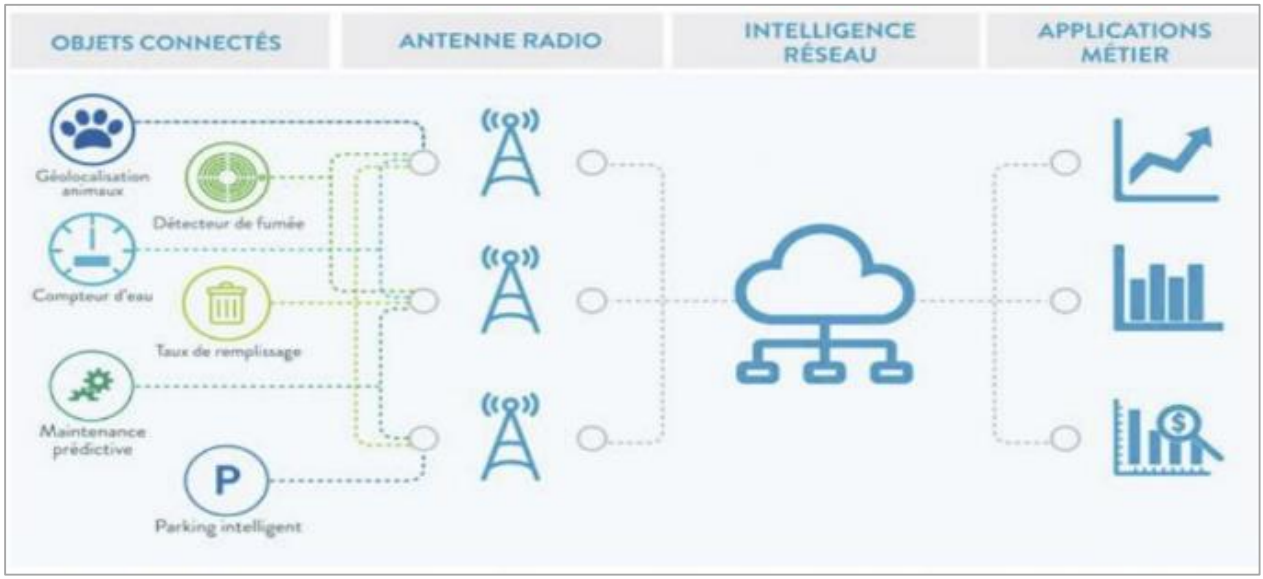
UVSQ

université PARIS-SACLAY

## 1ème Année Master Informatique, Semestre 2

## Module : SIMULATION

# *Estimation du taux de collision dans un réseau LoRa*





**Rapport réalisé par :**

SARAH OUHOCINE (AMIS)

DJILLALI BOUTOUILI (AMIS)

ABDELOUAHAB TAFAT (AMIS)

**Projet proposé par :**

M. YOUSSEF AIT EL MAHJOUB

M. FRANCK QUESSETTE


**Année universitaire 2021/2022**

## 1. Introduction

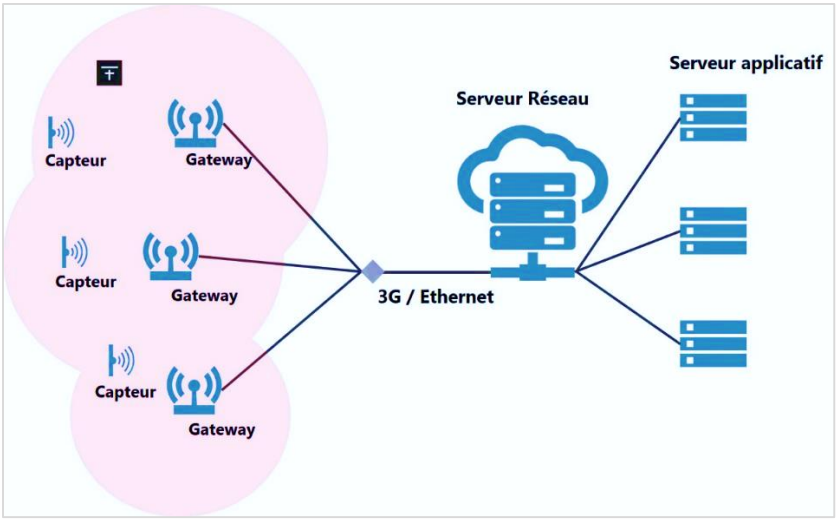
Tout comme tout autre domaine de la technologie en générale, et en l'informatique plus précisément, les réseaux de communication ont connu et ne cessent de connaitre de plus en plus d'avance technologique.

Parmi ces derniers, on retrouve une technologie qui a vite fait entendre son nom sur le marché national français et international de l'internet des objets, et c'est les réseaux **LoRa**.

Les réseaux LORA appartiennent à la famille des LPWAN, abréviation pour "Low Power Wide Area Network" qui ne se caractérisent pas par leur haut débit mais plutôt par leur efficacité énergétique avec des couts de maintenance minimes et une couverture pouvant être très étendues.

Plusieurs géants de la communication ont directement commencé à intégrer des services à réseaux LORA, on peut citer Bouygues Telecom, Orange et bien d'autres.

Les réseaux LORA dans leurs architectures, se composent principalement de trois éléments principaux sont : les capteurs, les passerelles et un serveur central.




Dans cette étude, on s'intéresse à la communication capteurs passerelles (En Rose). Ceci en implémentant un simulateur en temps continu afin de mesurer certains indices de performances notamment la probabilité de collision à chaque état, le temps d'émission moyen d'un paquet, le nombre de paquets perdus après la 7eme émission échouées…etc. en variant plusieurs variables qui sont le nombre de capteurs K, la durée d'attente moyenne avant émission i, la durée Moyenne d'émission e et la durée Moyenne d'attente après collision avant réémission.

## 2. Implémentation

Dans cette partie, nous allons aborde la partie de l'implémentation du simulateur **LORA,** nous allons commencer par justifier nos choix techniques avant de faire une présentation des fonctions utilisées dans le code

### 2.1 Variables de la simulation :

Les principaux paramètres de la simulation sont :

- Le nombre de capteurs **K** : C'est le nombre de capteurs à envoyer des paquets en destination des passerelles.
- Le temps d'attente moyen avant émission d'un paquet **i**
- Le temps d'émission moyen d'un paquet d'un capteur c vers une passerelle **e**.
- Le temps d'attente moyen avant émission d'un paquet d'un capteur c vers une passerelle p après une collision **w**.

Les principales variables de la simulation sont :

- Le nombre de paquets envoyés par chaque capteur par chaque état.
- Le nombre de collisions par état pour tous les capteurs.
- Le nombre de tentatives d'envois par état pour tous les capteurs.
- Le Temps de simulation.
- Une variable CANAL qui décrit si le canal est libre ou pas.
- Le temps d'émission par état.
- Un échéancier contenant les évènements à venir.

### 2.2 La structure de données de l'échéancier :

L'échéancier est l'un des éléments les plus importants de la simulation, il nous sert à gérer les événements à venir dans un temps continu.

Nous avons opté dans notre implémentation à un échéancier sous forme de **liste linéaire chainée** pour les avantages que cette structure apporte par rapport à un tableau, notamment :

- La possibilité de supprimer un nœud évènement déjà passé afin de libérer de la mémoire.
- Une taille allouée dynamiquement.

Nous avons aussi défini la structure d'un évènement comme suit :

- **Numéro du capteur** : Le numéro du capteur qui va lancer l'événement.
- **Date d'arrivée** : Le temps d'arrivée de l'évènement en question.
- **Type de l'évènement** : Le type de l'évènement (Début d'émission ou Fin d'émission)

Et voici ci-dessous la structure d'un nœud de l'échéancier :

- **Evènement** : Un objet de type évènement comme décrit ci-dessus.
- **Suivant :** Un pointeur vers le prochain évènement de la liste.

### 2.3 Explication du code :

Nous allons essayer de présenter toutes les fonctions et structures que nous avons utilisé dans notre code pour aboutir au simulateur :

#### 2.3.1 Les fonctions de génération des variables aléatoire :

- **unifC () :** renvoie une variable aléatoire continue dans ]0..1[
- **Expo_Duree(lambda) :** renvoie une variable aléatoire de loi exponentielle lambda.

#### 2.3.2 Les fonctions de gestion de l'échéancier :

- **Initevent() :** allouer de la mémoire pour un évènement tout en l'initialisant.
- **ajouter_event(struct event evn) :** ajouter un évènement à l'échéancier en dernier.
- **extraire_event( ) :** extraire et supprimer de l'échéancier l'évènement avec la date la plus petite.
- **supprimer_event(num_capteur, event_type) :** supprime un évènement programme dans l'échéancier.

#### 2.3.3 Les fonctions de gestion des états des capteurs :

- **nouveau_etat_post_collision( etat_capteur) :** le changement de l'état d'un capteur après une collision.
- **nouveau_etat_emission( etat)** : le nouvel état d'un capteur au début d'émission.
- **convert_etat_capteur(etat) :** retourne l'état du capteur en chaine de caractère.
- **convert_etat_emission_to_int(etat) :** retourne l'état du capteur en entier.

#### 2.3.4 Les fonctions des traitements des évènements :

- **debut_emission(evenement)** : lance un évènement de début d'émission par un capteur.
- **fin_emission(evenement) :** lance un évènement de fin d'émission par un capteur.
- **Traitement_Event(evenement) :** traite un événement depuis.

- **Traitement_Collision(numero_victime, numero_provocateur, current_time)** : traite une collision entre un capteur victime et un capteur provocateur à un temps précis.

### 2.3.5 Les fonctions du simulateur :

- **initialisation() :** va initialiser les variables de la simulation a leur valeur initiale.
- **condition_arret() :** va retourner si le simulateur a atteint sa condition d'arrêt ou pas encore.
- **moyenne_distribution(float Tab[R]) :** va calculer la moyenne de la distribution sur R simulations ( Utile pour calculer l'intervalle de confiance ).
- **ecart_type(float Tab[R], moyenne) :** va calculer l'écart type de la distribution sur R simulations ( Utile pour calculer l'intervalle de confiance ).
- **simulateur :** la fonction qui lance le simulateur.
- **Main :** le programme principale du simulateur LORA

## 3. Interprétation et explication des courbes

Dans cette partie, nous mettons en avant les résultats de la simulation, et ceux en générant, en utilisant l'outil **Gnuplot**, des courbes et graphiques que nous allons essayer de décrire et d'expliquer ci-dessous.

Nb : Toutes les courbes et résultats que nous allons afficher ci-dessous sont pour des paramètres fixées à :

- Le nombre de capteurs **K = 5.**
- Temps moyen d'attente avant première tentative d'émission **i = 0,1**
- $\forall j \in \{1, \ldots, 7\}$, **ej = 10**
- $\forall j \in \{1, \ldots, 6\}$ ,**wj = 0.25**
- Une condition d'arrêt de **1000 messages envoyés** correctement par chaque capteur.

Sauf si les valeurs des paramètres sont écrites explicitement dans l'une des simulations ci-dessous.

### 3.1 Probabilité de collision dans les états ej en fonction du Temps :

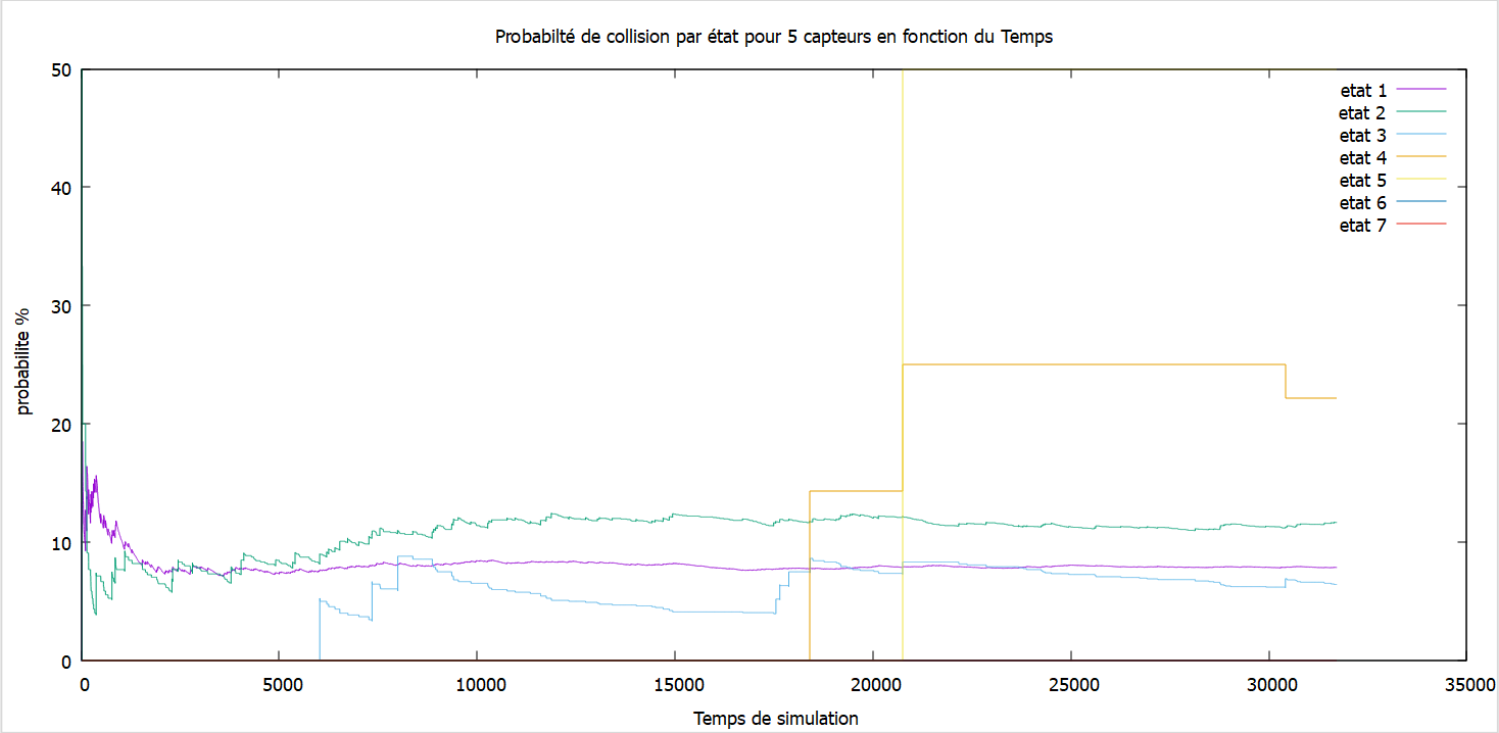
Nous commencerons avec Les premiers résultats de la simulation, et qui sont les plus importants d'ailleurs, c'est la probabilité de collision à l'émission d'un paquet dans chacun des états j d'un capteur en fonction du temps de la simulation.

Afin de pouvoir générer ces résultats, nous aurons besoin de :

- Calculer le nombre de tentatives d'envoi de paquet de chaque état à chaque fois avant de lancer un nouvel évènement de l'échéancier.
- Calculer le nombre de collisions de chaque état à chaque fois avant de lancer un nouvel évènement de l'échéancier.
- La probabilité de collision P = (le nombre de collisions) / (le nombre de tentatives) .

Voici ci-dessous les résultats de cette simulation à l'aide de graphe crée par **GNUPLOT.**

### 3.1.1 Paramètres par défaut :




#### 3.1.1.1 Description :

Le schéma représente très clairement « La probabilité de collision pour 5 capteurs en fonction du temps pour les 7 états possible du capteur » avec une condition d'arrêt de 1000 messages envoyés avec succès par chacun des 5 capteurs.

On remarque que :

- Pour les états « 1, 2, 3 », Les probabilités commencent par être très perturbées au début de la simulation pour aller se stabiliser au fur à mesure du temps vers une valeur de proche des 10%.

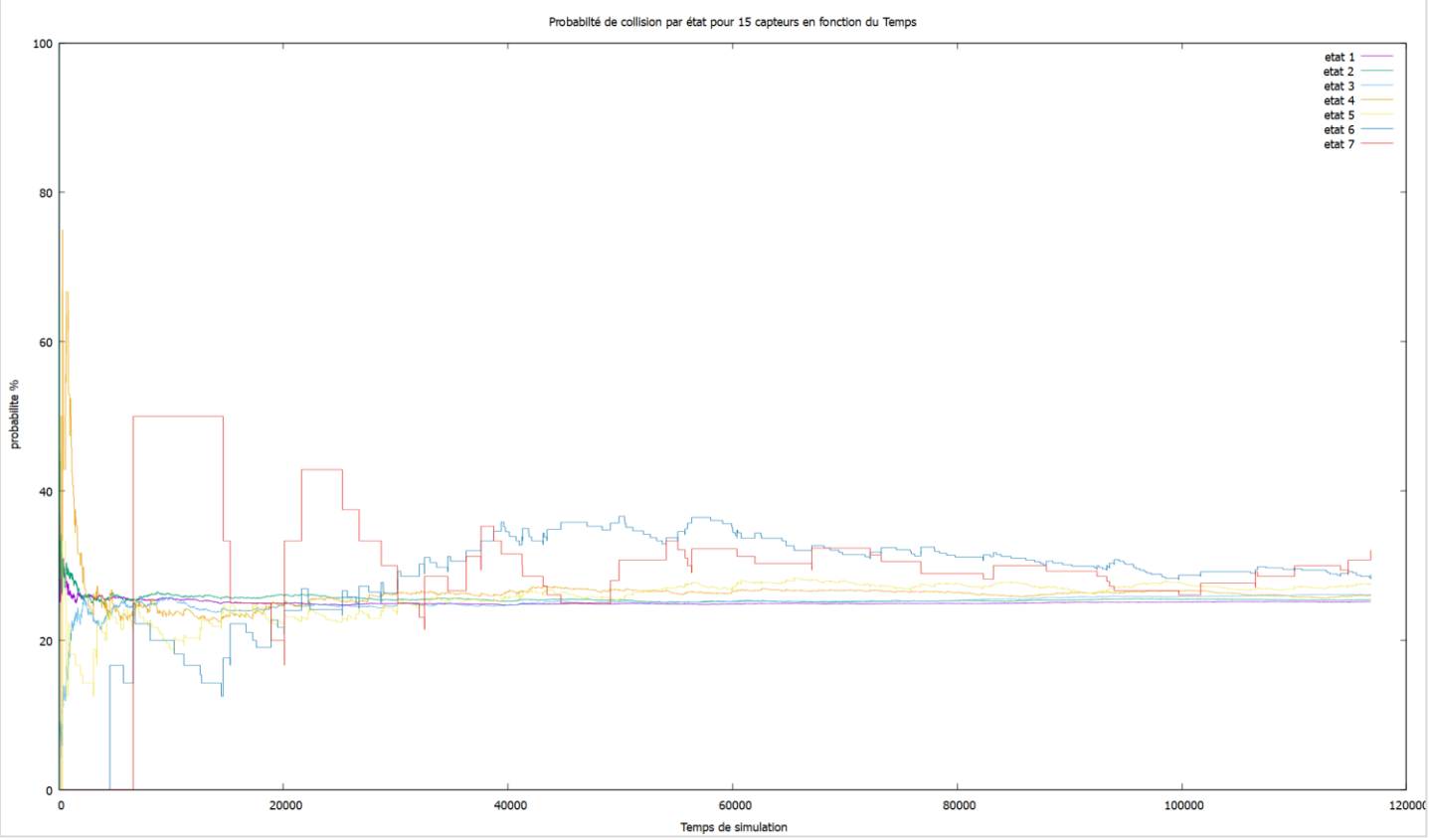
- Les états « 4,5 » Quant à eux sont extrêmement instable avec des pats allant jusqu'à 20%.
- Alors que on n'a même pas de résultats pour les états « 6 et 7 » tout simplement car le simulateur n'est jamais arrivé à une collision d'état 6 ou 7.

#### 3.1.1.2 Explication :

La simulation est très clairement encore dans un régime transitoire non stable, on ne peut pas savoir si la probabilité va accroitre, décroitre, converger vers une certaine valeur ou diverger avec les résultats ci-dessus. (On n'a même pas eu un seul échantillon de tentative d'envoie depuis l'état 6).

C'est pour cela, qu'on a décidé de refaire la simulation mais avec d'autres paramètres mieux choisis afin d'avoir des résultats plus significatifs.

### 3.1.2 Paramètres personnalisés :




#### 3.1.2.1 Description :

Le schéma représente très clairement « La probabilité de collision pour **15 capteurs** en fonction du temps pour les 7 états possible du capteur » avec une condition d'arrêt de **10000 messages envoyés avec succès** par chacun des 15 capteurs.

On a décidé d'augmenter le nombre de capteur ainsi que la condition d'arrêt pour arriver à des résultats plus significatifs que ceux de la simulation précédente.

On remarque que :

- Pour tous les états « 1, 2, 3, 4, 5, 6, 7», Les probabilités commencent par être assez perturbées au début de la simulation pour aller se stabiliser au fur à mesure du temps vers une valeur de proche des 25% pour e1, e2, e3, e4 et e5 alors que e6 et e7 sont aux alentours des 27%.

**3.1.2.2 Explication :**

Le schéma et les résultats montrent que la simulation converge vers un régime stationnaire, les probabilités de collisions se stabilisent approximativement à 25% après un certain nombre de messages envoyés.

### 3.2 Intervalle de confiance à 90% de la probabilité de collision dans l'état e2 pour 50 simulations différentes :

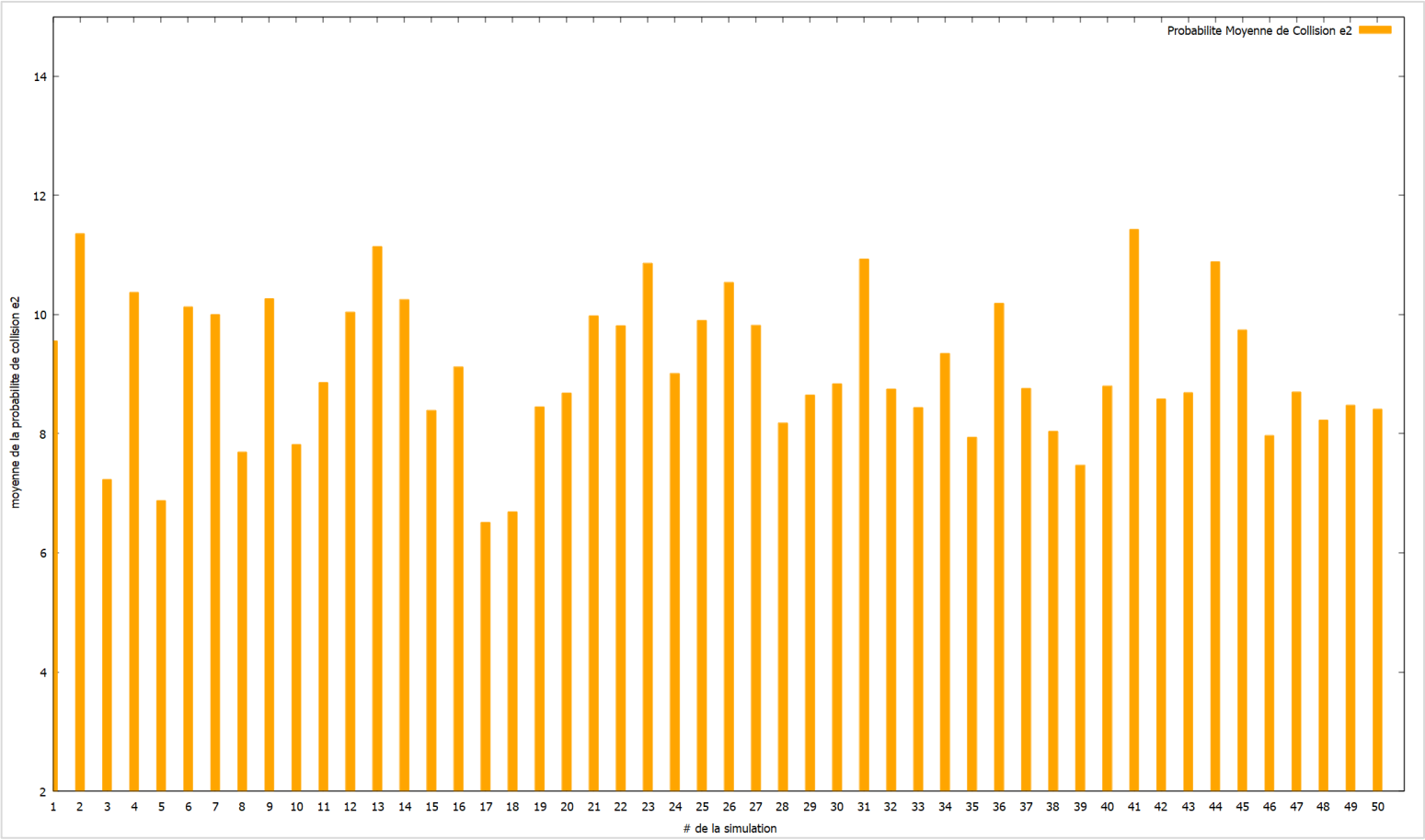
Nous allons dans ce qui suit, lancer 50 simulations différentes avec les mêmes paramètres, et nous allons nous intéresser à estimer la moyenne de distribution à partir de la moyenne des 50 échantillons de la distribution. Ou en d'autres termes à un intervalle de confiance de 90% de la probabilité de collision dans l'état e2.

Nous allons procéder comme suit :

- Dans un premier temps lancer 50 simulations différentes avec les mêmes paramètres de simulation.
- Calculer la moyenne de probabilité de collision pour e2 pour chaque simulation.
- Nous calculons la moyenne de la distribution.
- Nous calculons son écart type.
- En utilisant le Théorème central limite et comme n = 50 > 30, alors l'intervalle de confiance devient pour sigma = Ecart type, x_Barre étant la moyenne et Alpha le niveau de confiance :

$$\left[\overline{x} - t_{1-\frac{\alpha}{2}} \frac{\sigma}{\sqrt{n}} \quad , \quad \overline{x} + t_{1-\frac{\alpha}{2}} \frac{\sigma}{\sqrt{n}}\right]$$

Pour 50 simulations différentes, on obtient les résultats suivants :




### 3.2.1 Description et explication :

Le schéma représente « La probabilité moyenne de collision pour **5 capteurs** en fonction du temps pour les 7 états possible du capteur » avec une condition d'arrêt de **1000 messages envoyéS avec succès** par chacun des 5 capteurs sur **50 simulations différentes.**

Le schéma montre clairement des **moyennes incluse entre [ 6.51 ; 11.43 ] ,** en utilisant ces résultats on peut calculer **un intervalle de confiance de 90% de la probabilité de collision dans l'état e2.**

La moyenne de la distribution **xBarre donne 8.80,** et l'écart type **SIGMA donne 1.88**

Donc on peut calculer en utilisant le théorème central limite l'intervalle de confiance en question comme suit :

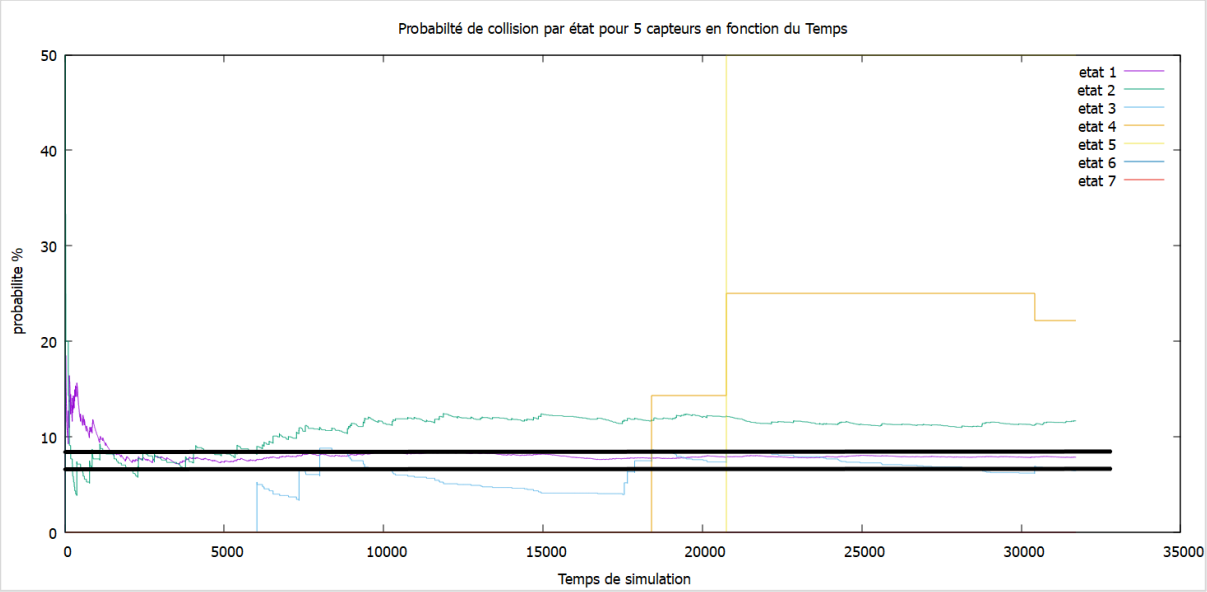
**IC = [ xBarre – T1-α/2 * Sigma / √n ; xBarre + T1-α/2 * Sigma / √n ] ……..(1)**

Et donc : en substituant les valeurs qu'on a, ainsi que **T1-α/2 = T0.95 = 1.645 de la table de la loi <u>NORMAL</u>** on obtient :

**IC** = [ 8.80 - 1.645 * 1.88 / √50 ; 8.80 + 1.645 * 1.88 / √50 ] = **[8.362 ; 9,237 ]**

Ceci veut juste dire que à 90% , la probabilité de collision à l'état deux avant les 1000 messages envoyés est comprises entre **[8.362 ; 9,237 ].**

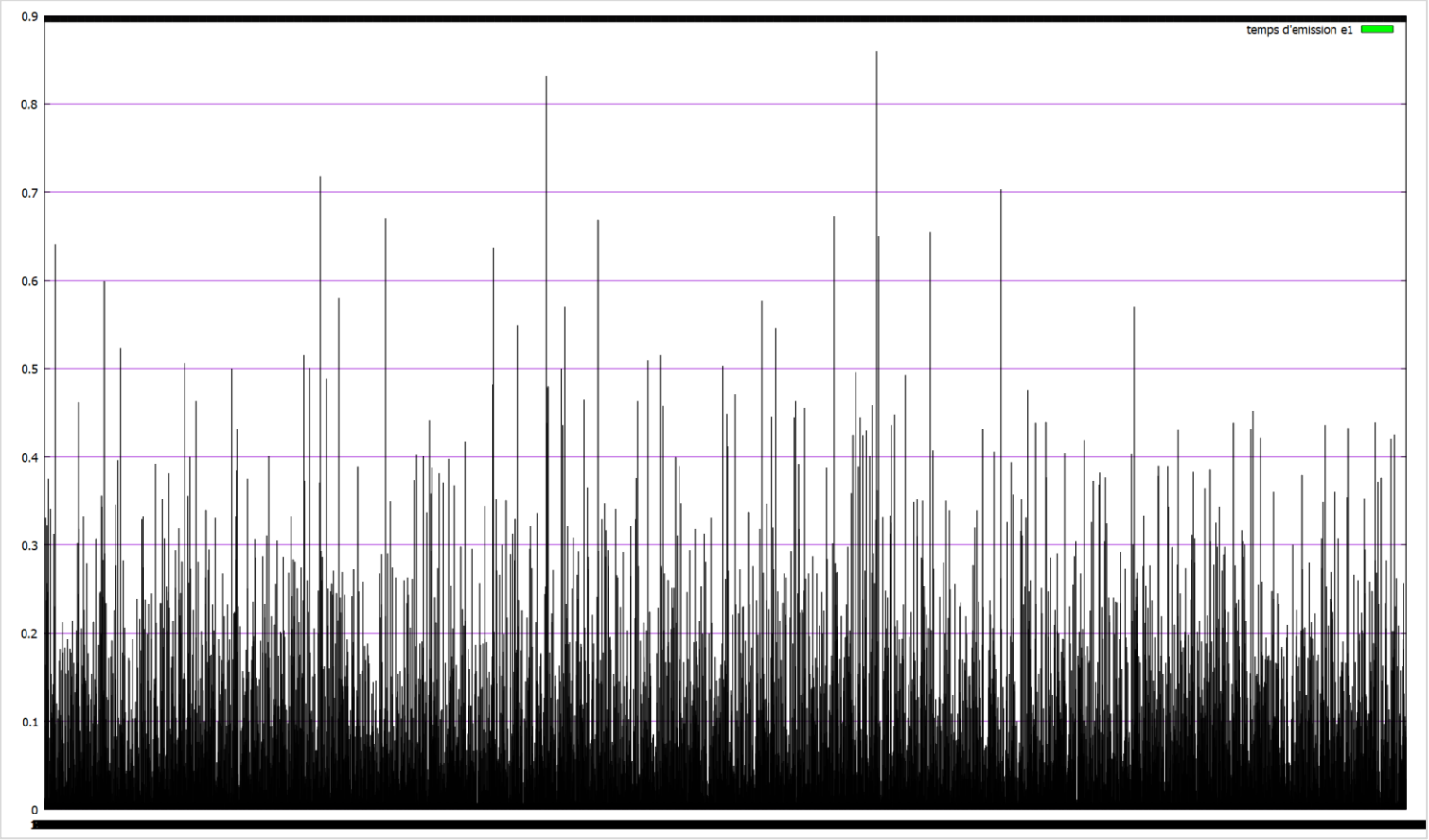
**Ci-dessous on intègre <u>l'intervalle de confiance</u> (EN NOIR) à la figure précédente :**




NB : L'intervalle de confiance semble ne pas être très précis parce que les 50 expériences ont été faites sur des échantillons crées sous condition d'arrêt d'envoi de 1000 messages par capteurs, ce qui fait que les résultats n'étaient pas très stables comme expliqué avant.

**3.3 Tracer deux histogrammes du temps d'émission observé. Le premier concerne l'état e1, le deuxième concerne l'état e2. Indiquer la moyenne du temps observé dans chaque histogramme.**

### 3.3.1 Histogramme du temps d'émission observe sur l'état e1




#### 3.3.1.1 Description et explication

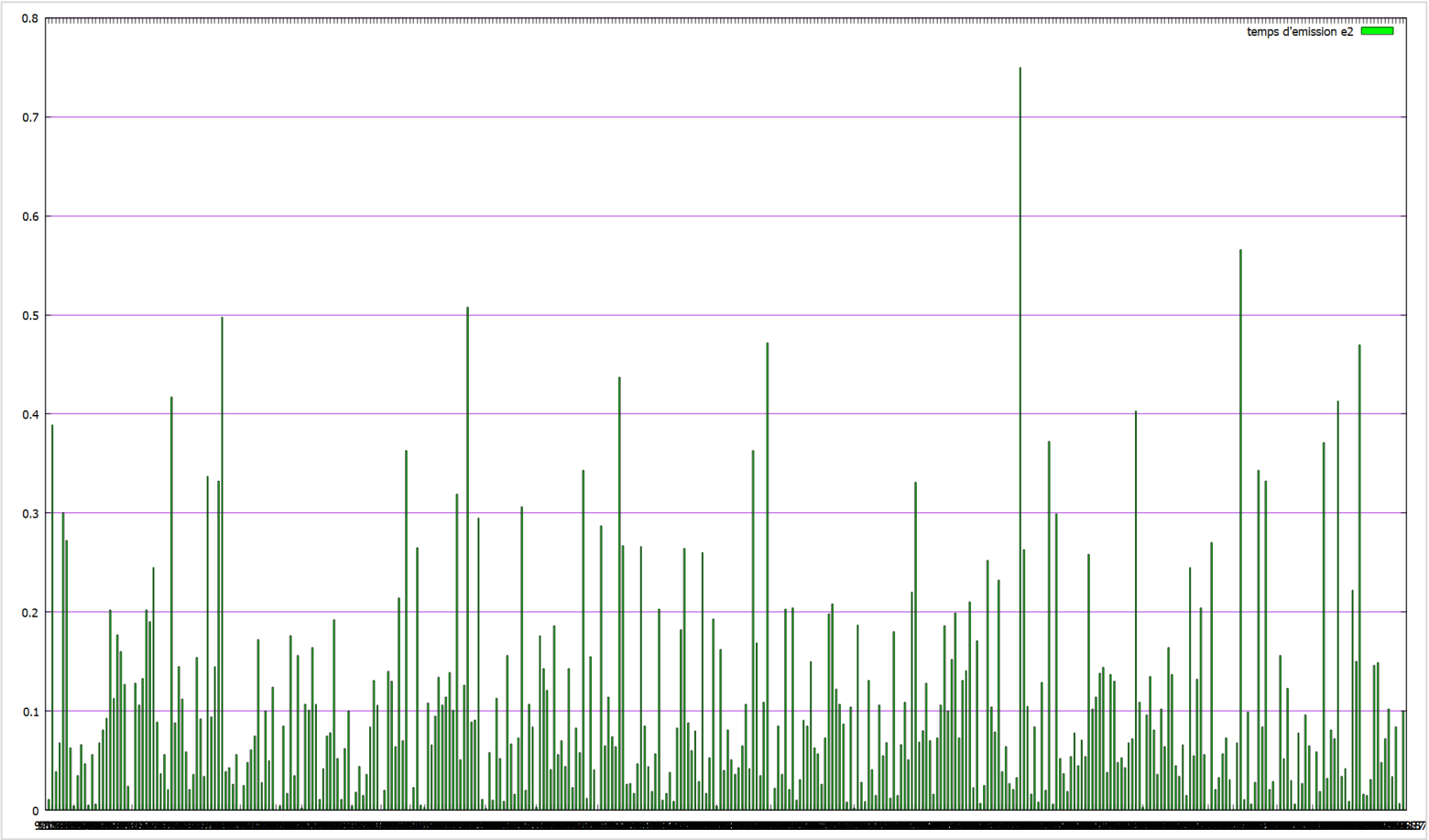
Le schéma représente « L'évolution du temps d'émission par l'état e1 au fil du temps de la simulation pour **5 capteurs** » avec une condition d'arrêt de **1000 messages envoyée avec succès** par chacun des 5 capteurs**.**

Depuis le schéma, on peut voir que le temps d'émission varie de valeurs très minimes allant à **0.05 jusqu'à des temps d'émission pouvant aller à 0.85**

Cette large marge de différence peut être expliquée par deux raisons principale :

- La première est que le temps d'émission est aléatoire à chaque émission de paquet, car La durée d'émission d'un paquet est une va qui suit une loi Expo(e)
- La deuxième et qui justifie les durée d'émission assez petite, est l'arrivé d'une collision juste après le début d'émission à e1, ceci implique l'arrêt d'émission à e1, le traitement de la collision et le transfert du capteur a l'état e2.

### 3.3.2 Histogramme du temps d'émission observe sur l'état e2




#### 3.3.2.1 Description et explication

Le schéma représente « L'évolution du temps d'émission par l'état e2 au fil du temps de la simulation pour **5 capteurs** » avec une condition d'arrêt de **1000 messages envoyéS avec succès** par chacun des 5 capteurs**.**

Depuis le schéma, on peut voir que le temps d'émission varie de valeurs très minimes allant de **0.01 jusqu'à des temps d'émission pouvant aller à 0.75**

Cette large marge de différence peut être expliquée comme celle de l'état e1 par deux raisons principales :

- La première est que le temps d'émission est aléatoire à chaque émission de paquet, car La durée d'émission d'un paquet est une va qui suit une loi Expo(e)
- La deuxième et qui justifie les durées d'émission assez petite, est l'arrivé d'une collision juste après le début d'émission à e2, ceci implique l'arrêt d'émission à e2 après un temps très petit, le traitement de la collision et le transfert du capteur a l'état e3.

### 3.3.3 Moyenne du temps observé :

Pour trouver la moyenne du temps d'émission observé à l'état e1, il faudrait connaitre le nombre de paquets émis par e1.

Car :

$$moyenne_{temps_{emission}} = \frac{Temps\ d'emission\ totale}{Nombre\ de\ paquet\ emis}$$

```
D:\Projet simulation\Lora\main.exe
Etat e1 : Nombre de packets = 4710 , temps d'emission total = 493.66
Etat e2 : Nombre de packets = 368 , temps d'emission total = 36.13
```

Donc :

- **Moyenne_e1 = 0,1048**
- **Moyenne_e2 = 0,098**

### 3.4 Probabilité de collision moyenne en fonction du nombre de capteurs k :

Dans ce qui suit, nous allons faire varier le nombre de capteurs K de 1 à 150, et pour chaque K nous allons calculer la probabilité de collision moyenne dans le réseau. Nous soulignons que cette étude est faite avec des paramètres ayant des valeurs par défaut (comme énoncé sur l'énoncé ) c'est-à-dire :

- Temps moyen d'attente avant-première tentative d'émission **i = 0,1**
- $\forall j \in \{1, \ldots, 7\}$, **ej = 10**
- $\forall j \in \{1, \ldots, 6\}$ ,**wj = 0.25**
- Une condition d'arrêt de **1000 messages envoyés** correctement par chaque capteur.

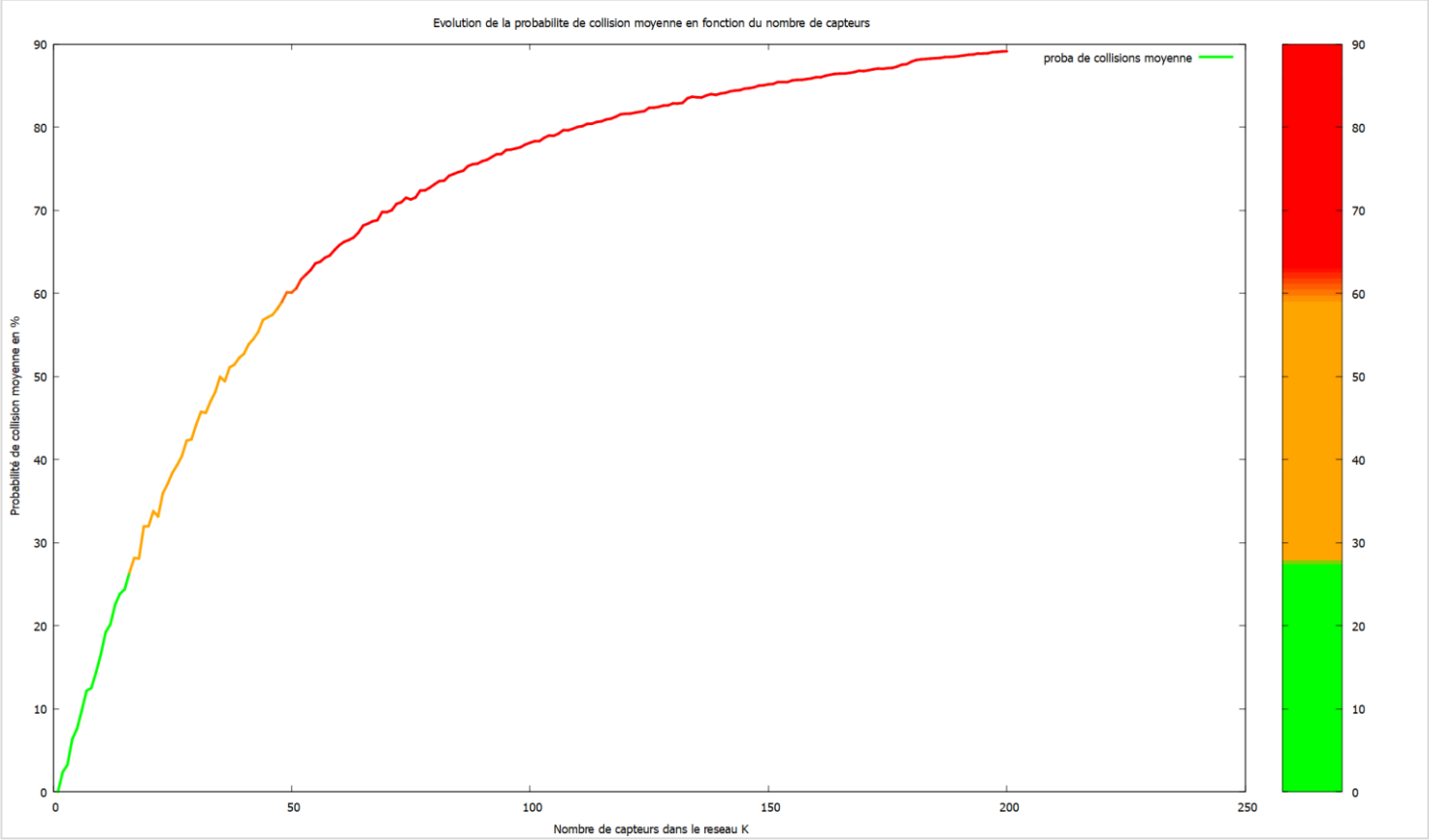
Pour calculer la probabilité moyenne de collision pour un nombre de capteurs donnée K :

- Nous calculons la probabilité de collisions à chaque instant t de la simulation et ceux en divisant le nombre de collisions total par le nombre de tentatives totales.

$$\boldsymbol{Proba_{collision(t)} = \frac{Nombre\ de\ collision(t)}{Nombre\ de\ tentatives(t)}}$$

- On multiplie chaque probabilité calculée précédemment fois sa fréquence d'apparition, c'est-à-dire, on multiplie la proba fois son temps d'apparition / le temps de la simulation totale.
- On somme tous les produits calculés précédemment pour trouver la moyenne qu'on cherche.

Voici la courbe qu'on trouve avec les paramètres et calculs expliqué Ci-dessus :




### 3.4.1 Description :

Le schéma représente « l'évolution de la probabilité de collision moyenne dans un réseau Lora en fonction du nombre de capteurs K dans le réseau qui varie de 1 à 150 » avec une condition d'arrêt de **1000 messages envoyés avec succès** par chacun des k capteurs. La courbe est associée à une palette de couleurs « Vert, Orange, Rouge » pour une meilleure représentation de la probabilité en question.

La courbe suit une croissance très rapide pour des valeurs de K très petits, c'est-à-dire , que un petit changement sur la valeur de K va entrainer un grand changement sur la probabilité de collision moyenne , et ceci plus K est petit, par contre, plus K grandit plus la courbe aura

tendance à se stabiliser, c'est-à-dire , que un changement sur la valeur de K va entrainer un changement très petit sur la probabilité de collision moyenne.

### 3.4.2 Explication :

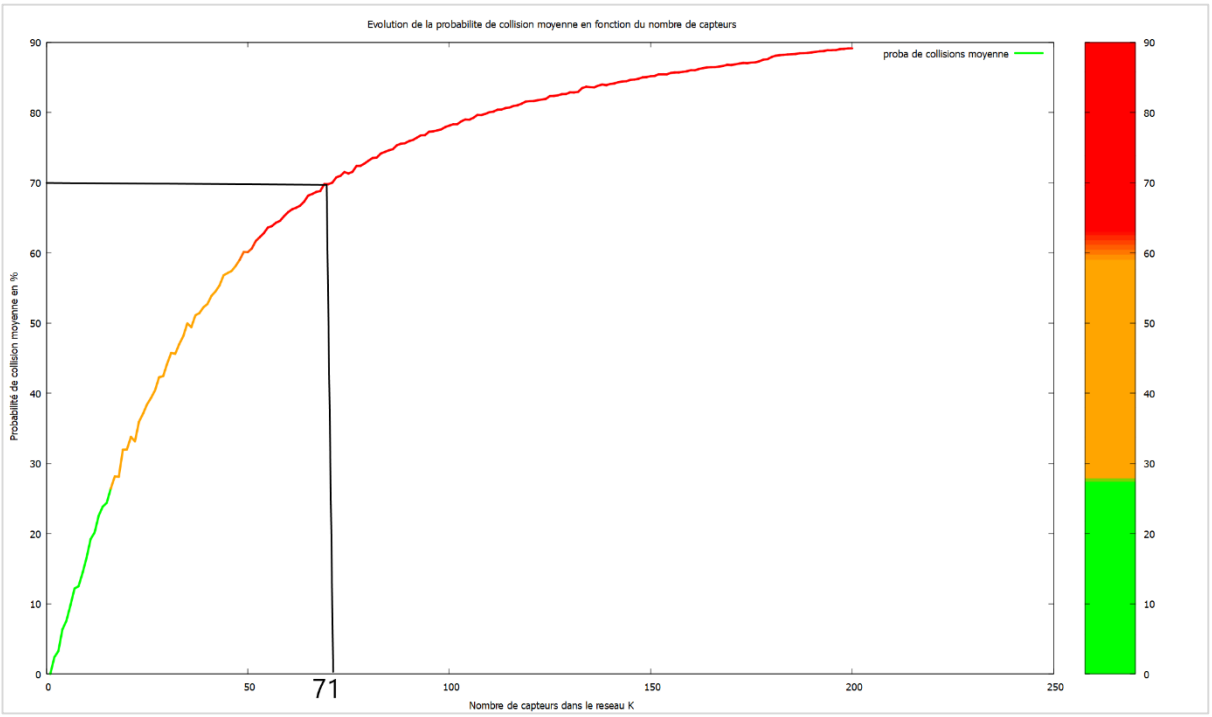
Il est tout à fait normal d'avoir 0% probabilité de collision pour un seul capteur dans le réseau, plus le nombre de capteurs augmentent, plus les collisions augmentent et donc la probabilité de même augmente de façon très forte.

À un certain moment, la tendance de croissance de la courbe diminue un peu, car pour un nombre très important de collisions et K capteurs, le fait de rajouter un capteur en plus, va entrainer peu de collisions en plus par rapport au nombre de collision totale déjà accumule avec les K capteurs précédemment, ce qui fait que la probabilité de collision avec (K+1) va croitre mais juste un tout petit peu. Jusqu'à peut-être se stabiliser à 99,99% pour K allant à l'infinie.

## 4. Réponse aux questions

### 4.1 À partir de quelle valeur de K la probabilité de collision dépasse les 70% ? Justifiez

En lisant la courbe précédente de « l'évolution de la probabilité de collision moyenne dans un réseau Lora en fonction du nombre de capteurs K dans le réseau qui varie de 1 à 150 » , la probabilité de collision franchit la barre des 70% à partir de **71 capteurs dans le réseau.**

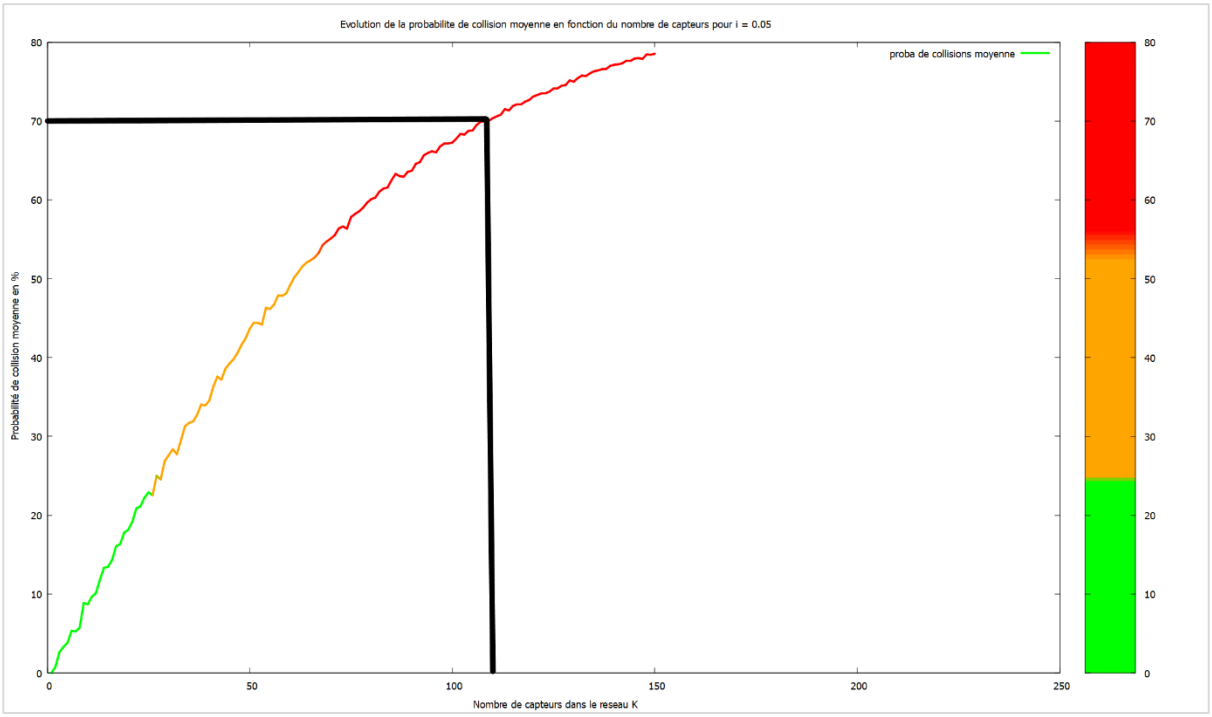


### 4.2 Que proposeriez-vous pour remédier à ce taux très élevé de collisions ? Justifiez

Afin de diminuer le taux de collisions qui est très élevé, nous proposons les 3 solutions suivantes :

#### 4.2.1 Augmenter le temps d'attente i avant la première tentative d'émission :

Pour cela il suffirait de diminuer la valeur de i de 0.1 à une valeur plus petite, nous prenons 0.05 comme exemple et relançons la simulation, nous obtenons la courbe suivante :



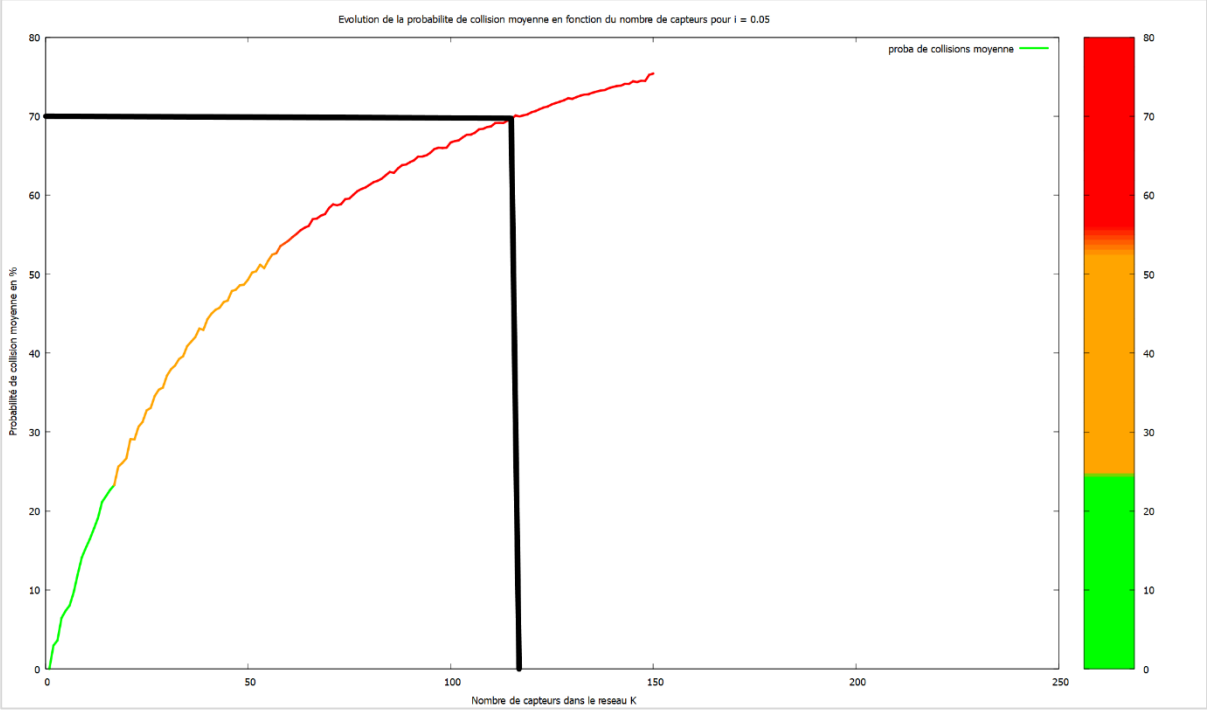

Depuis la courbe on retrouve que le taux atteint une valeur de 70% à partir de 114 capteurs dans le réseau, et donc on a réussi à diminuer la probabilité de collision moyenne qui était de 70% à 71 capteurs auparavant.

Cependant cette solution a ses inconvénients, bien évidemment augmenter le temps d'attente avant la première tentative diminue très clairement la probabilité d'une collision, mais aussi augmente le temps total d'émission des paquets car un délai d'attente légèrement augmente sur 10000 paquets envoyés entrainent une grande différence en termes de temps.

#### 4.2.2 Augmenter le temps d'attente w avant réémission après une collision :

La deuxième solution va toucher à la façon de traiter les collisions, le fait d'augmenter le temps d'attente avant réémission du paquet qui a subi une collision va permettre un meilleur taux de collision moyen.

La courbe ci-dessous provient d'une simulation avec w = 0.1 au lieu de 0.25.




Depuis la courbe on retrouve que le taux atteint une valeur de 70% à partir de 116 capteurs dans le réseau, et donc on a réussi à diminuer la probabilité de collision moyenne qui était de 70% à 71 capteurs auparavant.

Cependant, et comme pour la proposition précédente, cette solution vient avec le même désavantage, une probabilité de collision inferieur coute un temps de transfert des paquets plus grand.

### 4.2.3 <u>Diminuer le temps d'émission d'un paquet :</u>

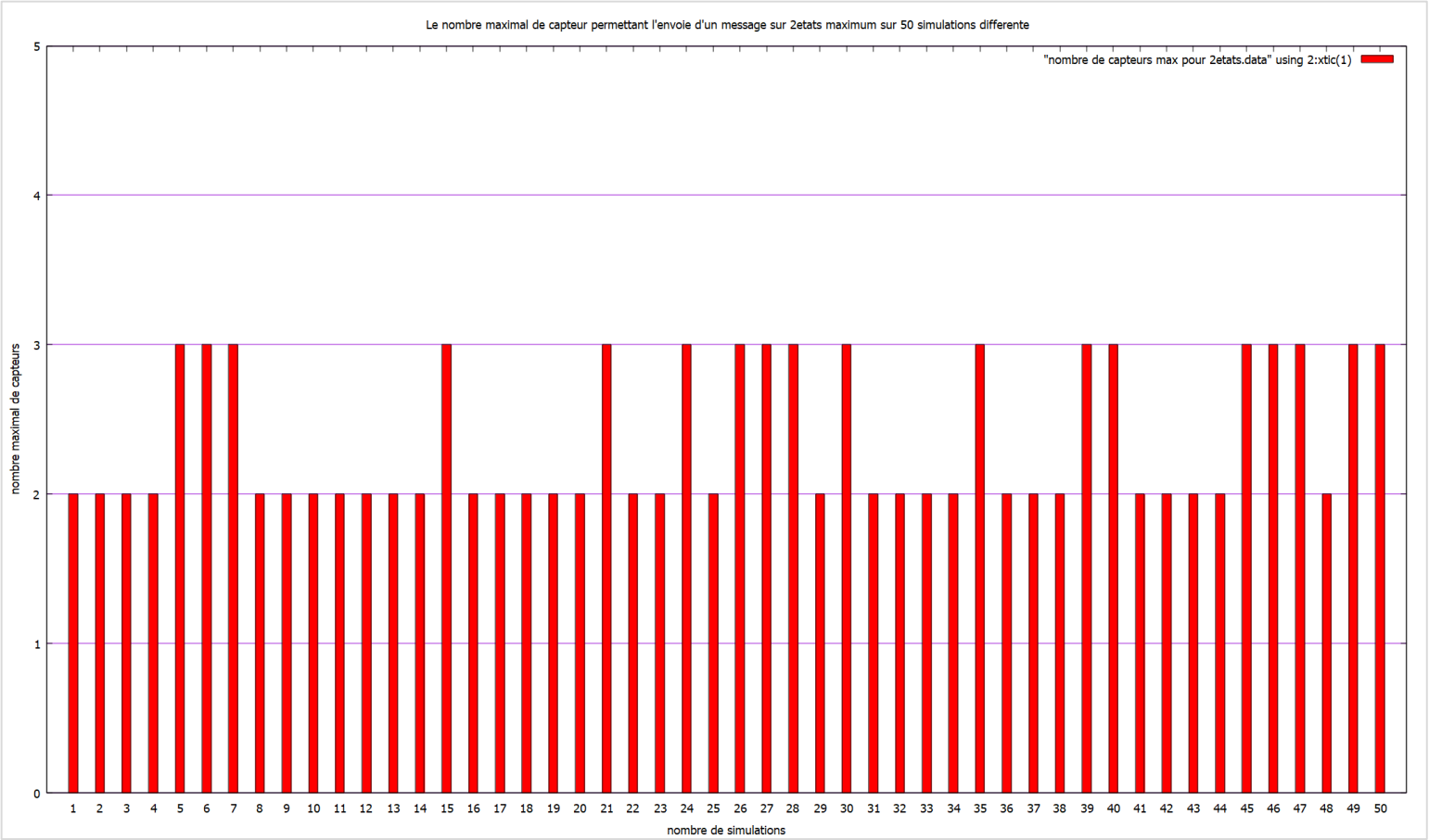
Bien sûr, cette solution est évidente, diminuer le temps d'émission des paquets va très certainement diminuer le taux de collision dans le réseau, cependant, la diminution de ce temps est une **solution matériel et non pas logiciel**, en effet, pour permettre une émission de données plus rapide il faut utiliser des technologies plus sophistiquée en terme de communication **réseau (chose qui pourrais couter plus cher, mais qui sera certainement plus performant)**

**4.3 Combien de capteurs faut-il utiliser, afin d'assurer que dans 90% des cas, un message réussisse son émission en deux tentatives maximums ?**

Commençons par comprendre ce qui est attendu, s'assurer que dans 90% des cas un message réussisse son émission en deux tentatives maximums est équivalent à dire que la probabilité de collision à l'état 2 est inférieur ou égale 10%, ou que 90% des messages qui ont été envoyé ont été envoyé par l'état 1 ou 2.

Pour répondre à cette question, nous avons lancé 50 simulations différente , et pour chaque simulation, nous allons calculer le nombre de capteurs maximum que nous pouvons mettre dans le réseau sous condition que un message sois envoyé au max dans deux tentatives.

Ci-dessous les résultats que nous avons trouvés :




Des expériences précédentes, on a trouvé les résultats suivants :

- Kmax = 3 avec une proportion de 18 fois sur 50 =====> f = 0.36
- Kmax = 2 avec une proportion de 32 fois sur 50 =====> f = 0.64

Comme on cherche une estimation à 90%, Nous allons chercher l'intervalle de confiance à 90% des proportions précédentes.

**Pour K = 3 :** Ic = [0.36 - 1,645 $\sqrt{\frac{0.36*0.64}{50-1}}$ , 0.36 - 1,645 $\sqrt{\frac{0.36*0.64}{50-1}}$ ]

Donc **IC3 = [ 0.29 , 0.428 ]**

Il est très clair à partir de l'intervalle de confiance précèdent que la probabilité qu'un message soit envoyé à deux tentatives maximum avec 3 capteurs dans le réseau est très faible (Il varie entre 29% et 42%) , qui est très loin de la probabilité recherche qui est de 90%.

Et donc le nombre de capteurs maximal qu'on peut prendre pour être sur que à 90% des cas un message soit transmis en deux tentatives maximum est de **K = 2 (on a eu 100% de réussite sur une population de n = 50 échantillons ou simulations)**

## 5. Conclusion

La réalisation de ce projet était une chance de mettre en pratique les concepts théoriques vu en cours durant tout le semestre, que ce soit dans l'aspect de la programmation de simulateurs, génération des courbes et graphiques des résultats du simulateur ou même l'interprétation et l'analyse des résultats. Ça traitait aussi un sujet très intéressant qui est les réseaux LORA ou nous avons acquis des connaissances sur cette technologie qui est la référence des communications dans le monde de l'IOT.

Nous avons aussi certes rencontré plusieurs difficulté dans la réalisation de ce travail, notamment dans la compréhension de quelques questions qui on trouve n'était pas toujours très clair, mais nos enseignants étaient toujours présent, attentifs et très réactifs à nos questions avec des réponses très rapide et précises à nos besoins. Sur ce nous remercions les enseignants du modules pour leur implication et les efforts fournies avec nous.

---

# FIN