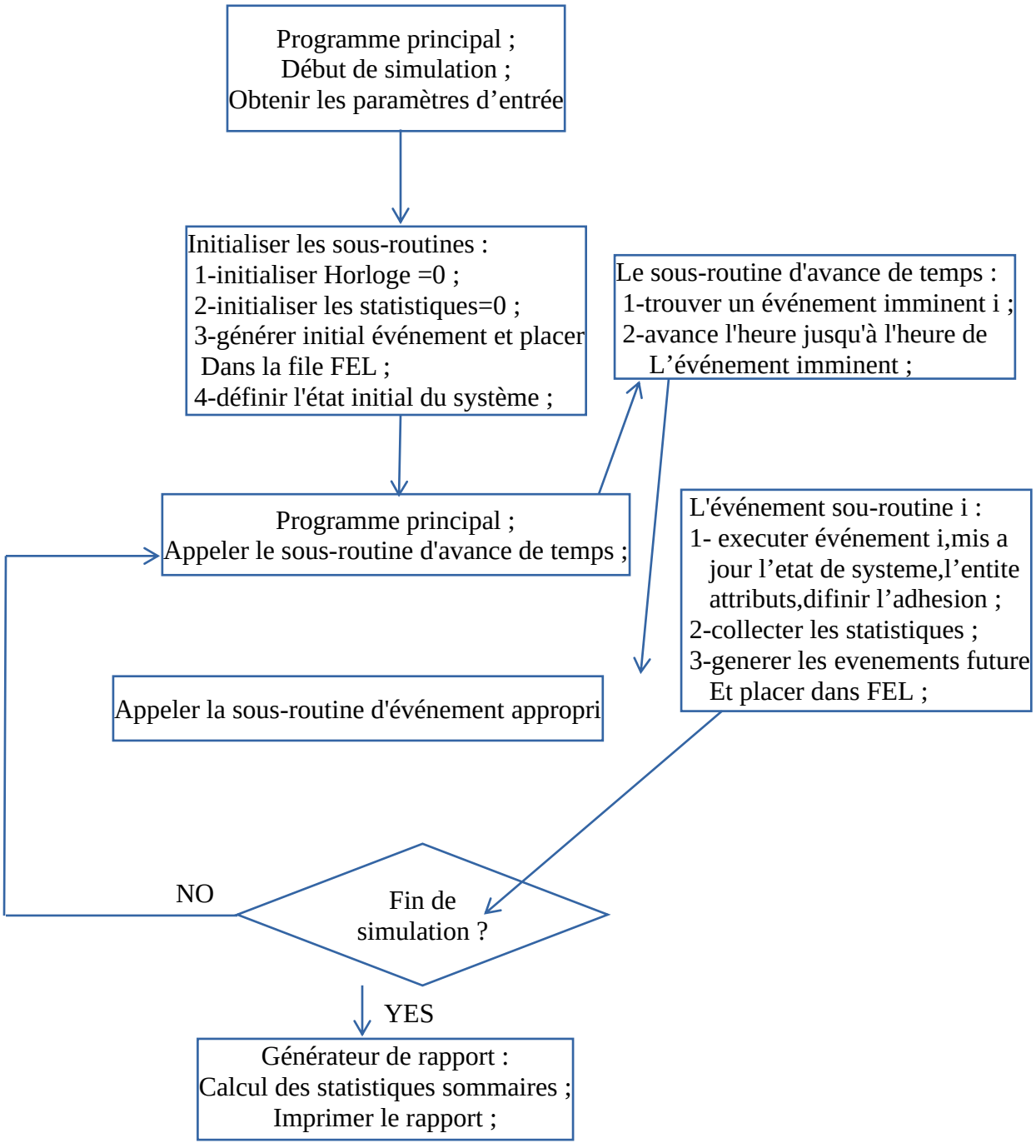
### 3.6.A. L'approche de planification d'événements *(Event Scheduling)*

Cette approche se concentre sur les événements et leur impact sur l'état du système. Le processus d'avance dans le temps est effectué sur la base d'une liste ordonnée appelée liste d'événements futurs (FEL). Lorsqu'un nouvel événement est généré dans le système, il est inséré dans l'ordre chronologique dans la liste puis l'horloge passe à l'heure suivante correspondant au premier élément du FEL, traite l'événement, modifie l'état du système et introduit tous les changements nécessaires dans le FEL. Il faut noter que l'opération d'insertion peut devenir coûteuse s'il y a trop d'événements futurs qui sont générés et doivent être insérés dans la liste. [28]

Les étapes de cette approche peuvent être bien résumé comme ça :[34]

1- Initialiser le modèle.
2- Si la période de simulation a expiré ou si d'autres conditions d'arrêt sont remplies, terminer la simulation.
3- Avancez l'horloge de simulation jusqu'à une valeur fournie par l'indicateur de temps du premier événement prévu pour se produire (parce que les événements sont classés par ordre chronologique, cette heure d'événement est lue à partir du premier avis d'événement conservé sur la liste d'événements).
4- Invoquer une routine d'événement correspondant au premier événement programmé pour se produire (la procédure à appeler est également indiquée par une entrée respective du premier avis d'événement).
5- Supprimez le premier avis d'événement de la liste d'événements et passez à l'étape 2

Néanmoins, cette approche est définitivement plus rapide que l'approche analyse d'activité, car elle évite la vérification inutile des activités à chaque avance d'horloge.[28]




Un organigramme de l'approche de planification d'événements *(Event Scheduling)*

### 3.6.B. L'approche d'analyse d'activité *(Activity Scanning)*

Cette approche se concentre sur les activités et les conditions qui permettent à une activité de démarrer. Dans cette approche, le temps est décomposé en petits incréments ; à chaque instant (ou avance d'horloge), le modèle vérifie les conditions pour chaque activité, et celles qui satisfont aux conditions sont alors lancées. On peut noter que cette approche peut devenir extrêmement lente si l'incrément de temps est trop petit. Malheureusement, de grands incréments de temps ne fourniront pas une représentation correcte de la façon dont les activités

doivent être exécutées, car certaines activités qui satisfont aux conditions devront attendre la prochaine avance d'horloge pour démarrer. Il convient également de noter que si les incréments de temps sont trop petits, une grande partie des contrôles d'activité seront inutiles car il n'y aura aucun changement dans le système. Par exemple, si le temps est subdivisé en incréments de 0,01 s et qu'il existe une tâche qui pourrait commencer à t = 10, cela implique 1000 vérifications des conditions de démarrage pour toutes les activités qui doivent être prises en compte.

Ce n'est peut-être pas si problématique, compte tenu de la puissance de calcul actuelle, mais on peut voir que cela pourrait facilement devenir incontrôlable s'il y avait trop d'activités ou si les incréments de temps étaient beaucoup plus petits. Ce type d'approche pourrait conduire à des simulations pouvant prendre plusieurs heures, voire plusieurs jours.[28]

### 3.6.C. L'approche de Processus-Interaction *(Process-Interaction)*

Cette approche se concentre sur les processus plutôt que sur les activités. Un processus est un ensemble d'événements, de retards et d'activités chronologiques qui définissent le cycle de vie d'une entité à mesure qu'elle se déplace dans le système. Avec cette approche, il est possible d'avoir de nombreux processus actifs à un moment donné et l'interaction entre les processus pourrait devenir assez complexe. Cette approche produit un code plus modulaire et est considérée comme meilleure du point de vue de la programmation. [28]

## 3.7. Méthodes de simulation en régime permanent

Lors d'une simulation il existe des méthodes utilisées pour collecter des données d'observation en régime permanent (*Steady-State Simulation Methods*) tels-que :

### 3.7.A. La méthode des sous-intervalles (*Subinterval Method*) :

La méthode sous-intervalles (*Batch Means*) est l'une des techniques d'intervalle de confiance les plus populaires pour l'analyse de sortie d'une simulation en régime permanent. [35].

Soit la durée de la simulation M, et la variable observée X. De début de la simulation, la phase transitoire de K observations sera rejetée. La durée de régime permanent (M-K) est divisée à N intervalles, dans chaque intervalle il y a $n = \frac{M-K}{N}$ observations.

Un paramètre important dans la méthode Batch Means est le choix de la taille du Batch (lot). Le choix de la taille du lot a un impact direct sur la qualité de l'estimateur de la variance et de l'intervalle de confiance (*Schmeiser (1982) et Politis et Domano (1992)*). [35]

L'avantage de la méthode de sous-intervalle est que l'effet des conditions transitoires est atténué.[36]

L'inconvénient de la méthode est que les sous-intervalles ou (lots) successifs sont nécessairement corrélés. L'effet de corrélation peut être atténué en augmentant la base de temps de chaque lot.[36]

La figure 3.2 ci-dessous [36] représente le principe de la méthode de Batch Means :

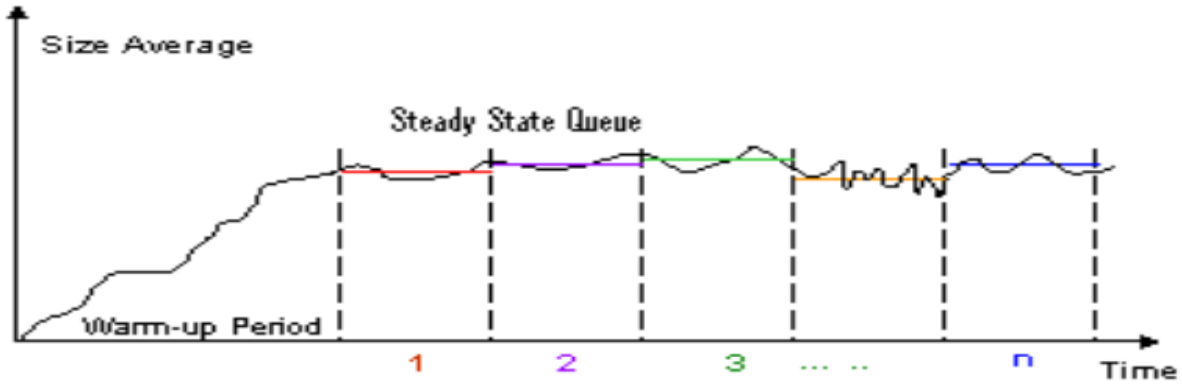



*Figure 3.2: La méthode des sous-intervalles (Batch Means Method)*

### 3.7.B. La méthode régénérative (*Regenerative Method*) :

La méthode régénérative pour estimer les paramètres en régime permanent est l'une des méthodes de base de l'analyse des sorties de simulation. Cette méthode dépend de théorèmes centraux limites pour les processus régénératifs et d'estimations faiblement cohérentes pour les constantes de variance apparaissant dans les théorèmes centraux limites.[37]
L'objectif de Regenerative Method est d'obtenir des intervalles de confiance asymptotiquement valides pour a lorsque la longueur de la simulation devient grande.
Cette méthode réduit l'effet d'autocorrélation, elle a l'inconvénient de donner un plus petit nombre de lots pour une longueur de série donnée.[36]

La figure 3.3 ci-dessous [36] représente le principe de la méthode régénérative :

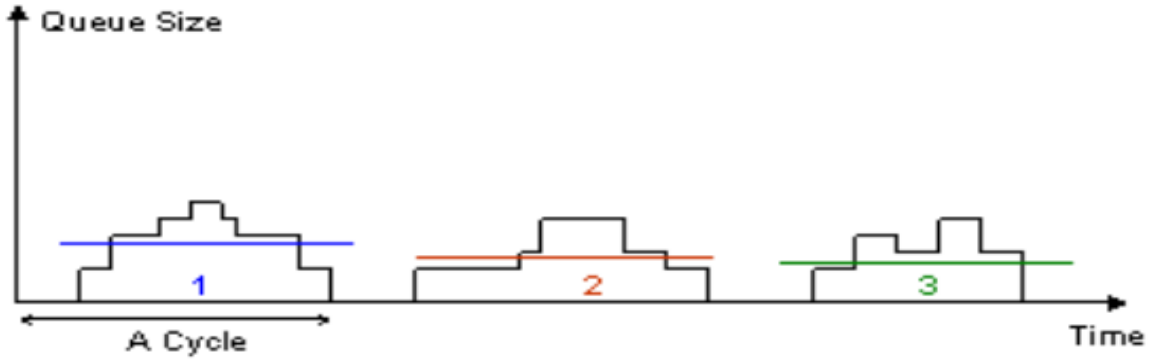



*Figure 3.3: La méthode régénérative (Regenerative Method)*

### 3.7.C. La méthode de réplication indépendante :*(Independent Replication Method)*

Avec la méthode de réplication indépendante, r exécutions indépendantes sont effectuées, qui sont utilisées pour calculer r moyennes d'échantillons répliqués, chacune basée sur, disons, m observations. Ensuite, $\sigma^2$ est estimé par m fois la variance d'échantillon des moyennes répliquées.[38]
L'avantage de la méthode est qu'elle donne des observations vraiment indépendantes. L'inconvénient est que chaque observation peut être biaisée par l'effet initial des conditions transitoires. Ce problème peut être atténué en rendant la longueur de course suffisamment grande.[36]
La figure 3.4 ci-dessous [36] représente le principe de la méthode de réplication indépendante :

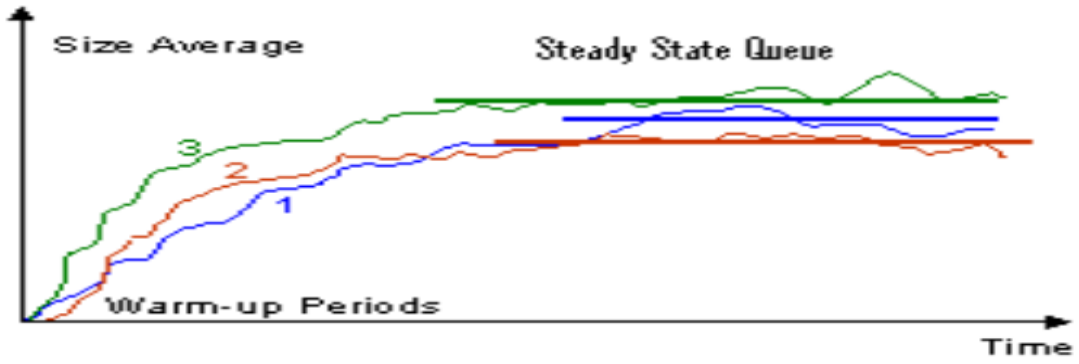



*Figure 3.4: La méthode de réplication indépendante*

## 3.8. Travaux connexes

L'économie d'énergie est le plus important problème dans les réseaux de capteurs sans fil (RCSF) pour la majorité des nœuds de capteurs équipés de batteries non rechargeables.
Donc l'application du modèle de file d'attente avec vacances pour réduire la consommation d'énergie des réseaux de capteur sans fil a fait l'objet de plusieurs travaux.
Dans les prochaines sections, Nous allons présenter des travaux dans ce domaine pour l'évaluation et l'analyse des performances des systèmes de fichier d'attente pour objectif de réduire la consommation d'énergie dans le nœud de capteurs.

### 3.8.A. Analyse des systèmes théoriques des files d'attente

**B. Boutoumi et N. Gharbi [4] :**

Dans ce modelé les chercheurs ont proposé une nouvelle approche pour bien minimiser l'Energie consomme et le délai d'attente dans un nœud de capteur, avec l'utilisation des réseaux de pétri stochastique (GSPN).
Ils ont prendre en considération la N-Policy, cette politique travail comme un schéma sommeil/réveil dans les nœuds de capteurs, si le nombre des clients dans la file d'attente (ou

buffer) atteint une certain seuil N le serveur prendre une vacance, selon leur résolution ils ont dit que la politique N est efficace pour la réduction de l'énergie consomme.
Ensuite, ils ont proposé une politique de sommeil/réveil des nœuds dans un RCSF nomme la politique Hybride, dans cette politique le serveur prendre des vacances aléatoires.
Le serveur revient de vacances dans deux cas :
1- A l'instant de l'arrivée du N-ième paquet.
2-Sinon à la fin d'une période de vacances qui est une durée aléatoire à distribution exponentielle de paramètre, malgré si le nombre des paquets dans le buffer est inférieur à la seuil N.
L'hybride est efficace dans les RCSFs ou le taux d'arriver est variable. Ceci permet d'éviter la rapidité dans la saturation du buffer et le gaspillage d'énergie à cause a la retransmission des paquets perdus dans le cas où le taux d'arriver est élevé. De plus dans le cas où le taux des arrives est petit, la vacance aléatoire permet de réduire le délai d'attentes des paquets.
A la fin, ils donnent les formules pour les mesures de performance du RCSFs et une analyse bien explique pour montrer l'effet des deux politiques de vacances sur le réseau

**Bachira Boutoumi et Nawel Gharbi [6] :**

Dans ce modèle, un capteur en sommeil peut devenir semi-busy si le nombre de paquets en attente atteint le seuil N2 et commence le service avec le taux de vacance de travail $\mu 2$.
Dès que le nombre de paquets en attente atteint le seuil N1 ($N2<N1$) le nœud passe à l'état occupé et commence le service avec le taux normal $\mu 1$ ($\mu 2 < \mu 1$)
De plus, ils ont proposé la modélisation et l'analyse des performances de la nouvelle file d'attente en utilisant le formalisme stochastique généralisé des réseaux de Petri (GSPN). Pour cela, ils ont développé les formules pour des indices de performance dans l'état de system est stationnaire et de consommation énergétique.
Ils ont proposé une technique d'économie d'énergie et d'efficacité de latence pour les nœuds de capteurs sans fil en duplex intégral, basée sur une combinaison de politique de vacances normales et de vacances de travail, qu'ils ont appelé la politique de vacances de travail à deux seuils.
Enfin, ils ont donné une analyse détaillée qui prouve l'efficacité de l'approche proposée.

**Sanjeev Ghosh and Srija Unnikrishnan [3]:**

Ont proposé un modèle pour optimiser la consommation d'énergie, base sur la file d'attente M/M/1 et N-Policy.
Le nœud de capteur allume la transmission de radio à l'arrivée d'un paquet et éteint à des périodes de fin de service, Ces auteurs ont fait une supposition que les paquets arrivent un nœud suivant avec le taux d'arrivée moyen $\lambda$ qui est considéré être un processus de Poisson.Les temps de service radio suivent une distribution exponentielle et a signifié $1/\mu$.
Un schéma est étudié dans qui un capteur le nœud passe à l'état occupé, c'est-à-dire qu'il démarre son émetteur radio lorsque le tampon du nœud est rempli au moins de nombre de paquets égal au seuil(N) et le nœud bascule à l'état inactif quand tampon est vide. Ces commutations de nœuds entre l'état inactif et l'état occupé sont appelées transitions.

L'objectif principal est de réduire la consommation d'énergie des nœuds de capteurs individuels en diminuant le nombre de transitions, les auteurs étudient le comportement d'un nœud capteur unique lorsque ce schéma est implémenté.

**P. Jayarajan, R. Maheswar, G.R. Kanagachidambaresan, V. Sivasankaran, M. Balaji and Jagannath Das [33]:**

Ont aussi proposent un modèle pour réduire la consommation d'énergie dans les RCSFs en utilisant les files d'attente M/D/1 N-Policy avec priorité.

Les réseaux de capteurs sont classés en capteurs homogènes et réseaux de capteurs hétérogènes (HSN). Dans réseaux homogènes, tous les nœuds seront du même type avec référence au matériel et à l'énergie. Le principal problème de l'utilisation du même type de capteur est que la rotation du rôle en tant que CH et les capacités matérielles requises et également faire agir le nœud en tant que CH. L'autre classification, les HSN comprend deux différents types de nœuds capteurs. Dans la plupart des articles de recherche, il est déduit que les HSN peuvent améliorer la durée de vie du réseau de manière significative et fournit également de meilleures performances de réseau dans termes d'énergie.

Dans chaque cluster, le capteur H agit en tant que CH et les capteurs L agissent en tant que membre du cluster (CM). Ici, un modèle analytique d'un HSN orienté cluster est développé en utilisant un seuil de file d'attente basé sur le modèle de file d'attente prioritaire M/D/1 en considérant la défaillance des nœuds. La performance de la technique proposée au moyen de la consommation d'énergie et du délai est analysée en tenant compte des défaillances des nœuds.

Dans HSN, un seul cluster est considéré. Dans ce cluster, en fonction de la criticité du paquet, le paquet est classé en haute priorité (HP) ou faible Paquet prioritaire (LP). Aucune condition de seuil (N=1) n'est considérée pour les paquets HP, car les paquets HP doivent être entretenus immédiatement mais le NPM est pris en compte pour les paquets LP car il peut gérer un certain retard. Le nœud de capteur transitera depuis IDLE à l'état BUSY immédiatement s'il reçoit un paquet HP et qu'il attendra que N paquets passent de l'état IDLE à BUSY et le transfert de données réel à lieu pendant l'état BUSY. Si seulement le défaut se produit pendant l'état BUSY, puis la transmission est arrêtée et le défaut au niveau du nœud est détecté. Une fois après le nœud défectueux récupère, la transmission du paquet se poursuit dans le État OCCUPÉ.

### 3.8.B. Des recherches applique dans le domaine des files d'attentes avec la simulation à événement discret

**Mohammad Ehsanifar, Nima Hamtas et Mahshid Hemesy** [40] :

Ont proposé un modèle de simulation pour fournir des mesures de performances floues pour les files d'attentes avec un taux d'arrivé et taux de service floués et transforme une fille d'attente floue à une fille d'attente traditionnelle.

Ils ont tenté de développer une distribution plus pratique des modèles de files d'attente au moyen de la théorie des ensembles flous, lorsque les temps d'arrivée et les temps de service sont des variables floues.

Le but de cette étude est de prédire les temps d'attente de chaque client dans un modèle de file d'attente M/M/C dont les arrivées suivent un processus de poisson et les temps de service sont distribués de manière exponentielle. Ce modèle qui a de nombre de serveurs C, ses temps inter-

arrivées et ses temps de service sont des variables floues aléatoires. Ils ont choisi d'utiliser la méthode de simulation pour modéliser des systèmes complexes et bien comprendre les caractéristiques des files d'attente. Pour illustrer comment le modèle est exprimé pour analyser la file d'attente floue, ils ont considéré une situation réelle dans une banque dans laquelle il y a cinq guichets automatiques avec le même type de service.
Enfin, Pour réduire une file d'attente floue à une famille de files d'attente nettes, l'approche α-cut a été appliquée et afin de confirmer la validité de notre modèle, ils ont utilisé des nombres flous triangulaires.

**Nekaa Hamza et Kias Mohamed [52] :**

Ils ont proposé un modèle comparatif entre deux politiques de vacances (la politique N et la politique Hybride) de but d'avoir l'influence de chaque politique et d'étudier son effet dans les RCSFs.
Ils ont utilisé la simulation a évènements discrets pour développe un outil qui nous montre des graphes en fonction de variation des certaines variables comme le taux d'arrivé et le taux de service et la seuil N.
Beaucoup des mesures de performance ont été abordé dans ce travail tel que la consommation d'énergie et le délai d'attente et le délai de réponse.

## 3.9. Conclusion

D'ici, dans de ce chapitre, se voit l'importance de la simulation dans la modélisation de systèmes tels que les systèmes de théorie des files d'attente, nous avons présenté ses caractéristiques et ses types et nous avons également vu des travaux connexes qui nous ont données des idées et certains résultats et travaux dans ce domaine

Dans le chapitre suivant, et après avoir présenté tous les aspects théoriques concernant notre travail, nous pourrons concevoir notre application et notre modèle.

# *4. Conception de l'outil de simulation des nœuds de capteurs sans fils*

# 4.1. Introduction

Une conception de système consiste à créer une représentation de ce système de manière virtuelle, afin que nous puissions mieux le comprendre et analyser tous ses aspects.

Donc, dans ce chapitre, nous sommes sur le point de construire notre conception de système avec des aspects à la fois prioritaires et non prioritaires, avec et sans préemption et avec l'utilisation de la méthode de politique N (N-vacance), et à la fin nous montrerons quelques mesures de performance pour nous aider à évaluer le système.

# 4.2. Les modélisations mathématiques d'un nœud de capteurs

Pour construire un système, nous devons le concevoir, un modèle de file d'attente avec vacance et avec trois classes de priorité pour les arrivées fait partie des modèles que nous utiliserons pour représenter le comportement d'un nœud de capteur sans fil.

Et bien sûr, nous devrons décrire la politique N-vacances que nous allons utiliser, et toutes ses caractéristiques.

Un seul nœud de capteur sans fil effectue une grande partie des changements de mode vers un autre mode ainsi que les divers traitements de mémoire de l'unité de traitement

Et on sait maintenant que l'unité émettrice est la plus énergivore dans un nœud capteur sans fil et quand on limite son utilisation en le mettant dans un état inactif, on permet des gains de durée de vie très considérable pour la batterie du capteur.

## 4.2.A. Modèle ordinaire sans vacances

Le modèle ordinaire c'est le modèle qui ne contient pas des vacances pour les serveurs, et tous les paquets arrivants avec la même priorité et sont servis par la politique FIFO.

La figure 4.1 ci-dessous représente l'architecture du modèle ordinaire :

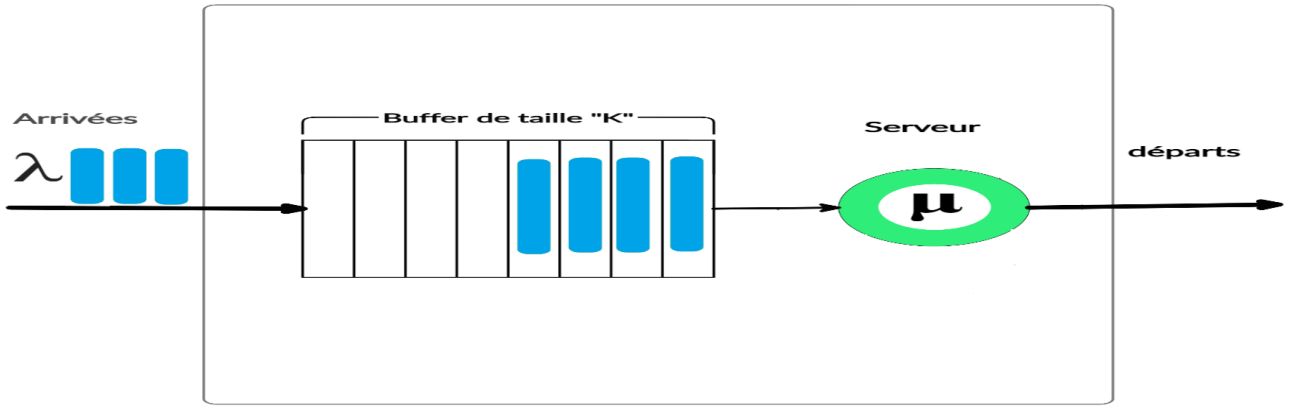



*Figure 4.1: Structure de base d'un nœud de capteur avec le modèle ordinaire*

### 4.2.B. Modèle N-vacances avec une seul classe de priorité

Dans un nœud de capteur sans fil nous avons deux modes principaux qui alternent entre eux :

- Mode libre : le nœud ne peut recevoir que les paquets
- Mode occupé : le nœud écoute le canal, génère des paquets et reçoit et transmet également les données.

Pendant la période d'occupation, le capteur transite entre l'état libre (inactif) où il peut générer et recevoir des paquets et l'état occupé (actif) où il peut générer, recevoir et transmettre.
On constate que l'énergie perdue pour la transition entre l'état libre et l'état occupé engendre une surconsommation d'énergie appelée énergie de commutation.
La politique de vacances N est utilisée comme un schéma de réveil de file d'attente efficace pour réduire la consommation d'énergie dans un nœud de capteur. A cet effet, un capteur qui est à l'état inactif passe à l'état occupé si et seulement si le nombre de paquets dans la file d'attente atteint un certain seuil N.
Cette politique permet de minimiser la commutation énergétique induite par la fréquence de transition entre l'état inactif et occupé.[4]

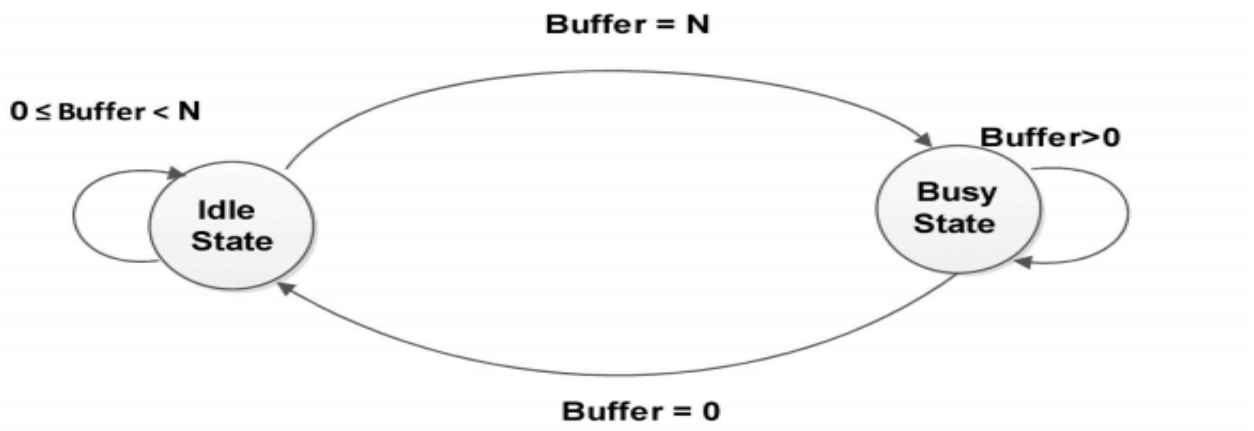



*Figure 4.2: Diagramme d'état transition d'un nœud de capteur avec la politique N-vacances [4]*

La figure 4.2 présente le diagramme d'état transition du nœud de capteur avec N-vacances, où un capteur qui est à l'état inactif passe à l'état occupé dès que le nombre de paquets dans la mémoire (*Buffer*) atteint le seuil N. Cette politique est connue comme une technique efficace pour optimiser la consommation d'énergie dans les nœuds de capteurs sans fil pendant la période active en minimisant l'énergie de commutation provoquée par la fréquence de transition entre les états inactif et occupé.[4]

Nous décrivons le système correspondant à un nœud de capteur par un modèle de file d'attente à un serveur avec vacances et un buffer selon la notation A/B/1/K où :

A : Type de processus d'arrivé pouvant être (**M** : distribué de manière exponentielle, **D** : temps déterministe entre les arrivées ou le service, **G** : entre les arrivées aléatoires ou le temps de service avec toute distribution générale).

B : Type de processus de service (**M**, **D** ou **G**).

K : Taille de buffer pouvant être finie ou infinie.

Le serveur possède deux états de fonctionnement : actif (décrite en vert) et inactif (décrite en rouge), qui présente l'état de service et de vacances de serveur. Le passage de l'état inactif à l'état actif se fait selon la politique N-vacances.

La figure 4.3 ci-dessous représente l'architecture du modèle proposé pour le capteur dans le cas sans prioritaire :

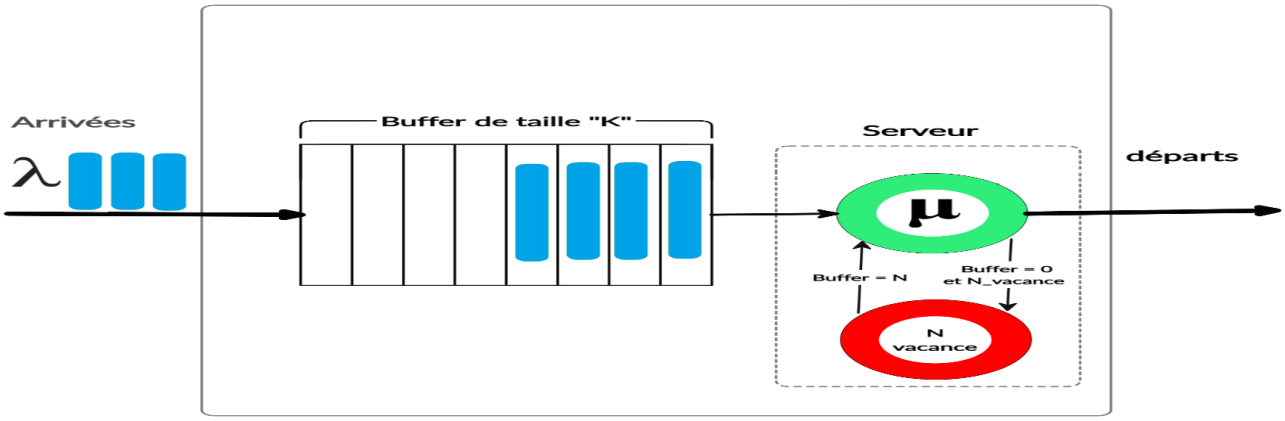



*Figure 4.3: Structure de base d'un nœud de capteur avec un seul classe de priorité [52]*

Le serveur est dans un état inactif et les paquets entrants sont stockés. La stratégie N-vacances est considérée comme un modèle de file d'attente de réveil pour que le serveur passe d'inactif à actif.

Une fois que le nombre de paquets en attente dans le tampon atteint "N", le serveur passe à l'état actif. Le nœud peut recevoir, trier et transmettre les paquets en attente selon leurs ordres FIFO jusqu'à ce que le buffer soit vide (service exhaustif). A ce moment, le serveur passe dans un état inactif.

La figure 4.4 ci-dessous représente le flux d'exécution lors de l'arrivé d'un paquet

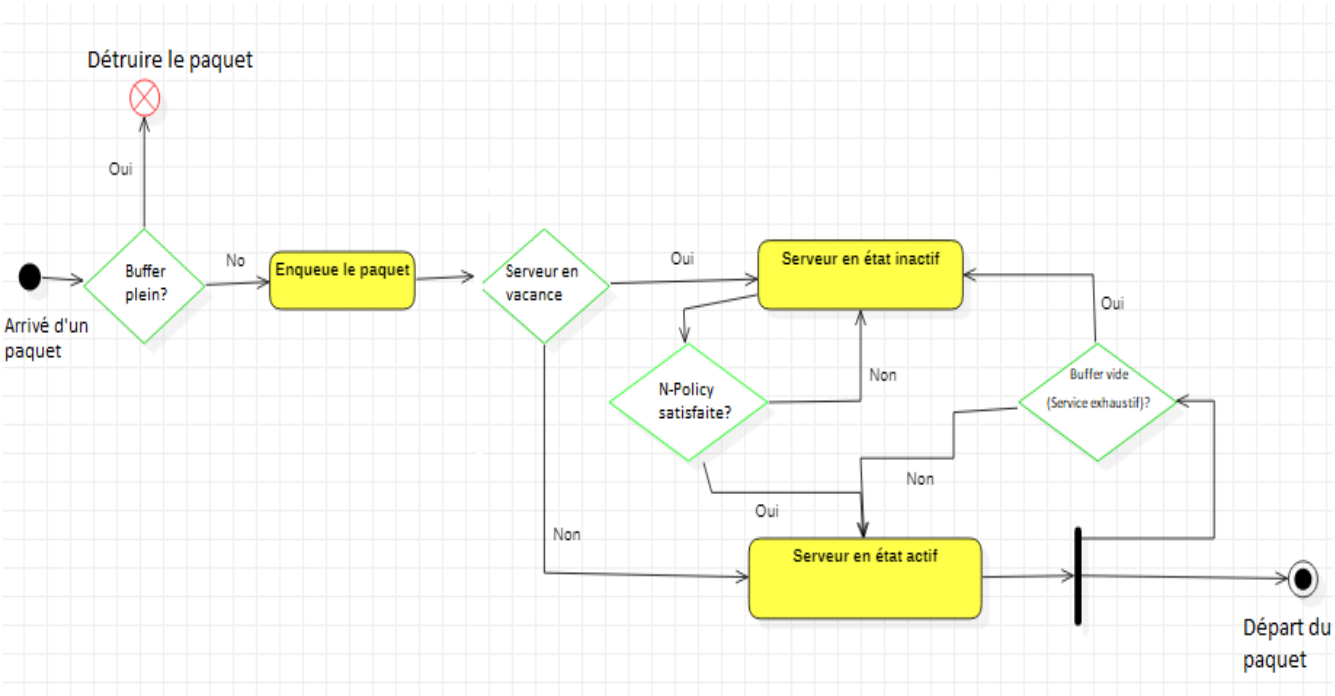



*Figure 4.4: Diagramme d'activité représente le modèle avec un seul classe de priorité*

### 4.2.C. Modèle N-vacances avec trois classes de priorité

Dans ce cas la politique N est appliqué exactement comme le modèle d'une seule priorité, c'est juste le serveur débute son service par les paquets avec la priorité plus élevée.

Nous décrivons le système correspondant à un nœud de capteur par un modèle de file d'attente à un serveur avec vacances et un buffer selon la notation A/B/1/K où :

A : Type de processus d'arrivé pouvant être (**M** : distribué de manière exponentielle, **D** : temps déterministe entre les arrivées ou le service, **G** : entre les arrivées aléatoires ou le temps de service avec toute distribution générale).
B : Type de processus de service (**M**, **D** ou **G**).
K : Taille de buffer pouvant être finie ou infinie.
Les paquets arrivent selon un des trois processus d'arrivé et ont trois types de priorité :

- Faible (Low) : décrite en bleu, qui a l'ordre le plus faible de priorité.
- Élevé (High) : décrite en vert.
- Très élevé (Very high) : décrite en rouge, qui a l'ordre le plus élevé de priorité.

Le serveur possède deux états de fonctionnement : actif et inactif qui présente l'état de service et de vacances de serveur. Le passage de l'état inactif à l'état actif se fait selon la politique N-vacances, avec deux politiques d'ordonnancement (non préemptif et préemptif).

Dans la politique préemptive : nous supposant qu'à chaque fois un paquet de priorité HIGH, il n'attente pas dans la file d'attente et il est servis immédiatement malgré s'il y'a un paquet en servis, le paquet en servis de moins priorité sera détruit.
La figure 4.5 ci-dessous représente l'architecture du modèle proposé pour le capteur :

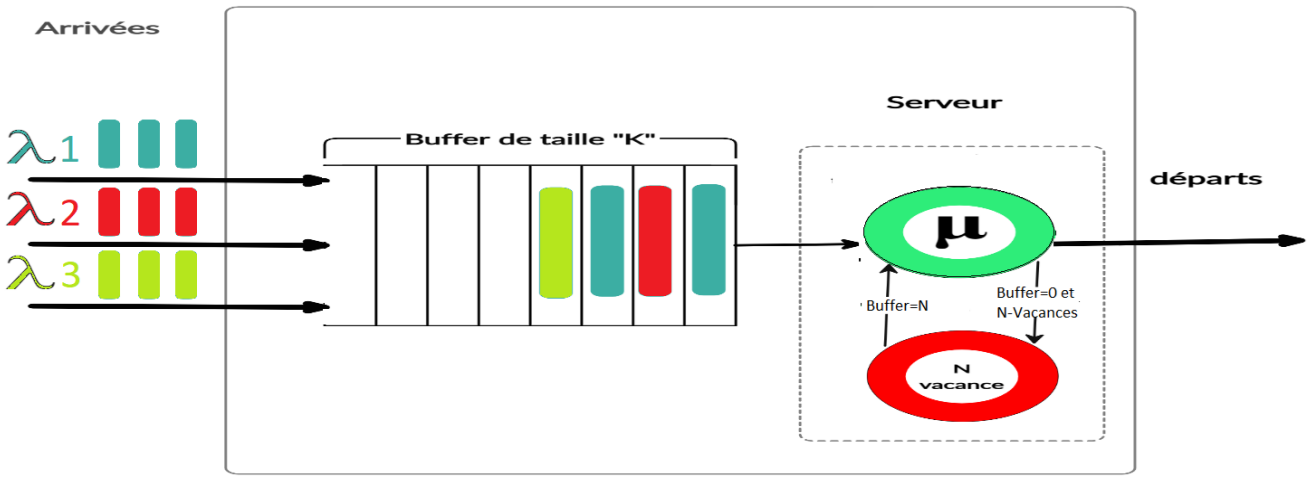



*Figure 4.5: Structure de base d'un nœud de capteur avec trois classes de priorité*

Initialement, le serveur est à l'état inactif, et les paquets arrivants sont stockés et triés selon leurs ordres de priorité dans le buffer (chaque classe de priorité est arriver a son taux d'arrivé donc le taux d'arriver totale et la somme des taux d'arriver des trois classes de priorité). La politique N-vacances est considérée comme un schéma de réveil en file d'attente pour le passage du serveur de l'état inactif à l'état actif.
Une fois le nombre des paquets en attente dans le buffer atteint « N », le serveur passe à l'état actif. Le nœud peut recevoir, trier et transmettre les paquets en attente selon leurs ordres de priorité jusqu'à le buffer soit vide (service exhaustif). A ce moment, le serveur passe à l'état inactif.
La figure 4.6 ci-dessous représente le flux d'exécution lors de l'arrivé d'un paquet

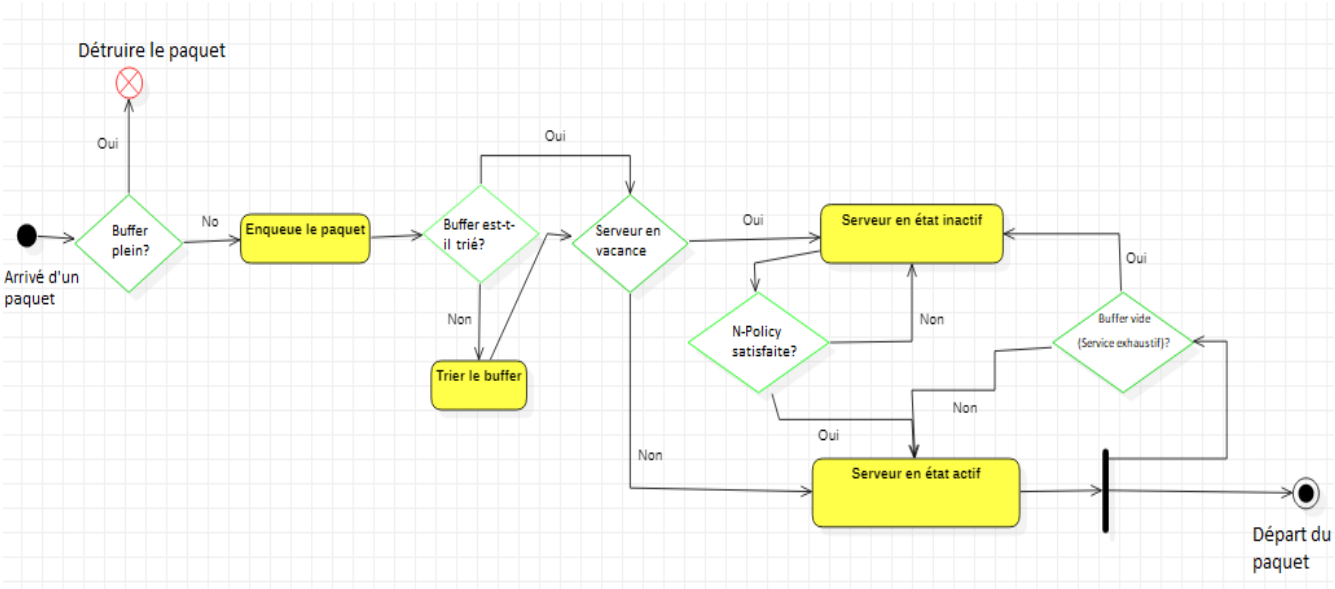



*Figure 4.6: Diagramme d'activité représente le modèle proposé avec trois classes de priorité*

## 4.3. Conception du simulateur

Nous avons choisi la simulation à événements discrets avec l'approche de planification d'événements (Event Scheduling) pour des raisons d'efficacité et de rapidité de calcul, ce système se change d'états à des points discrets, d'autre part, ce simulateur nous permet de calculer les mesures de performances de nœud de capteur qui est l'un des notre objectifs, telles que le temps de séjour et le temps d'attente...etc.
Les concepts de la simulation à événement discret que nous avons utilisé sont :

- **Le système :** le comportement d'un nœud de capteur sans fil.
- **Le modèle :** une file d'attente avec vacances A/B/1, avec différent distributions (M ou G ou D) et la capacité de buffer peut être finie ou infinie.
- **Entités :** serveur, paquets.
- **Les événements :** Arrivée, Départ, Début de service.

### 4.3.A. Variables

Les variables nécessaires utilisées et leurs descriptions sont décrites dans le tableau ci-dessous.

| Variable | Description |
|---|---|
| Horloge | Représente le temps simule. |
| Etat de serveur | Représente l'état de serveur : libre, occupe, vacances. |
| Buffer | La file qui contient les paquets arrivants |
| FEL | (Futur Event List) la liste d'événements |
| K | La capacité maximale de buffer |
| cpt | Compteur des paquets dans le buffer |
| N | La Seuil pour quitter l'état de vacance |
| λ | Le taux d'arrivée des paquets totale |
| μ | Le taux de service |
| Priorité | La classe de priorité de chaque paquet : (2 très élevé, 1 élevé, 0 faible) |
| Chose | Le type de distribution choisi |
| Sigma | Standard déviation de temps de service (juste pour la distribution 'G') |
| ECidle | La consommation d'énergie en état inactive |
| ECbusy | La consommation d'énergie en état active |
| ECswitch | La consommation d'énergie du changement de l'état |
| ECpaquet | La consommation d'énergie pour chaque paquet |

*Table 3 : Tableau des variables*

### 4.3.B. Les événements

La simulation à événements discrets avec l'approche des planifications des événements (Event Schedulling) il faut assure que les événements sont dans l'ordre, c'est pour ça, nous avons créé une liste triée selon l'heure « FEL » qui planifie les événements. Chaque élément contient l'heure à laquelle l'événement doit se produire et le type d'événement qui doit être exécuté à ce moment.

Les types des événements que nous avons utilisés dans notre simulation sont :

- **L'événement arrivé :**

  L'arrivé d'un paquet au système, il est généré pour la première fois lors de l'initialisation pour planifier le premier paquet avec sa priorité et son taux d'arrive. Les autres événements d'arrivées sont générés aléatoirement avec la distribution choisie.

- **L'événement début de service :**

  Le début de service d'un paquet dans le serveur. Il est exécuté lorsqu'une arrivée du paquet si le serveur vide ou bien lors de départ d'un paquet si le buffer n'est pas vide.

- **L'événement départ :**

  C'est la sortie du paquet du système, elle est générée quand le paquet s'éloigne du système.

### 4.3.C. Les routines

Le programme de simulation contient plusieurs routines telles que, initialisation, boucle principale et statistiques… etc. dans deux cas : le premier cas d'un buffer infini ou la capacité de buffer n'est pas important, et pour le deuxième cas d'un buffer finie ou la capacité maximale est important, 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**

  Cette routine est exécutée par l'événement « arrivée ». En générale, elle présente l'arrivé d'un nouveau paquet pour l'enfiler dans le buffer et vérifier s'il y a possibilité de rendre le serveur actif s'il ne l'est pas.

Les deux pseudo code ci-dessous montre la routine arrivée dans les deux cas : buffer infinie et buffer finie.

  - **Buffer infinie :**

**Routine Arrivée**

```
0 :  Début;
1 :  Enfiler le paquet dans le buffer avec priorité ;
2 :  cpt++; //incrémente le nombre des paquets dans le buffer
3 :  Si (Serveur en vacance)
4 :      Si(cpt ==N) //atteint le seuil
5 :          Mettre le serveur occupe;
6 :          Planifie événement « Début de service» pour le paquet: heure =Horloge
```

```
7:        FinSi ;
      FinSi ;
8:    Switch (Distribution Type):
9 :           Cas 'M':
10:             Planifie événement «Arrivée» pour le paquet: heure =Horloge+Période de Inter-Arrivée; //les temps sont générer en maniéré distribue exponentiellement
11:           Cas 'D':
12:             Planifie événement «départ» pour le paquet: heure =Horloge+Période de Inter-Arrivée;//les temps sont avec distribution déterministe
13:           Cas 'G':
14:             Planifie événement «départ» pour le paquet: heure =Horloge+Période de Inter-Arrivée;//les temps sont avec distribution général
15:           Défaut:
16:               Afficher ('not a valid entry');
17:   Fin;
```

*Figure 4.7: Pseudo Algorithme pour la routine Arrivée dans le cas buffer infinie capacité*

- **Buffer finie :**

**Routine Arrivée**

```
0 :   Début;
1 :   Si(cpt <=K) alors // buffer n'est pas plein
2 :      Enfiler le paquet dans le buffer avec priorité;
3 :      cpt ++; //incrémente le nombre des paquets dans le buffer
4 :      Si(Serveur en vacance)
5 :         Si(cpt ==N) //atteint la seuil
6 :            Mettre le serveur occupe;
7 :            Planifie événement « Début de service» pour le paquet: heure =Horloge
8 :         Fin;
9 :      Fin;
      Fin Si ;
10 :     Switch(Distribution Type):
11 :     Cas 'M':
12:      Planifie événement «Arrivée» pour le paquet: heure =Horloge+Période de Inter-Arrivée;
           //les temps sont générer en maniéré distribue exponentiellement
13:      Cas 'D':
14:      Planifie événement «départ» pour le paquet: heure =Horloge+Période de Inter-Arrivée;
```

```
        //les temps sont avec distribution déterministe

15:   Cas 'G':
16:     Planifie événement «départ» pour le paquet: heure =Horloge+Période de Inter-Arrivée;
        //les temps sont avec distribution général
17:   Défaut:
18:     Afficher ('not a valid entry');
30: Fin.
```

*Figure 4.8: Pseudo Algorithme pour la routine Arrivée dans le cas de buffer finie capacité*

- **Routine Début de service**

Cette routine est exécutée par l'événement « début de service ». Elle mettre le serveur occupé pour servir un paquet depuis le buffer et le faire planifier un événement de départ. Le pseudo code ci-dessous montre un pseudo algorithme d'une routine début de service.

**Routine Début de service**

```
0 :  Début ;
2 :  Défiler le paquet du buffer ;
3 :  Switch (Distribution Type) :
4 :  Cas 'M':
5 :     Planifie événement « départ » pour le paquet : heure =Horloge+Période de service ;
        //les temps sont générer en manière distribue exponentiellement
6 :  Cas 'D':
7 :     Planifie événement « départ » pour le paquet : heure =Horloge+Période de service ;
        //les temps sont avec distribution déterministe
8 :  Cas 'G':
        Planifie événement « départ » pour le paquet : heure =Horloge+Période de service ;
        //les temps sont avec distribution général
9 :  Défaut :
10:     Afficher('not a valid entry');
11: Fin;
```

*Figure 4.9: Pseudo Algorithme pour la routine début de service*

- **Routine Départ**

Cette routine est exécutée par l'événement « départ ». Elle met le serveur en état libre, si le buffer est vide, elle le met en vacances, sinon, elle planifier les événements nécessaires. Un pseudo algorithme est illustré ci-dessus.

**Routine Départ**

```
0 :   Début ;
2 :  Si (cpt ==0) alors
3:   Mettre le serveur en vacances ;
4:   Sinon
5:      Planifie événement « Début de service » pour le paquet : heure =Horloge
6:   cpt -- ; //décrémente le nombre de paquet
7:   Fin ;
```

*Figure 4.10: Pseudo Algorithme pour la routine de départ*

- **Routine Initialisation**
  - **Buffer infinie :**

**Routine Initialisation**

```
0 : Début ;
1 : Horloge =0 ;
2 : cpt =0; //compteur des paquets
3 : Définir le buffer vide ;
4 : Définir le serveur en vacance ;
5 : Définir la liste d'événement future FEL vide ;
6 : Récupération et initialisation les variables N, λ, µ, chose, sigma,
7:  Planifie événement d'arrivée pour le premier paquet : horloge =0;
8:  Fin ;
```

*Figure 4.11: Pseudo Algorithme pour la routine initialisation pour le cas d'un buffer infinie*

  - **Buffer finie :**

**Routine Initialisation**

```
0 : Début ;
1 : Horloge =0 ;
2 : cpt =0; //compteur des paquets
3 : Définir le buffer vide ;
4 : Définir en vacance ;
5 : Définir la liste d'événement future FEL vide ;
6 : Récupération et initialisation les variables K, N, λ, µ, chose, sigma,
7:  Planifie événement d'arrivée pour le premier paquet: horloge =0;
```

*Figure 4.12: Pseudo Algorithme pour la routine initialisation pour le cas d'un buffer finie*

- **Routine de la boucle principale**

Cette routine 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 boucle principale**

```
0 : Début ;
1 : Appeler la fonction Initialisation ;
2 : Tant Que (Horloge ≤= temps prédéfinie)
3 :     Défiler le prochain événement de la liste FEL ;
4 :     Horloge=temps de l'événement ;
5 :     Appeler une fonction d'exécution de routine en fonction de type d'événement ;
6 : Fin Tant Que ;
7:  Appeler la fonction d'exécution de la routine Statistique ;
8:  Fin
```

*Figure 4.13: Pseudo Algorithme pour la routine Boucle Principale*

- **Routine Statistiques**

Elle est exécutée à la fin de la simulation pour donner des résultats des mesures finales de la simulation.
Ils seront bien détaillés à la fin du chapitre.

## 4.3.D. Diagramme de classe

Le Langage de Modélisation Unifié (UML), est un langage de modélisation graphique à base de pictogrammes conçu comme une méthode normalisée de visualisation dans les domaines du développement logiciel et en conception orientée objet. Parmi ces diagrammes, nous allons nous concentrer sur le diagramme de classes qui décrit l'architecture conceptuelle du système.
Le diagramme de classes et sa description sont illustrés respectivement dans la figure 4.14 et le tableau 3 suivants :

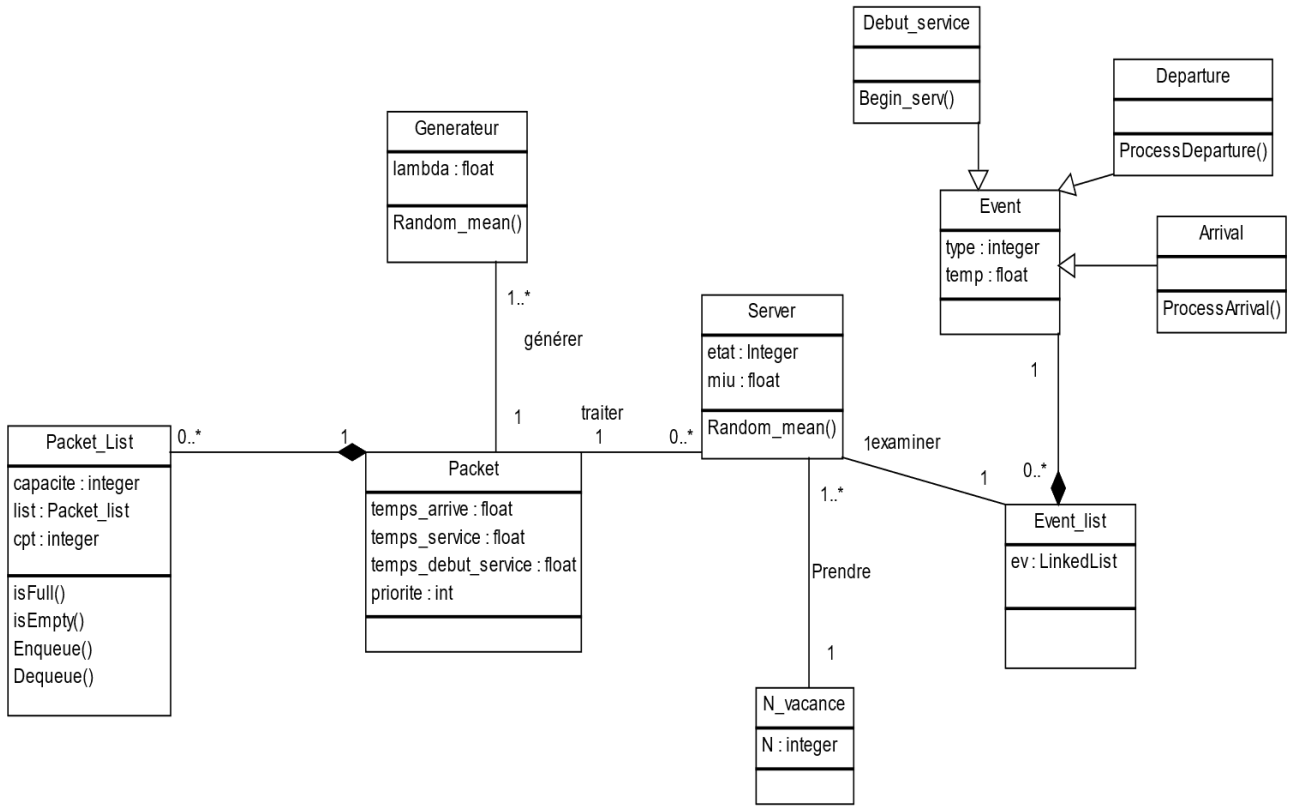



*Figure 4.14: Diagramme de classes*

| Classe | Attribut | Type | Définition | Méthode |
|---|---|---|---|---|
| Générateur | Lambda | float | Taux d'arrivée λ | Random_mean() |
| Packet | temps_arrivée | float | Temps d'arrivée de paquet | |
| | temps_service | float | Temps de service de paquet | |
| | temps_début_service | float | Temps du début service de paquet | |
| | Priorité | integer | La classe de priorité de paquet | |
| Packet_list | Capacité | integer | Taille maximale de la file d'attente | IsFull()<br>IsEmpty()<br>Enqueue()<br>Dequeue() |
| | List | PriorityQueue | La file d'attente | |
| | Cpt | integer | Compteur des paquets dans la file d'attente | |
| Server | État | integer | L'état de serveur : 1 pour occupe | Random_mean() |

| | | | | |
|---|---|---|---|---|
| | | | 2 pour vacance | |
| | Miu | float | Taux de service | |
| N_vacance | N | Integer | La seuil | |
| Event | Type | Integer | Type événement :<br>1 pour arrivée<br>2 pour départ<br>3 pour vacance | |
| | Temps | float | Le temps d'événement | |
| Event_list | Ev | LinkedList | Liste d'événement future | |
| Arrival | | | Événement d'arrivée | ProcessArrival() |
| Debut_service | | | Événement de debut de service | ProcessDeparture() |
| Departure | | | Événement de départ | Begin_Serv() |

*Table 4 : Tableau descriptif des classes*

## 4.4. Les mesures de performance

On a vu les différents aspects nécessaires pour la conception du notre simulateur, maintenant nous devons connaître les mesures de performance du modèle de file d'attente utilisé.
Pour bien étudier le changement des classes de priorité et le changement des distributions d'arrivée et de service et leurs influences sur la performance d'un nœud de capteur, dans les deux cas : buffer avec une infinie capacité et buffer avec une finie capacité.

### 4.4.A. Temps de réponse moyenne (Séjour)

Représente le temps moyen passé par un paquet depuis de son arrivé jusqu'à de son départ du serveur. [52]

**Temps de réponse moyenne (Séjour)**

```
    Lors l'événement de départ
0 : Temps de réponse =Horloge – temps d'arrivée de paquet ; //d'un seul paquet
1 : Totale temps de réponse+=temps de réponse //totale temps de réponse des toutes les paquets
2 : cpt_paquet_servis++; //incrémente le nombre des paquets servis
    A la fin de la simulation (La routine statistique)
3 : Réponse moyenne= Totale temps de réponse /cpt_paquet_servis ;
4:  Fin.
```

*Figure 4.15: Pseudo algorithme pour le calcul de temps de réponse moyenne*

### 4.4.B. Temps d'attente moyenne

Représente le temps moyen passé par un paquet dans le buffer depuis de son arrivé jusqu'à de son début de service par le serveur. [52]

**Temps d'attente moyenne**

```
    Lors l'événement de début de service
0:  Temps d'attente = Horloge – temps d'arrivée de paquet ; //d'un seul paquet
1:  Totale temps d'attente+=temps d'attente //totale temps d'attente des toutes les paquets
    Lors l'événement de départ
2:  cpt_paquet_servis++;  //incrémente le nombre des paquets servis
    A la fin de la simulation (La routine statistique)
3:  Attente moyenne= Totale temps d'attente /cpt_paquet_servis ;
4:  Fin.
```

*Figure 4.16: Pseudo Algorithme pour le calcul de temps d'attente moyenne*

### 4.4.C. Débit

Le taux de paquets qui peuvent être traitées par le serveur, et mesuré en paquets par unité de temps (secondes). [52]

**Débit**

```
    Lors l'événement de départ
0 : cpt_paquet_servis++ ; //incrémente le nombre des paquets servis
    A la fin de la simulation (La routine statistique)
1 : Débit =cpt_paquet_servis/Horloge ;
2 : Fin.
```

*Figure 4.17: Pseudo algorithme pour le calcul de débit*

### 4.4.D. Probabilité de vacance

Représente la probabilité que le serveur soit à l'état de vacance. Si aucun paquet se trouve dans la file d'attente des paquets(buffer) le serveur prend des vacances après un départ de paquet. [52]

**Probabilité de vacance**

```
     Lors l'événement de départ
0 :  Si(cpt ==0) alors
1 :      Début de vacance =Horloge ;
2 :  Fin ;
     Lors l'événement d'arrivée
3 :  Si(cpt ==N) alors
4 :      Temps de vacance+= (Horloge – Début de vacance) ;
5:   Fin ;
```

```
    A la fin de la simulation (routine statistique)
6:  Si (État de serveur == vacance ) alors
7:    Temps de vacance+= (Horloge – Début de vacance);
8:  Fin;
9:  Probabilité de vacance = Temps de vacance / Horloge ;
10: Fin ;
```

*Figure 4.18: Pseudo algorithme pour le calcul de la probabilité de vacance*

### 4.4.E. Probabilité de blocage

Représente à la probabilité que la capacité du buffer du capteur atteigne sa limite K (taille maximale du buffer) durant la simulation. [52]

**Probabilité de blocage**

```
    Lors l'événement d'arrivée
0 : Si(cpt ==K) alors
1 :   Début de saturation = Horloge ;
2 : Fin ;
    Lors l'événement de départ
3 : Temps de saturation +=( Horloge – Début de saturation );
    A la fin de la simulation (routine statistique)
4 : Probabilité de blocage = Temps de saturation / Horloge ;
5:  Fin ;
```

*Figure 4.19: Pseudo algorithme pour le calcul de la probabilité de blocage*

### 4.4.F. Taux de perte

Représente si un nouveau paquet ne peut être pas stocké à cause de la saturation du buffer. D'un autre côté, il peut représenter le pourcentage des paquets rejeté. [52]

**Taux de perte**

```
    Lors l'événement d'arrivée
0 : Totale_paquet++ ; //compteur des paquets arrivants
1 : Si(cpt ==K) alors
2 :   cpt_paquet_perdu++ ; // incrémente le compteur des paquets perdu
3 : Fin ;
    A la fin de la simulation (routine statistique)
4 : Taux de perte =cpt_paquet_perdu / Totale_paquet ;
5:  Fin ;
```

*Figure 4.20: Pseudo algorithme pour le calcul de taux de perte*

### 4.4.G.Utilisation de système

L'utilisation de système est définie comme la fraction de temps pendant lequel le serveur est occupé. [52]

**Utilisation de système**

A la fin de la simulation (routine statistique)
0 : Utilisation = taux d'arrivée (λ) / taux de service (μ) ;
1 : Fin ;

*Figure 4.21: Pseudo algorithme pour le calcul de l'utilisation de système*

### 4.4.H.Énergie consommée

Elle est calculée comme suit [24] :

$$EC = Pv * ECidle + (1 – Pv) * ECbusy + ECswitch * Nc + L * Ecpaquet$$

- **Calcule de longueur moyenne de buffer L :**

Représente le nombre moyen de paquets en attente dans le buffer.

**Longueur moyenne de buffer L**

Lors la boucle principale
0 : Défiler l'événement ;
1 : Dict[cpt] += Temps de l'événement – Horloge ;
A la fin de simulation
1 : Pour tout < cle,valeur> e ∈ Dict
2 : L = e.Cle * e.Valeur ;
3 : FinPour ;
4 : Fin ;

*Figure 4.22: Pseudo algorithme pour le calcul de la taille moyenne de buffer*

- **Nombre de cycle Nc :**

Le nombre de transitions de l'état inactif à l'état occupé par unité de temps. [52]

**Nombre de cycle Nc**

Lors l'événement d'arrivée
0 : Si (cpt == N) alors
1: Nombre_de_cycles++ ;
2: Fin ;
A la fin de simulation (routine statistique)
3: Nc= Nombre_de_cycles/Horloge ;
4: Fin ;

*Figure 4.23: Pseudo algorithme pour le calcul le nombre des cycles*

## 4.5. Conclusion

Dans ce chapitre nous avons modélisé le système étudié, ainsi réalisé une conception sur les aspects important liés au simulateur et à l’approche de simulation à événement discret que nous avons choisi. Nous avons présenté la politique de vacance proposée, puis une modélisation avec les files d’attente avec vacances. De plus, nous avons présenté les mesures de performances pour l’évaluation des performances d’un nœud de capteur sans fil.

Dans le prochain chapitre, on va faire des comparaisons entre les différents résultats obtenus par notre système.

# 5. *Résultat et études expérimentales*

# 5.1. Introduction

Notre objectif est maintenant de mettre en œuvre un outil de simulation pour évaluer les performances d'un nœud de capteur sans fil. Dans cet outil, nous avons mis en œuvre la théorie que nous avons apprise dans les deux derniers chapitres, afin que nous puissions voir les différents effets sur notre système.

Ce chapitre est dédié à présenter les fonctionnalités de notre application, à montrer toutes les différentes influences de certains paramètres du système tels que le taux de service et le seuil N et les différentes distributions sur la consommation d'énergie.

# 5.2. Présentation de l'application

## 5.2.A. Pour Quoi le C#

C# a de nombreuses autres raisons d'être populaire et demandé. Quelques-unes des raisons sont mentionnées ci-dessous :

- Facile à démarrer : C# est un langage de haut niveau, il est donc plus proche d'autres langages de programmation populaires tels que C, C++ et Java et devient ainsi facile à apprendre pour tout le monde.
- Largement utilisé pour développer des applications de bureau et Web : C# est largement utilisé pour développer des applications Web et des applications de bureau. C'est l'un des langages les plus populaires utilisés dans le bureau professionnel. Si quelqu'un veut créer des applications Microsoft, C# est son premier choix.
- Communauté : plus la communauté est grande, mieux c'est, car de nouveaux outils et logiciels seront développés pour l'améliorer. C# a une grande communauté, donc les développements sont faits pour le faire exister dans le système et ne pas s'éteindre.
- Développement de jeux : C# est largement utilisé dans le développement de jeux et continuera de dominer. C# s'intègre à Microsoft et a donc un large public cible. Les fonctionnalités C# telles que Automatique Garbage Collection, les interfaces, l'orientation objet, etc. font de C# un langage de développement de jeux populaire.

### 5.2.B. Fenêtre d'accueil

Représente la première fenêtre qui apparait dans le début de programme de simulation.

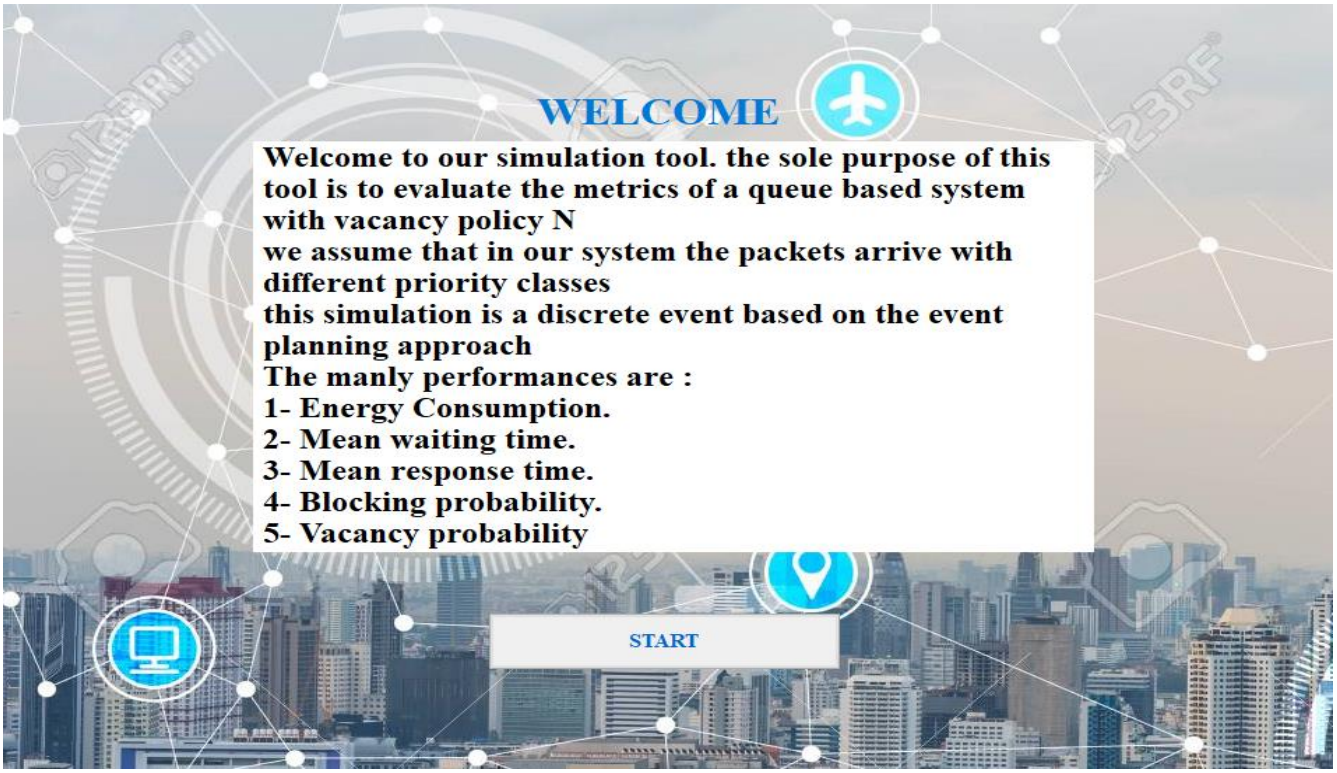



*Figure 5.1: La fenêtre d'accueil de l'application*

### 5.2.C. Fenêtre des choix des paramètres

L'utilisateur doit saisir tout les paramètres d'entre pour que la simulation fonctionnée il doit choisis le type de variable étudié ( soit la seuil N soit le taux de service soit la capacite maximale de buffer K soit l'énergie dans l'état initial) ,afin d'obtenir des graphes montrons l'influence de ces variables.

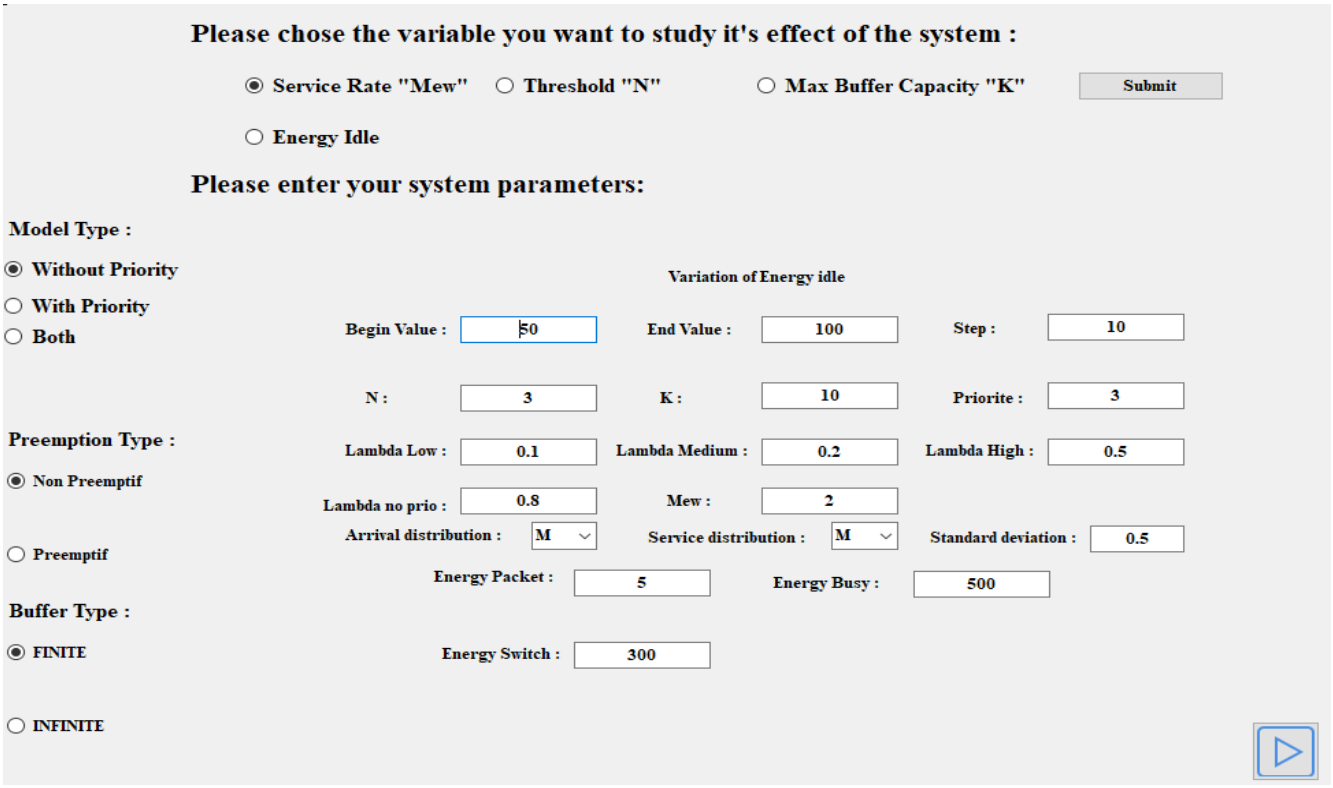



*Figure 5.2: La fenêtre des choix des paramètres*

## 5.2.D. Fenêtre des résultats

Cette fenêtre permet d'afficher les résultats obtenus de simulation sous forme des graphes. Et cette fenêtre contient des buttons qui affiche les graphes suivants :

1- **Throughput :**
   Permet d'afficher le graphe de débit.
2- **Mean Waiting Time :**
   Permet d'afficher le graphe du temps d'attente moyen.
3- **Mean Response Time :**
   Permet d'afficher le graphe du temps réponse moyen.
4- **Vacancy Probability :**
   Permet d'afficher le graphe de la probabilité de vacance.
5- **Blocking Probability :**
   Permet d'afficher le graphe de la probabilité de de blocage.
6- **Energie Consumption :**
   Permet d'afficher le graphe du la consommation d'énergie.

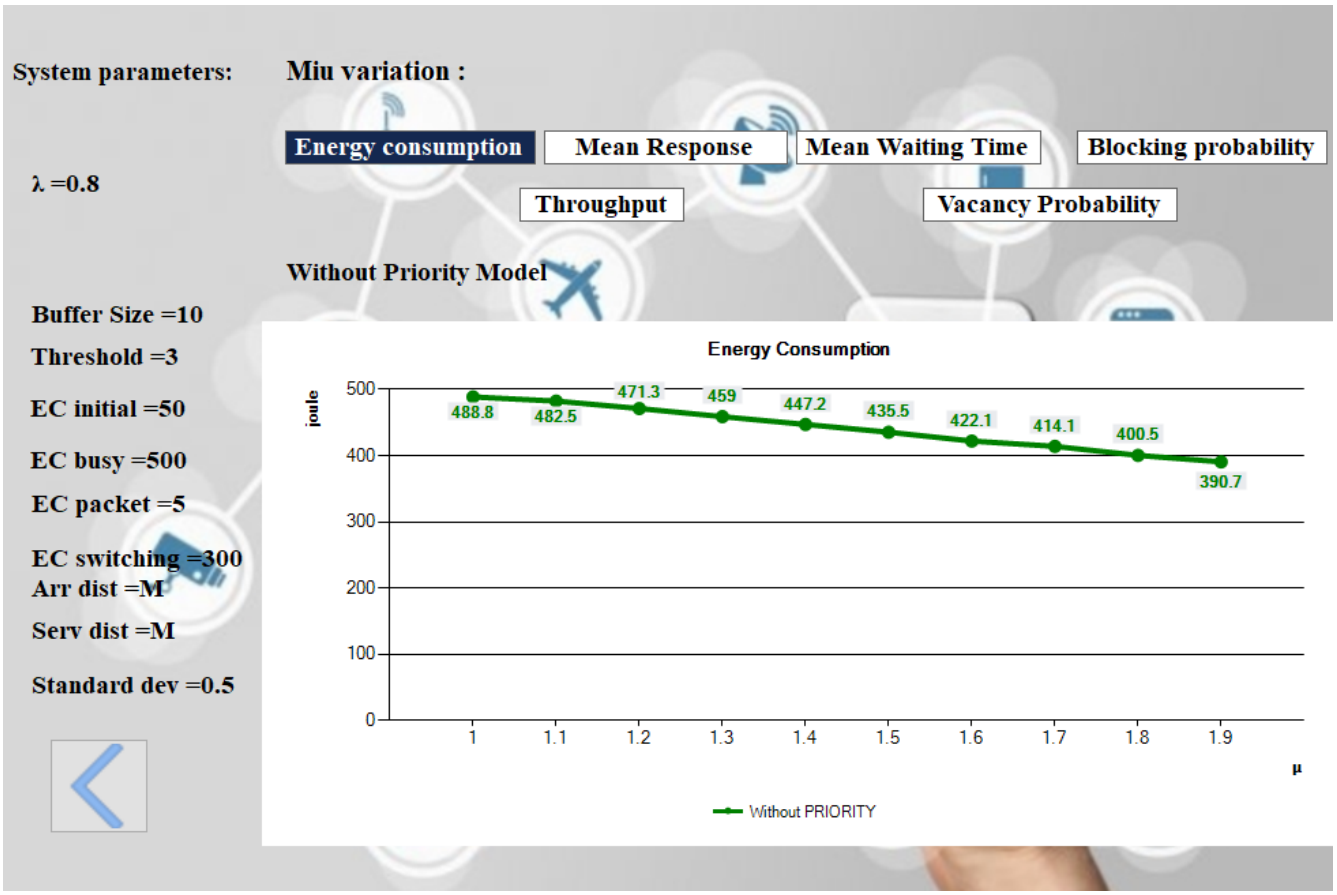



*Figure 5.3: la fenêtre des résultats*

## 5.3. Les fonctions de génération des variables aléatoires

Avoir une routine pour générer des valeurs aléatoires pour les variables est une étape très nécessaire dans une simulation, avec une distribution aléatoire bien déterminé.

Notre modèle basé sur une file d'attente A/B/1, les temps inter-arrivées et les temps de service sont distribués de manière :

**M** : Exponentielle, **G** : Générale et **D** : Déterministique.

Les principales caractéristiques de chaque distribution sont implémentées avec le langage java dans les figures suivantes ci-dessous :

- Pour la distribution M :

```
public  float Exponontiel(float ma)
{
   RNGCryptoServiceProvider Random = new RNGCryptoServiceProvider();
   uint s = uint.MaxValue;
   byte[] array = new byte[4];
   Random.GetBytes(array);
   s = BitConverter.ToUInt32(array, 0);
   var X= (s / (double)uint.MaxValue);
   return (float)Math.Log(X) / (-1 * ma);
}
```

*Figure 5.5 : implémentation d'un générateur pour la distribution M [52]*

- Pour la distribution G :

```
public double GE(double mean, double standard_deviation)
{
    RNGCryptoServiceProvider Random = new RNGCryptoServiceProvider();
    uint s = uint.MaxValue;
    byte[] array = new byte[4];
    Random.GetBytes(array);
    s = BitConverter.ToUInt32(array, 0);
    var X = (s / (double)uint.MaxValue);
    return (float)(mean + (X * standard_deviation));
}
```

*Figure 5.6 : implémentation d'un générateur pour la distribution G*

- Pour la distribution D :

La distribution déterministe n'a pas besoin de générateur car on utilise les temps inter arrivées et les temps de service saisis tels quels est.

## 5.4. Résultats numériques

Pour montrer l'efficacité du notre modelé, on va présenter des résultats numériques qu'on a obtenus dans les deux cas (buffer fini et infini).

Les paramètres du système utilisés dans l'étude expérimental sont liste dans le tableau 5 ces dessous :

| Paramètre | Valeur |
|---|---|
| Temp maximal de simulation | 10000 |
| Le seuil de file d'attente N | de 3 a 9 |

| | |
|---|---|
| Taux d'arrive pour la Low priorité | 0.1 |
| Taux d'arrive pour la Medium priorité | 0.2 |
| Taux d'arrive pour la High priorité | 0.5 |
| Taux d'arrive pour le modèle non prioritaire | 0.8 |
| Taille maximale du buffer K | de 10 a 30 |
| Taux de service moyen µ | de 1 a 2 |
| Energie dans l'état idle | de 50 a 100 |
| Energie dans l'état busy | 500 |
| Energie de chaque paquet | 5 |
| Energie de switch | 300 |
| La distribution d'arrive | M |
| La distribution de service | M |
| Nombre des class de priorité | 3 |
| Politique d'ordonnancement | Non préemptif |

*Tableau 5 : les paramètres de système*

### 5.4.A. Effet de la variation de taux de service

Nous avons montré l'impact de la variation de taux de service sur la consommation moyenne d'énergie et le délai d'attente et la probabilité de blocage (dans le cas fini).

***La consommation d'énergie moyenne :***

Les deux figures 5.7 et 5.8 qui apparaissant ci-dessous représente la consommation d'énergie pour les deux modèles (avec trois class de priorité et sans class de priorité) pour différents taux de service moyens.

1. Buffer fini :

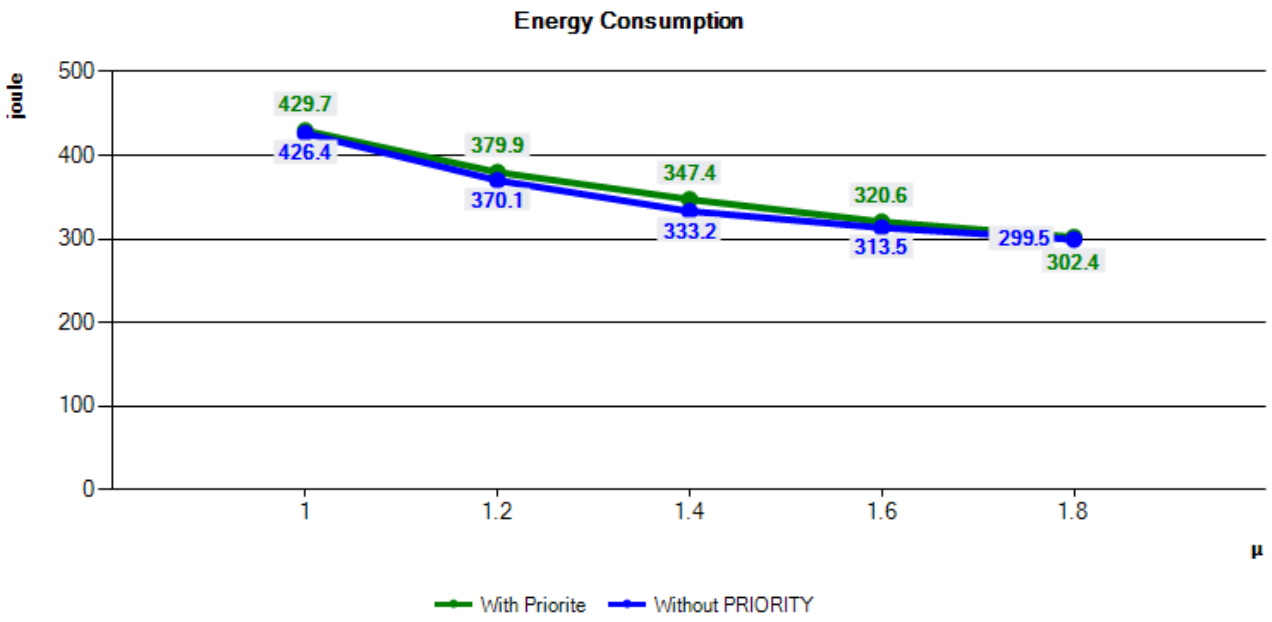



*Figure 5.7 : L'énergie consomme moyenne en fonction de taux de service*

2. Buffer infini :

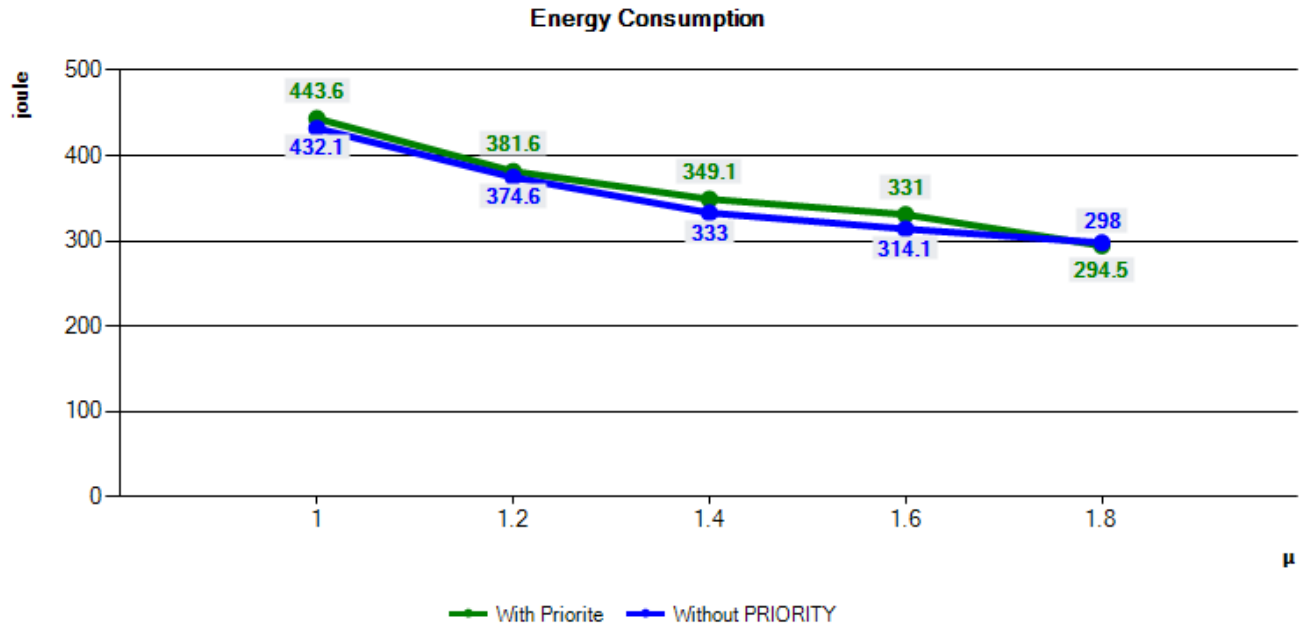



*Figure 5.8: La consommation d'énergie moyenne en fonction de taux de service dans le cas infini*

- Analyse :

  La consommation d'énergie est la même pour les deux modèles. L'augmentation du taux de service par le nœud de capteur sans fil affecte sur la consommation d'énergie, par ce que le serveur fait plusieurs services.

***Le délai d'attente :***

Les figures 5.9 et 5.10 apparaissant ci-dessous représente l'influence dans le délai d'attente pour les deux modèles (avec trois class de priorité et sans class de priorité) pour différents taux de service moyens.

1. Buffer fini :

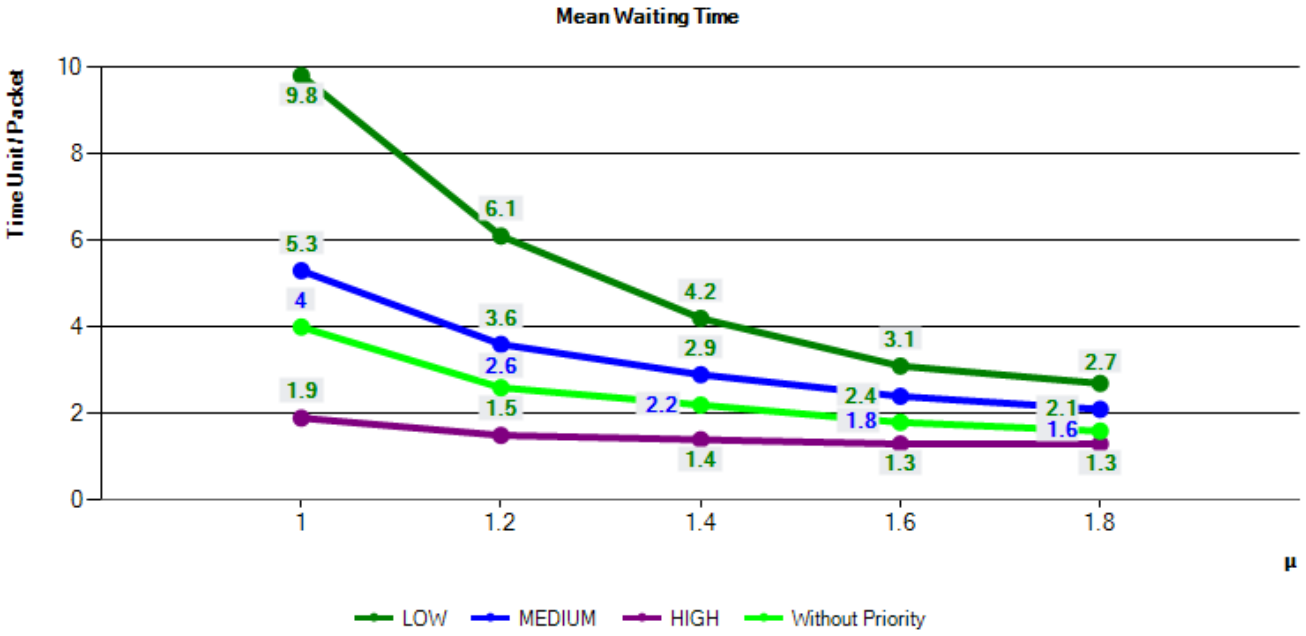



*Figure 5.9 : le délai d'attente moyenne en fonction de taux de service*

2. Buffer infini :

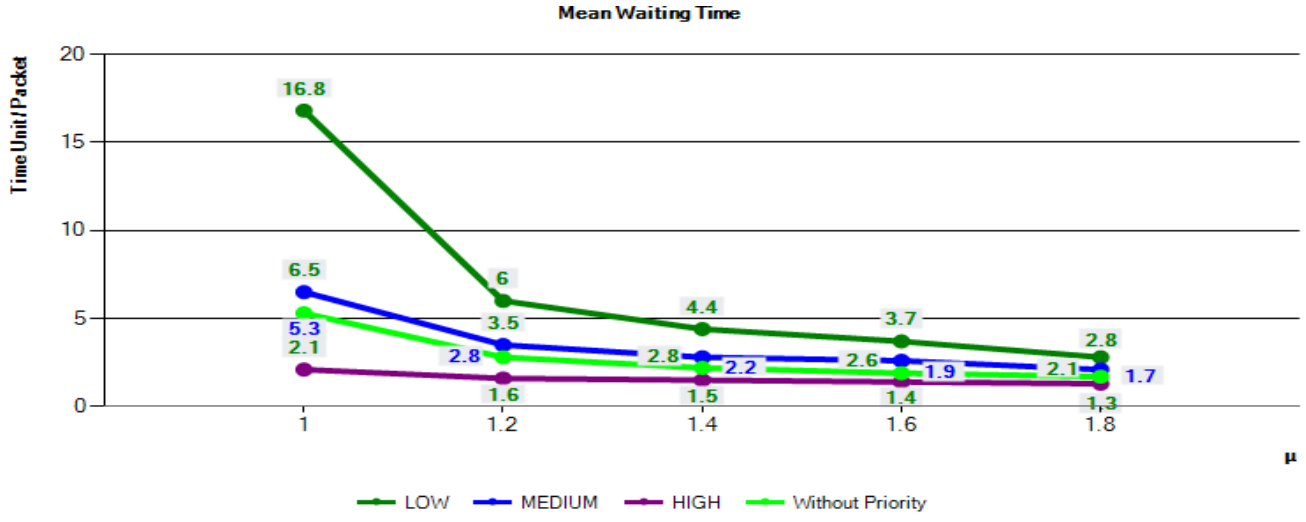



*Figure 5.10: le délai d'attente moyenne en fonction de taux de service dans le cas infini*

- Analyse :

  L'augmentation du taux de service diminue le temps d'attente des paquets dans le buffer dans les deux modèles, et en remarquant que le modèle non prioritaire a un temps d'attente inférieur aux classes faible et moyenne priorité sur la réduction du temps d'attente moyen dans le buffer, mais les paquets avec High priorité sont les plus faible concernant le temps d'attente.

***La probabilité de blocage :***

La figure 5.11 apparaissant ci-dessous représente l'influence dans la probabilité de blocage pour les deux modèles.

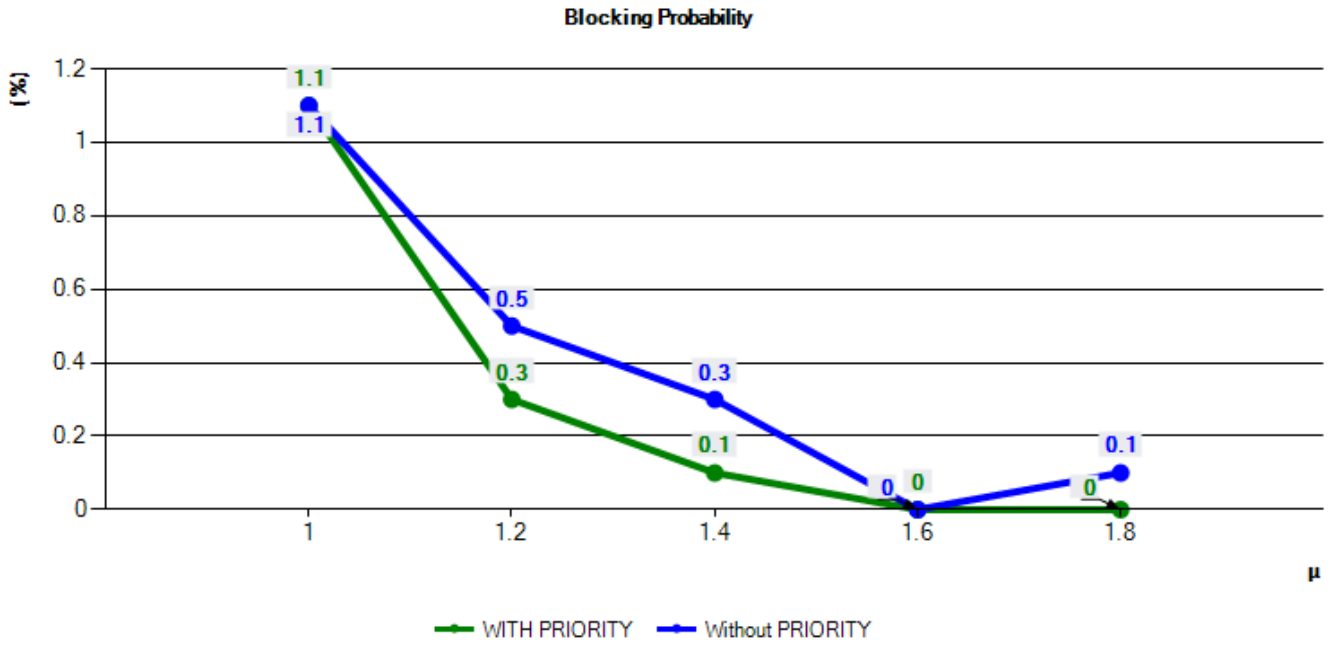



*Figure 5.11: La probabilité de blocage en fonction de taux de service*

- Analyse :

  Dans les deux modèles on remarque que la probabilité de blocage se diminue a cause que le serveur servis les paquets plus rapidement et sa augment le temps pour que le buffer atteindre son capacite maximal

## 5.4.B. Effet de variation de la seuil N

Ici nous avons montré l'impact de la variation du seuil N sur la consommation moyenne d'énergie et le délai d'attente et la probabilité de blocage (dans le cas fini).

### *La consommation d'énergie moyenne*

La figure 5.14 apparaissant ci-dessous représente la consommation d'énergie pour les deux modèles (avec trois class de priorité et sans class de priorité) pour différents seuil N.

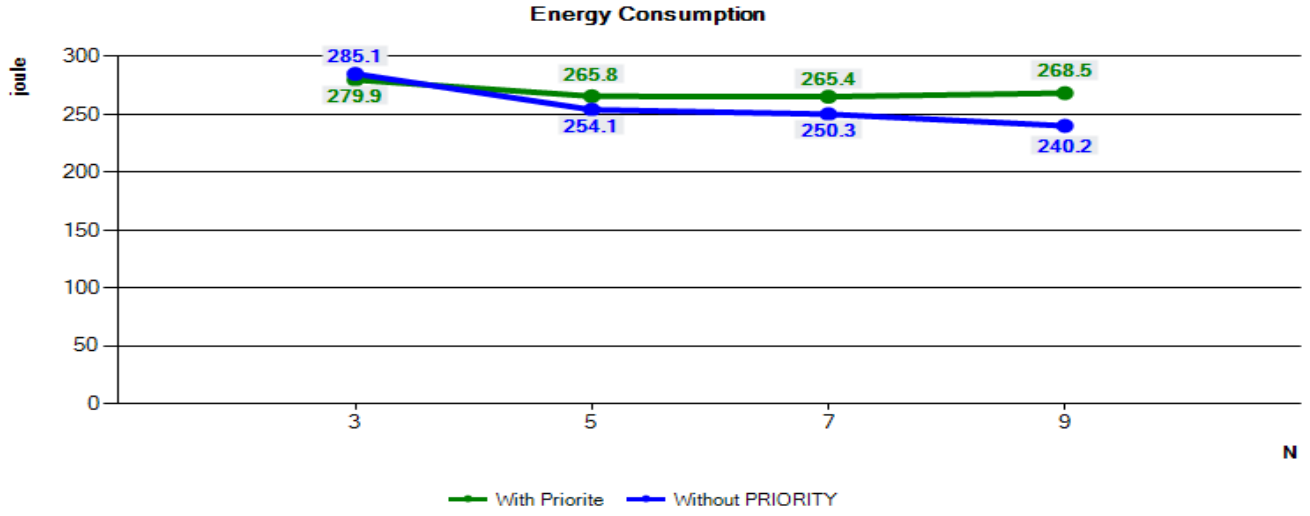



*Figure 5.14: la consommation d'énergie moyenne en fonction de la seuil N*

- Analyse :

  Dans les deux modèles étudiés on voir que l'augmentation de la seuil N, influence par un effet décroissance dans la consommation énergétique par ce que la durée de vacance du notre capteur sans fil sera étendre.

### *Le délai d'attente :*

La figure 5.15 apparaissant ci-dessous représente l'influence dans le délai d'attente pour les deux modèles (avec trois class de priorité et sans class de priorité) pour différents seuil N.

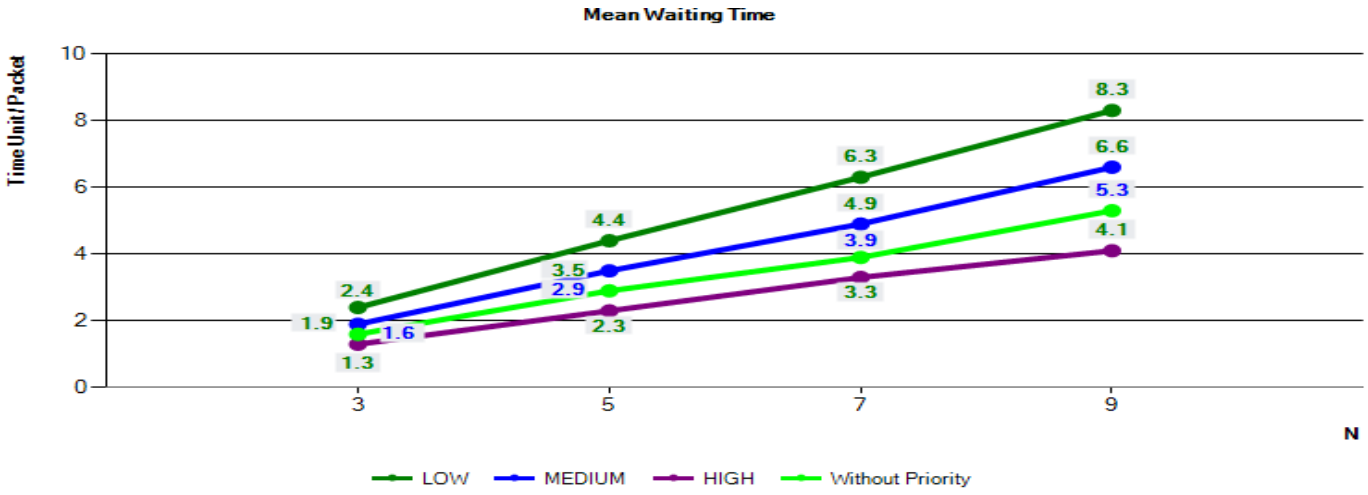



*Figure 5.15: Le délai d'attente moyenne en fonction de la seuil N*

- Analyse :

  Dans les deux modèles étudié, le délai d'attente sera augmenté avec n'importe quelle classe de priorité.

***La probabilité de blocage :***

La figure 5.16 apparaissant ci-dessous représente l'influence dans la probabilité de blocage pour les deux modèles.

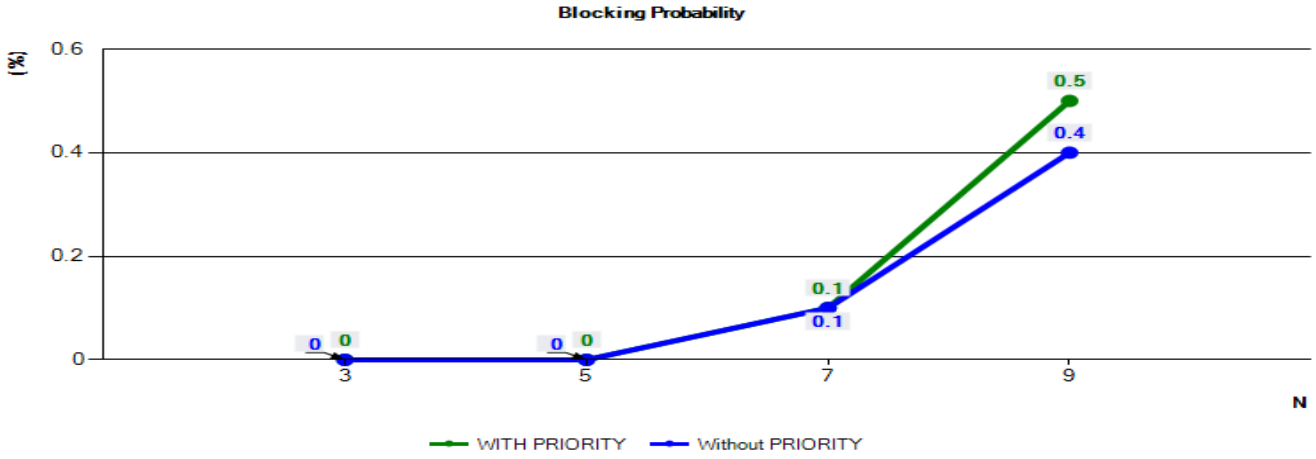



*Figure 5.16: La probabilité de blocage en fonction du seuil N*

- Analyse :

  Dans les deux modèles étudiés, la probabilité de blocage sera toujours dans un état d'augmentation a cause de la rapidité d'atteindre de capacite maximale de notre buffer.

### 5.4.C. Effet de variation de l'énergie initial

Nous avons montré l'impact de la variation de l'énergie dans l'état initial du buffer sur la consommation moyenne d'énergie.

***La consommation d'énergie moyenne :***

Les figures 5.12 et 5.13 qui apparaissant ci-dessous représente la consommation d'énergie pour les deux modèles (avec trois class de priorité et sans class de priorité) pour l'énergie initial.

1. Buffer fini :

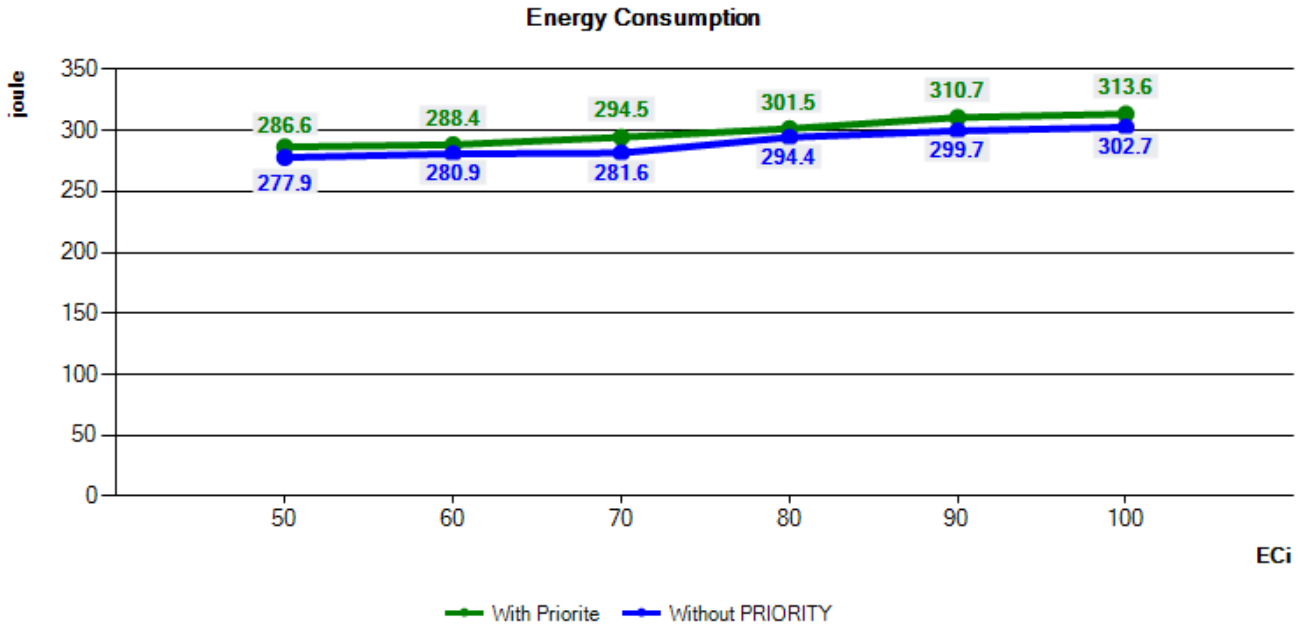



*Figure 5.12: Energie consomme moyenne en fonction de l'énergie en état idle*

2. Buffer infini :

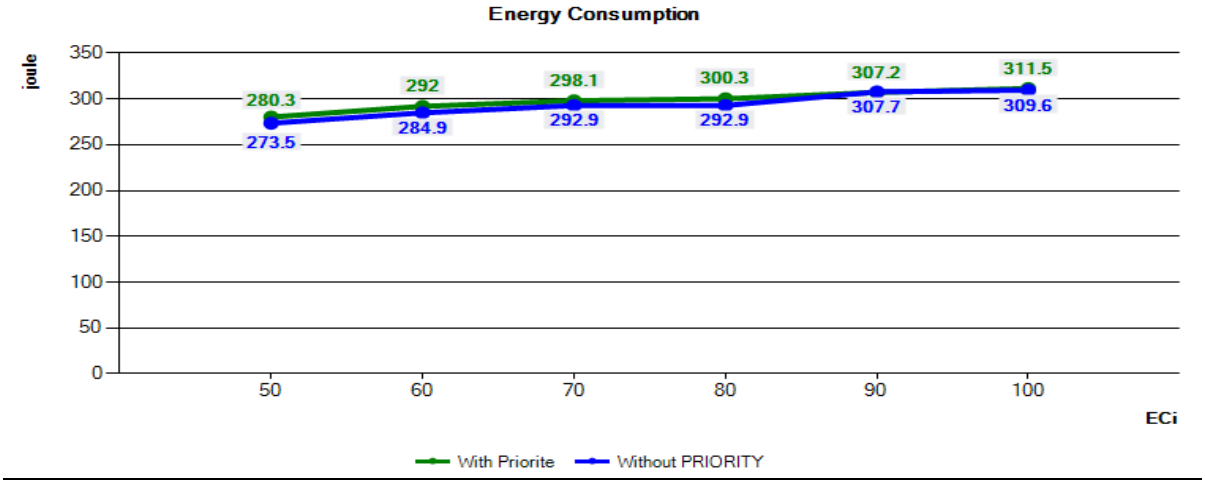



*Figure 5.13: l'énergie consomme moyenne en fonction de l'énergie initial dans le cas fini*

- Analyse :

  Dans les deux modèles , la consommation énergétique sera augmenté toujours quand en variée l'énergie initial du buffer

### 5.4.D. Effet de variation de la capacité maximal du buffer

Nous avons montré l'impact de la variation de la taille maximale du buffer sur la consommation moyenne d'énergie et le délai d'attente et la probabilité de blocage (dans le cas fini).

***La consommation d'énergie moyenne :***

La figure 5.17 apparaissant ci-dessous représente la consommation d'énergie pour les deux modèles (avec trois class de priorité et sans class de priorité) pour différentes capacités maximales du buffer.

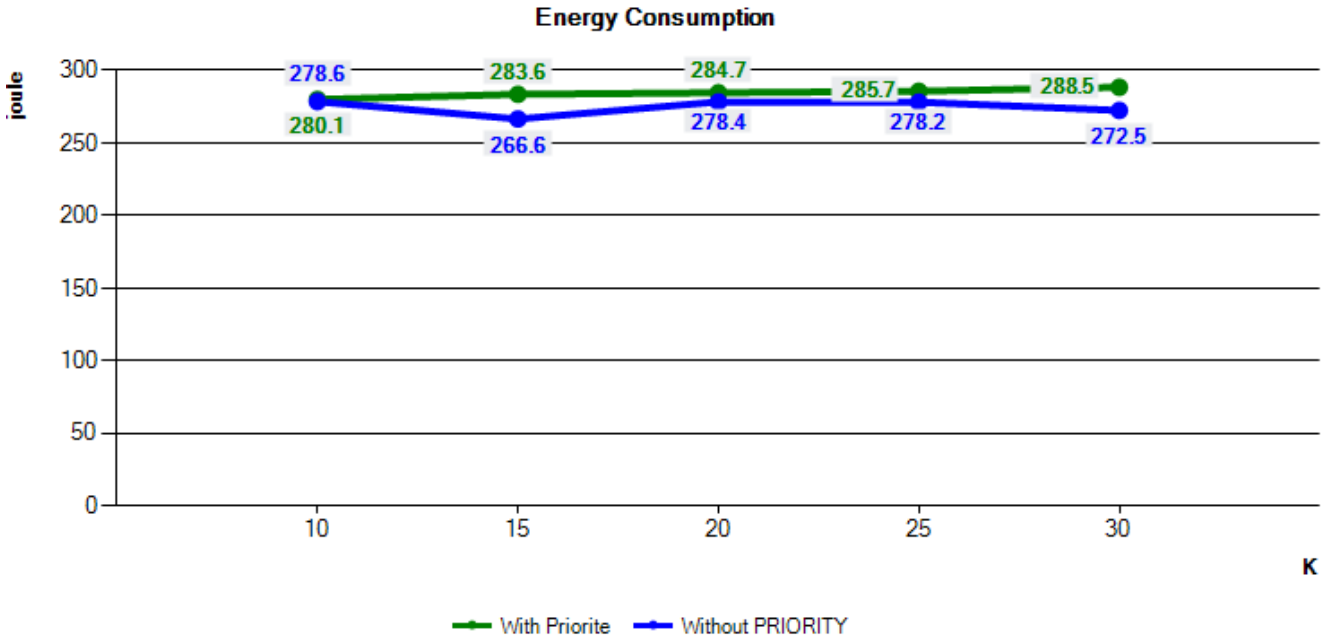



*Figure 5.17: La consommation d'énergie moyenne en fonction de la taille maximale du buffer K*

- Analyse :

  On remarque dans les deux modèles que la consommation d'énergie est presque la même et stable avec l'augmentation de la capacité maximale.

***Le délai d'attente :***

La figure 5.18 apparaissant ci-dessous représente l'influence dans le délai d'attente pour les deux modèles (avec trois class de priorité et sans class de priorité) pour différentes capacités maximales.

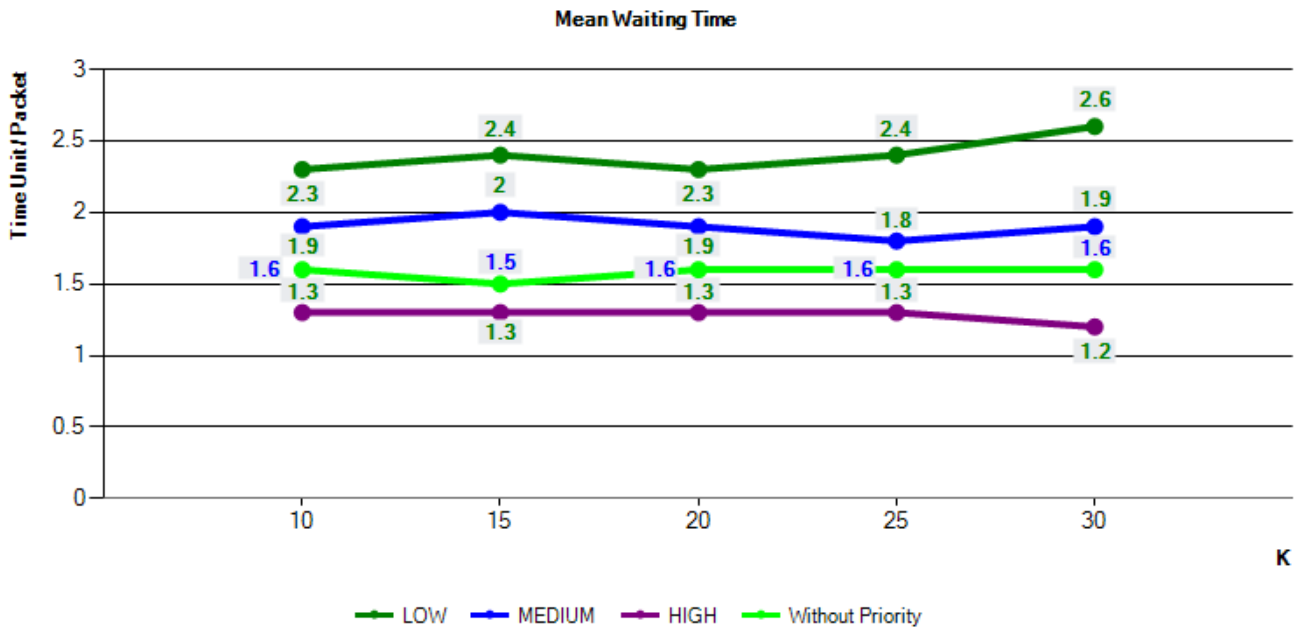



*Figure 5.18: Le délai d'attente moyenne en fonction de la taille maximale du buffer K*

- Analyse :

  Dans les deux modèles et avec n'importe quelle class de priorité, le délai d'attente est stable dans l'augmentation de la capacité maximale du buffer

### 5.4.E. Résultat final

De ce qui précède, on peut voir qu'il pas vraiment une grande différence concernant la consommation d'énergie entre les deux modèles proposés (avec/sans priorité) et que le changement de type de buffer (fini/infini) n'influence pas les résultats obtenus.

Et on peut aussi en déduire que pour n'importe quel type de différentiation des priorités de paquets et pour n'importe quel modèle utilisé, on remarquera toujours que la variation du taux de service diminue le temps d'attente par rapport à la variation de l'énergie initiale du buffer.

## 5.5. Conclusion

Au final, nous avons implémenté notre outil basé sur une simulation d'événements discrets avec une approche de planification d'événements.

Et après avoir présenté tous les aspects de notre outil et toutes ses caractéristiques, et d'après les résultats obtenus on peut dire que peut être le changement de topologie ou de protocole d'arrivé et de distribution peuvent influencer les performances de notre système.

# Conclusion générale

Dans notre travail, on a présenté les RCSFs avec leurs caractéristiques et spécifications et leurs différents types et domaines d'utilisation. On a également présenté les plusieurs défi qui rencontrent les réseaux de capteurs et on s'est particulièrement intéressé à la conservation d'énergie qui est, de nos jours, le défi majeur dans l'étude pour améliorer l'utilisation des RCSFs.

On a utilisé un système de files d'attente avec la politique de vacances N avec plusieurs classes de priorité, qui est une des meilleures méthodes pour modéliser la problématique posée.

La simulation à événements discrets nous a permis de mieux comprendre et étudier le système ciblé grâce à la possibilité de se focaliser sur les événements importants dans notre système.

Notre outil de simulation développé basé sur l'approche de simulation proposée, nous a permis de tester et évaluer les différentes mesures de performances d'un RCSF et voir l'effet que la politique de vacances, les différentes classes de priorité et la préemption de service ont importé sur le comportement de notre système variant les mesures de performance d'un cas à un autre. D'ici, cet outil nous fournit la possibilité de faire une comparaison entre les résultats obtenus pour un système avec/sans classes de priorité, avec/sans préemption.

Finalement, ce travail été dédié à la conception et implémentation d'un outil de simulation pour comparer le comportement d'un RCSF avec un traitement de clients qui diffère selon :

- La présence et absence d'un ordre de priorité entre les clients servis.
- La présence et absence de préemption de service entre les clients avec des différents ordres de priorité.
- Limité et illimité de la taille de la file d'attente.

La simulation à événement discrets est la méthode la plus adéquate pour la réalisation de ce travail grâce à ses éléments et caractéristiques qui nous aide à se focaliser sur les événements critiques du système.

## Bibliographie

[1] SADOUN, Balqies, "Applied system simulation : à review study," Information Sciences, vol. 124, pp. 173-192, 2000.

[2] JIANG, Fuu-Cheng, HUANG, Der-Chen, et WANG, Kuo-Hsiung, "Design approaches for optimizing power consumption of sensor node with N-policy M/G/1 queuing model," In: Proceedings of the 4th International Conference on Queueing Theory and Network Applications, pp. 1-8, 2009

[3] Srija Unnikrishnan, Sanjeev Ghosh «Delay Incurred in Wireless Sensor Networks Using N-Policy M/M/1 Queuing Model and its Mitigation» 2019

[4] BOUTOUMI, Bachira et GHARBI, Nawel, " An energy saving and latency delay efficiency scheme for wireless sensor networks based on GSPNs. In : 2017 4th International Conference on Control, Decision and Information Technologies (CoDIT)," IEEE, 2017. p. 0645-06

[5] JIANG, Fuu-Cheng, HUANG, Der-Chen, YANG, Chao-Tung, et al, " Design strategy for optimizing power consumption of sensor node with Min (N, T) policy M/G/1 queuing models," International Journal of Communication Systems, 2012, vol. 25, no 5, p. 652-671

[6] BOUTOUMI, Bachira et GHARBI, Nawel, " Two Thresholds Working Vacation Policy for Improving Energy Consumption and Latency in WSNs," In : International Conference on Queueing Theory and Network Applications.Springer, Cham, pp. 168-181, 2018.

[7] HASSON, Saad Talib, Simulation approach to model queuing Problems

[8] REZAEI, Zahra et MOBININEJAD, Shima, " Energy saving in wireless sensor networks," International Journal of Computer Science and Engineering Survey, vol. 3, p. 23, 2012

[9] KHRIJI, Sabrine, EL HOUSSAINI, Dhouha, KAMMOUN, Ines, et al, "Energy-efficient techniques in wireless sensor networks," In : Energy Harvesting for Wireless Sensor Networks: Technologies, Components and System Design, 2018.

[10] AMBEKAR, Dhanashri V., BHOI, Amol D., et KHARADKAR, R. D, "A survey on sensors lifetime enhancement techniques in wireless sensor networks," International Journal of Computer Applications, vol. 107, 2014.

[11] SINGH, Shio Kumar, SINGH, M. P., SINGH, Dharmendra K.,et al , " Routing protocols in wireless sensor networks–A survey.," International Journal of Computer Science & Engineering Survey (IJCSES), 2010, vol. 1, no 2, p. 63-83.

[12] Carlo Fischione,” An Introduction to Wireless Sensor Networks ‘’ Draft, version 1.8 September 2014.

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

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

[15]A. Kaufmann et R. Cruon, Les phénomènes d'attente: théorie et applications, Paris,Dunod,1961, 274p.

[16] HSU, Hwei P, " Theory and problems of probability, random variables, and random processes," New York : McGraw-hill, 1996.

[17] VERGNE, Anaıs et COMTE, Céline. , "Files d'attente.," 2018..

[18] Bose, S. K. (2013). An introduction to queueing systems. Springer Science & Business Media

[19 ] 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.

[20] KHALAF, Rehab, "On some queueing systems with server vacations, extended vacations, breakdowns, delayed repairs and stand-bys.," 2012. Thèse de doctorat. Brunel University, School of Information Systems, Computing and Mathematics.

[21] TIAN, Naishuo et ZHANG, Zhe George, "Vacation queueing models: theory and applications," Springer Science & Business Media, 2006.

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

[23] Zille Huma Kamal et Mohammad Ali Salahuddin ,'' Introduction to Wireless Sensor Networks'' , Wireless Sensor and Mobile Ad-Hoc Networks: Vehicular and Space Applications D. BenHaddou and A. Al-Fuqaha (Eds.), pp. 3-32, Springer, New York, 2015

[24] Rawat, P., Singh, K. D., Chaouchi, H., & Bonnin, J. M. , "(2013). Wireless sensor networks: a survey on recent developments and potential synergies.," The Journal of Supercomputing, 68(1), 1–48. doi:10.1007/s11227-013-1021-9..

[25] KC Rahman ,'' A Survey on Sensor Network '' ,Journal of Computer and Information Technology, 2010

[26] ANASTASI, Giuseppe, CONTI, Marco, DI FRANCESCO, Mario, et al, "Energy conservation in wireless sensor networks: A survey.," Ad hoc networks, 2009, vol. 7, no 3, p. 537-568

[27] ALTHOBAITI, Ahlam Saud et ABDULLAH, Manal, "Medium access control protocols for wireless sensor networks classifications and cross-layering," . Procedia Computer Science, 2015, vol. 65, p. 4-16.

[28] CHANCHAICHUJIT, Janya et SAAVEDRA-ROSAS, José F, "Discrete Event Simulation Concepts," In: Using Simulation Tools to Model Renewable Resources. Palgrave Macmillan, Cham, pp. 41-63, 2018.

[29] RAYCHAUDHURI, Samik, "Introduction to monte carlo simulation," In : 2008 Winter simulation conference. IEEE, pp. 91-100, 2008

[30] ĎUTKOVÁ, Silvia, ACHIMSKÝ, Karol, et HOŠTÁKOVÁ, Dominika, "Simulation of queuing system of post office," Transportation Research Procedia, pp. 1037-1044, 2019.

[31] FISHMAN, George S, "Discrete-event simulation: modeling, programming, and analysis," Springer Science & Business Media, 2013

[32] SINGH, Shio Kumar, SINGH, M. P., SINGH, Dharmendra K.,et al , " Routing protocols in wireless sensor networks–A survey.," International Journal of Computer Science & Engineering Survey (IJCSES), 2010, vol. 1, no 2, p. 63-83..

[33] P. Jayarajan, R. Maheswar, G.R. Kanagachidambaresan, V. Sivasankaran, M. Balaji and Jagannath Das «PERFORMANCE EVALUATION OF FAULT NODES USING QUEUE THRESHOLD BASED ON N-POLICY PRIORITY QUEUEING MODEL». 2018.

[34] Tyszer, J. (1999). Object-Oriented Computer Simulation of Discrete-Event Systems. The Kluwer International Series on Discrete Event Dynamic Systems.doi:10.1007/978-1-4615-5033

[35] Chiahon Chien ‘Batch size selection for the batch means method’ Proceedings of the 1994 Winter Simulation Conference ed. J. D. Tew, S. Manivannan, D. A. Sadowski, and A. F. Seila .

[36] Fatima Yousef Abdalla Barham ‘Simulation in Queuing Models: Using Simulation at Beit-eba crossing check-point‘A Thesis submitted in Partial Fulfillment of the Requirements for the Degree of Masters of Computational Mathematics, Faculty of Graduate Studies at An-ajah ational University, ablus, Palestine. 2008.

[37] Peter W.Glynn and Donald Ligleheart ‘Notes: Conditions for the Applicability of the Regenerative Method’ Published Online:1 Sep 1993. Operations Research Letters Volume 10, Issue 8, November 1991, Pages 437-443.

[38] Alexopoulos, C., Andradottir, S., Tanik Argon, N., & Goldsman, D. (2007). Replicated batch means for steady-state simulations with initial transients. 2007 Winter Simulation Conference. doi:10.1109/wsc.2007.4419617.

[39] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci : Wireless sensor NETWORKS : a survey, Broadband and Wireless Networking Laboratory, School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332, USA, 20 December 2001

[40] EHSANIFAR, Mohammad, HAMTA, Nima, et HEMESY, Mahshid, "A Simulation Approach to Evaluate Performance Indices of Fuzzy Exponential Queuing System (An M/M/C Model in a Banking Case Study)," Journal of Industrial Engineering and Management Studies, vol. 4, pp. 35- 51, 2017.

[41] 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.

[42] MULYODIPUTRO, Muhammad Dermawan et SUBANAR, Subanar, "Simulation of queue with cyclic service in signalized intersection system," International Journal of Advances in Intelligent Informatics, vol. 1, pp. 30-40, 2015.

[43] Larry Wasserman, *All of Statistics : A Concise Course in Statistical Inference*, New York, Springer-Verlag, 15 septembre 2004, 461 p ,définition 2.16, page 25.

[44] "ReVibe Energy - Powering The Industrial IoT". revibeenergy.com. Archived from the original on 22 September 2017. Retrieved 3 May 2018.

[45] "THE WORLD LEADER IN VIBRATION HARVESTER POWERED WIRELESS SENSING SYSTEMS". THE WORLD LEADER IN VIBRATION HARVESTER POWERED WIRELESS SENSING SYSTEMS. Archived from the original on 13 April 2018. Retrieved 3 May 2018.

[46] Saleem, Kashif; Fisal, Norsheila; Hafizah, Sharifah; Kamilah, Sharifah; Rashid, Rozeha; Baguda, Yakubu (2009). "Cross layer based biological inspired self-organized routing protocol for wireless sensor network". TENCON 2009 - 2009 IEEE Region 10 Conference: 1–6.

[47] Guowang Miao; Jens Zander; Ki Won Sung; Ben Slimane (2016). Fundamentals of Mobile Data Networks. Cambridge University Press. ISBN 978-1107143210.

[48] Aghdam, Shahin Mahdizadeh; Khansari, Mohammad; Rabiee, Hamid R; Salehi, Mostafa (2014). "WCCP: A congestion control protocol for wireless multimedia communication in sensor networks". Ad Hoc Networks. 13: 516–534.

[49] Magno, M.; Boyle, D.; Brunelli, D.; O'Flynn, B.; Popovici, E.; Benini, L. (2014). "Extended Wireless Monitoring Through Intelligent Hybrid Energy Supply". IEEE Transactions on Industrial Electronics.

[50] Xenakis, A.; Foukalas, F.; Stamoulis, G. (2016). "Cross-layer energy-aware topology control through Simulated Annealing for WSNs". Computers & Electrical Engineering. 56: 576–590.

[51] IBE, Oliver, "Markov processes for stochastic modeling," Newnes, 2013

[52] Hamza Nekaa, Kias. M. (2020). Un Outil de Simulation pour l'analyse des Performances d'un Nœud Capteur en Utilisant les Politiques de Vacance. Blida : Université de blida 01.

[53] osman Balci, 1995 . Principales and techniques of simulation validation, verification, and testing. Blacksburg ,Virginia 24061-0106, USA

[54] Health Services Organization in the Event of Disaster (PAHO, 1989, 129 pages)

[55] Processus stochastique, Jean Christopher Breton 2020,universite de rennes 1.

[56] Allouache Djedjiga, Azamoum Karima (2014), Optimisation de la consommation d'énergie dans les réseaux de capteurs sans fil. BEJAIA : Université de A.MIRA.

[57] Abdelbasset Trad (2014), Performance trade-offs of encryption algorithms for Wireless Sensor Networks , CTI Riyadh, KSA.

[58] Ian F. Akyildiz, Erich P.Stuntebeck (2006), wireless underground sensor network : Research challenges .

[59] Nils Morozs , Paul Daniel Mitchell, Rahul Mourya , Yury Zakharov, (2018) ,Robust TDA-MAC for practical underwater sensor network deployment: Lessons from USMART sea trials , Conference: ACM International Conference on Underwater Networks & Systems (WUWNet) At: Shenzhen, China

[60] Chan Min Park, Rana Asif Rahman, Byung-Seo Kim, (2017) Packet Flooding Mitigation in CCN-Based Wireless Multimedia Sensor Networks for Smart Cities.

[61] S. Munir, Biao Ren, Weiwei Jiao, Bin Wang, Dongliang Xie, J. Ma, (2017) Packet Flooding Mitigation in CCN-Based Wireless Multimedia Sensor Networks for Smart Cities. Computer Science 21st International Conference on Advanced Information Networking and Applications Workshops (AINAW'07).

[62] Aleksandar Milenkovic, Chris Otto & Piet C de Groen , (2005) A wireless body area network of intelligent motion sensors for computer assisted physical rehabilitation, Journal of NeuroEngineering and Rehabilitation

[63] Satyam Gupta, Manpreet Singh, Saumya Srivastava (2018) Wireless Sensor Network: A Survey