
Université Ferhat Abbas –Sétif 1

Faculté des Sciences

Département d'informatique

# Evaluation des Performances des Systèmes

*Master 1 : Réseaux et Systèmes Distribués (RSD)*

**Dr. Sarra CHERBAL**

Maitre de conférences classe A en Informatique

# Table des Matières

# Liste des Figures

# Préambule

Les systèmes et réseaux informatiques ont besoin de méthodes pour l'étude de leur comportement. Ces systèmes traitent des concepts complexes avec des demandes de service aléatoires, par conséquent, les méthodes probabilistes et statistiques sont couramment employées à des fins d'évaluation des performances et de la fiabilité. Les modèles analytiques les plus couramment utilisés sont les chaînes de Markov. Les progrès des méthodes automatisées d'espace d'état ont entraîné une utilisation intensive des chaînes de Markov dans l'analyse des performances et de la fiabilité.

Une chaîne de Markov se compose d'un ensemble d'états et d'un ensemble de transitions étiquetées entre les états. Un état peut modéliser diverses conditions dans le système étudié, ex. il peut s'agir du nombre de taches en attente d'utilisation de chaque ressource, le nombre de tâches en cours d'exécution, etc. Après un séjour dans un état, la chaîne de Markov fera une transition vers un autre état. Ces transitions sont étiquetées soit avec des probabilités de transition (dans le cas de chaînes de Markov à temps discret), soit avec des taux de transition (dans le cas de chaînes de Markov à temps continu). Le régime stationnaire des chaînes de Markov peut être étudié à l'aide d'un système d'équations linéaires avec une équation pour chaque état. La solution de ces équations donne des probabilités d'état de la chaîne de Markov à partir desquelles les mesures de performance souhaitées peuvent être facilement obtenues. Les mesures de performance qui présentent généralement un intérêt comprennent le débit, l'utilisation des ressources, la probabilité de perte et le retard (ou temps de réponse).

L'objectif de ce cours est d'exploiter les principales techniques de modélisation probabiliste qui sont les files d'attente Markoviennes pour l'analyse des performances des systèmes informatiques comme un réseau cellulaire, un système de transmission de paquets, un protocole d'accés aléatoire et un protocole de réseaux Wi-fi.

De l'autre côté, des modèles d'évaluation de performances sont basés sur la simulation à événements discrets (SED). La SED peut être appliquée à un système dans sa phase de conception pour vérifier qu'il répond aux exigences de conception et pour effectuer les compromis nécessaires. L'utilisation de la simulation est présentée dans la partie des Travaux Pratiques de ce cours.

# 1 Rappels des lois de probabilités

## 1.1 Processus de poisson

Le processus de Poisson sert à modéliser l'occurrence d'évènements successifs, par exemple : Le nombre d'appels reçus par un standard téléphonique dans une période de temps, Le nombre de voyageurs se présentant à un guichet dans la journée, etc.

On dit qu'une variable aléatoire dénombrable (discrète) $X$, à valeurs dans $N$, suit une loi de poisson de paramètre $\lambda$ ($\lambda > 0$), si:

$$\text{pour tout: } k \in N, P[X = k] = e^{-\lambda} \times \frac{\lambda^k}{k!} \quad X \sim P(\lambda)$$

- $P[X = k]$ : La probabilité d'avoir $K$ fois l'évènement $A$ pour une période de $T$
- $\lambda$ : le nombre moyen d'occurrence du phénomène observé pendant la durée donnée. Pour une période de $T$, un évènement se produit en moyenne $\lambda$ fois
- L'espérance et la variance : $E[X] = \lambda, V[X] = \lambda$.

**Exemple**
Un magasin reçoit généralement un nombre de deux clients par heure. $X_t$ est une variable aléatoire présentant le nombre de clients arrivant au magasin. $X_t$ suit la loi de poisson de paramètre $\lambda = 2$ clients/heure.

**Exercice**
Le taux de naissance au Canada est d'environ 43 naissances par heure.

- Quelle est la probabilité que durant les 5 prochaines minutes il y ait 3 naissances ou plus?
- Quelle est la probabilité que 10 minutes s'écoulent sans aucune naissance?

## 1.2 La loi exponentielle

La loi exponentielle sert à modéliser les temps et les durées :

- La durée de service,

- La durée d'un appel téléphonique,
- Le temps d'inter-arrivée
- ... etc.

$X_t$ est une variable aléatoire suivant la loi exponentielle de paramètre $\lambda$ et d'un espérance $d$ (le temps moyen ou la durée moyenne). Les probabilités de $X_t$ par rapport au temps $t$ sont calculées comme suit :

$$P(X \geq t) = e^{-\lambda t}$$

$$P(X \leq t) = 1 - e^{-\lambda t}$$

$$P(t_1 \leq X \leq t_2) = e^{-\lambda t_1} - e^{-\lambda t_2}$$

**Exemple**

Un magasin reçoit 1 client chaque $\frac{1}{4}$ heure (le temps moyen d'inter-arrivée des clients). Ce temps suit la loi exponentielle de paramètre $\lambda$.

$\rightarrow \lambda = \frac{1}{d}$. Tel que $d$ est la durée moyenne d'inter-arrivée.

Dans cet exemple : $\lambda = 4$ clients/heure.

On dit que :

- Le nombre de clients suit une loi de poisson de paramètre $\lambda$.
- Le temps d'inter-arrivée de ces clients suit une loi exponentielle de paramètre $\lambda$.
- $\lambda$ est le taux d'arrivées, d'entrées, de naissances ... (plusieurs nominations peuvent être appliquées)

# 2 Rappels sur les Files d'attente Markoviennes

## 2.1 Processus de Markov à temps continu

- Dans les chaînes de Markov à temps discret, les instants de temps sont discrets (0,1,2,...).
- Maintenant, nous allons analyser des situations où les observations se font de façon continue plutôt qu'à des moments discrets.
- $X_t$ = l'état du système au temps $t$
- Les points de changement d'états sont des points aléatoires dans le temps.

## 2.2 Processus de naissance et de mort

De base, c'est pour rendre compte de l'évolution de la taille d'une population, les processus de naissance et de mort sont des processus de Markov continus ($T \in R+$), à valeurs dans $E \in N$ tels que:

- Les seules transitions non négligeables possibles à partir de $n$ sont soient celles vers $n+1$ ou vers $n-1 \rightarrow$ Processus de saut.
- Pour les taux de transition dans un graphe de Markov, les arcs vers la droite représentent les taux de naissance, et ceux vers la gauche les taux de mort.
- La naissance : l'apparition d'individus au sein d'une population suivant une certaine loi de probabilité.
- Dans le cas de mort : quand l'état est 0, on ne parle pas de mort.

Dans le graphe markovien de processus de naissance et de mort :

- L'état présente l'effectif de population (de 0 à l'infinie)
- S'il y a une naissance on passe à l'état suivant
- S'il y a une mort on revient sur l'état précédent
- S'il n'y a ni naissance ni mort, on reste sur l'état actuel, avec une probabilité négligeable.

## 2.3 Les files d'Attente Markoviennes (F.A.M)

Les files d'attente peuvent être considérées comme un phénomène caractéristique de la vie contemporaine. On les rencontre dans les domaines d'activité les plus divers: (guichet de poste, trafic routier, central téléphonique, atelier de réparation,...).

L'étude mathématique des phénomènes d'attente constitue un champ d'application important des **processus stochastiques**.

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

Si un poste de service est libre, le client qui arrive se dirige immédiatement vers ce poste où il est servi, sinon, il prend sa place dans une queue d'attente dans laquelle les clients se rangent suivant leur ordre d'arrivée.

Un système de file d'attente comprend (Figure 2.1):

- Un espace de service avec une ou plusieurs stations de service montées en parallèle,
- Un espace d'attente dans lequel se forme une éventuelle file d'attente.

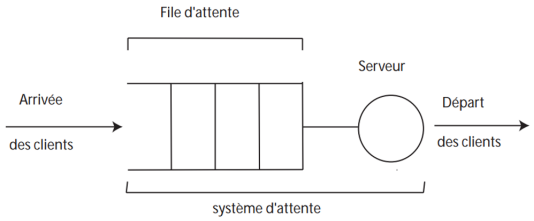


Figure 2.1: Un système de file d'attente

### 2.3.1 Les caractéristiques d'une F.A.M

Une file d'attente est décrite par plusieurs éléments:

1. Les instants d'arrivée des clients: sont en général aléatoires. Certaines hypothèses sur leurs lois :
    - Il n'arrive qu'un client à la fois.

- Les temps d'inter-arrivée sont des variables aléatoires indépendantes et de même loi. Cette hypothèse simplifie notablement l'étude de système.
- La loi la plus utilisée pour ce temps c'est « la loi exponentielle ». Dans ce cas, le modèle d'arrivée des clients est un processus de Poisson (évidement d'autres lois peuvent se présenter).

2. Les durées de service sont supposées être des variables aléatoires de même loi (ex. loi exponentielle) et indépendante du processus d'arrivée.

3. Le nombre de serveurs : est évidemment un paramètre important du modèle.

4. La longueur max de la file d'attente : c'est logique de supposer dans certains cas que la file d'attente puisse être d'une longueur infinie, aussi longue que l'on veut. Cependant dans d'autres cas, la longueur est limitée et lorsque cette limite est atteinte, un client arrivant ne peut entrer dans la file, donc il repart.

5. La discipline de service : le plus souvent les clients sont servis dans leur ordre d'arrivée, ce qu'on appelle la discipline FIFO (First In First Out). Mais d'autres disciplines pourraient être utilisées comme par exemple, servir en priorité certains types de clients, ceux demandant un service de courte durée.

### 2.3.2 Classification des systèmes d'attente

Pour la classification des systèmes d'attente, on recourt à une "notation symbolique" dite notation de Kendall, comprenant en général les 4 symboles : A/ S/ s/ K

La signification de ces symboles :

Loi d'interarrivée / Loi de service / nombre de serveurs / Longueur maximale de la file

- Les lois d'interarrivées et les lois de services sont notées symboliquement :
  - M pour une loi exponentielle (M pour Markov),
  - D pour une loi déterministe (v.a. constante),
  - U pour une loi uniforme,
  - G pour une loi quelconque (G pour générale).
- **s** : c'est le nombre de serveurs (puisque la station peut disposer de plusieurs serveurs en parallèles). Quand un client arrive à la station et il y a un serveur

libre, il entre instantanément en service, sinon, si tous les serveurs sont occupés, le client se place dans la file en attente de la libération d'un serveur. La plupart du temps les serveurs sont supposés identiques avec la même distribution et indépendants les uns des autres.

- **K** : c'est la capacité de système de file d'attente incluant le ou les clients en service.

Par exemple, une file M/M/s/$\infty$ signifie que :

- le flot d'arrivée des clients est poissonnien,
- la loi de temps de services est exponentielle,
- il y a « s » serveurs
- et la capacité de la salle d'attente est illimitée.
  PS: Lorsqu'on ne spécifie pas le dernier paramètre celui-ci est infini.
- Sauf avis contraire la discipline de service est FIFO.

### 2.3.3 Analyse mathématique

L'étude mathématique d'un système d'attente se fait le plus souvent par l'introduction d'un processus stochastique approprié $(X_t)$ $t \geq 0$

Tel que: $X_t$ est une v.a. qui présente le nombre de clients se trouvant dans le système au temps $t$.

Pour cela, on se ramènera généralement à un processus de Markov dont on déterminera le graphe des taux de transition. Pour déterminer la distribution stationnaire, on écrira, en chaque point du graphe, les équations de balance ("taux entrant égal au taux sortant") et $\sum_{i=0}^{\infty} p_i = 1$.

Le régime stationnaire du processus est défini par :

- $p_n = P(X = n)$, tel que : $p_n$ est la probabilité d'être dans l'état $n$ (i.e. d'avoir un nombre de client égal à $n$).
- $X_t$ admet une mesure stationnaire si et seulement si le paramètre $\lambda$ du processus de Poisson des Arrivées est strictement inférieur au paramètre $\mu$ de la loi exponentielle des services. $(\lambda < \mu)$

- La file d'attente se stabilise s'il n'arrive pas trop de clients pour saturer l'offre de service,

### 2.3.4 Caractéristiques d'exploitation du système

A partir de la distribution stationnaire du processus $(X_t)$ $t \geq 0$, on pourra obtenir d'autres caractéristiques d'exploitation du système telles que :

- Le nombre moyen $L$ de clients dans le système;
- Le nombre moyen $Lq$ de clients dans la queue d'attente ;
- La durée d'attente moyenne $Wq$ d'un client ;
- La durée de séjour moyenne $W$ dans le système (attente + service) ;

## 2.4 Étude de cas M/M/1

Considérant un guichet ou un serveur où:

- Les arrivées des clients à ce guichet suivent un processus de Poisson d'intensité $\lambda$.
- Le temps de service pour chaque client est une v.a. de loi exponentielle de paramètre $\mu$. Toutes les v.a. qui interviennent sont supposées être indépendantes.
- Les clients se mettent en file d'attente et sont servis selon leur ordre d'arrivée (discipline FIFO).
- La capacité de la file d'attente est illimitée.

C'est un processus de sauts à valeurs dans $N$. Quand un client arrive, le processus saute de $+1$ et quand un client s'en va à la fin de son service, le processus saute de $-1$.

$\rightarrow$ L'espace d'états $E$ est infini: $E = \{0, 1, 2, ...\}$.

La file peut être considérée comme un processus de naissance et de mort, pour lequel :

$$\lambda_n = \lambda$$

$$\mu_n = \begin{cases} \mu & si\ n \neq 0 \\ 0 & si\ n = 0 \end{cases}$$

Tous les paramètres de performances sont calculés dans le cas où la file est stable ($\lambda < \mu,\ i.e.\ u < 1$)

$\rightarrow$ pour le régime stationnaire de la file.

## 2.5 Étude de cas M/M/S

Ce système comporte $s$ serveurs indépendants. On conserve les hypothèses :

- processus d'arrivée des clients poissonien de taux $\lambda$
- et temps de service exponentiel de taux $\mu$ (pour chacun des serveurs).
- L'espace d'états $E$ est, comme pour la M/M/1 infini: $E = \{0, 1, 2, ...\}$.

On a un processus de naissance et de mort de taux :

$$\begin{cases} \lambda_n = \lambda \\ \\ \mu_n = \begin{cases} 0 & si\ n = 0 \\ n\mu & si\ 0 < n < S \\ S\mu & si\ n \geq S \end{cases} \end{cases}$$

La condition de stabilité est ici $\lambda < S\mu$ et exprime le fait que le nombre moyen de clients qui arrivent à la file par unité de temps doit être inférieur au nombre moyen de clients que les serveurs de la file sont capables de traiter par unité de temps.

# Utilisation des files d'attente Markoviennes pour l'évaluation des systèmes informatiques

## 3 Évaluation d'un Réseau cellulaire avec blocage

### 3.1 Concept cellulaire

Un système de **radiotéléphonie** utilise une **liaison radioélectrique** entre le terminal portatif (mobile station MS) et le réseau téléphonique. **La liaison radio** entre le téléphone mobile et le réseau doit être de qualité suffisante, ce qui nécessite la mise en place d'un ensemble de **stations de base (BTS)** sur l'ensemble du territoire que l'on souhaite couvrir, de telle sorte que le terminal soit toujours à moins de quelques kilomètres de l'une d'entre elles.

- Le principe consiste à diviser une région en un certain nombre de cellules desservies par un **relais radioélectrique** (la station de base BS).
- Ces cellules doivent être contiguës sur la surface couverte.
- L'hexagone est la forme régulière qui ressemble le plus au cercle et que l'on peut juxtaposer sans laisser de zones vides (Figure 3.1).

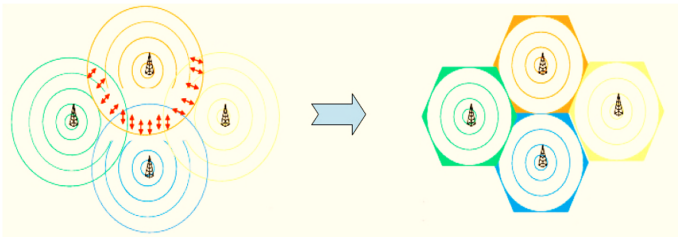

Figure 3.1: La forme des cellules

- Toutefois, la réalité du terrain est bien différente de ce modèle théorique, notamment en zone urbaine où de nombreux obstacles empêchent une propagation linéaire (Figure 3.2).

### 3.2 Problématique du système

- L'une des métriques d'évaluation importantes dans un réseau cellulaire est : la probabilité de blocage

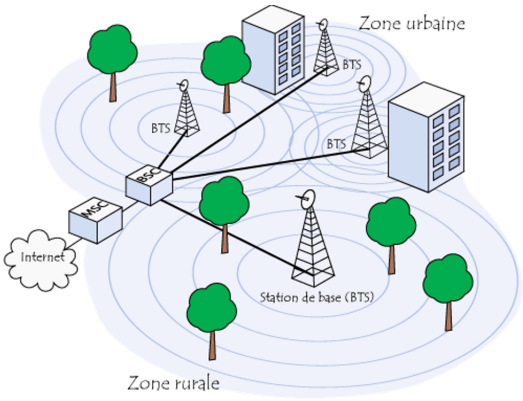


Figure 3.2: Un réseau cellulaire dans une zone urbaine et une zone rurale

- Le réseau bloque les appels parce que le nombre de canaux est limité.
- Considérant un système cellulaire avec une seule station de base qui sert les appels téléphoniques entrants et sortants (Figure 3.3).
- Il y a un nombre limité de canaux sans fils servant ces appels téléphoniques pour une cellule (ex. 6 canaux dans Figure 3.4).

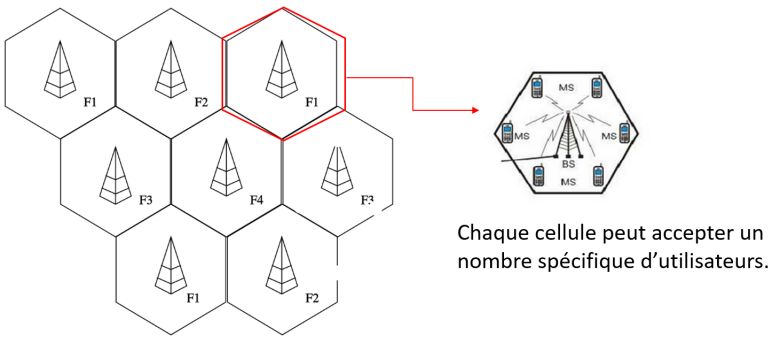


Figure 3.3: Une cellule avec un nombre spécifique d'utilisateurs

- Un canal peut être utilisé par un seul appel à la fois (ex. pour effectuer/recevoir un appel)
- Si tous les canaux sont entrain d'être utilisés, lors d'un nouvel appel, l'utilisateur reçoit un signal occupé et donc il doit essayer de se connecter ultérieurement.
- Une fois la connexion établie, le canal n'est plus accessible aux autres utilisateurs jusqu'à ce que l'utilisateur actuel mette fin à la connexion en raccrochant.
- La société de ce système cellulaire a pour politique de contrôler la probabilité de blocage en dessous de 1%.

- Cependant, une enquête initiale sur le système révèle que les clients des zones ont connu un blocage d'appels plus fréquent que ce seuil. Il s'avère que la population de la région n'a cessé d'augmenter.

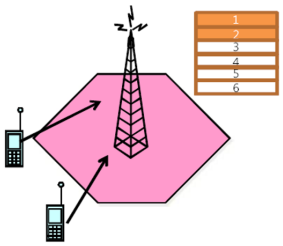


Figure 3.4: Un système cellulaire avec deux appels téléphoniques en cours.

## 3.3 L'objectif de l'étude

- L'objectif de l'étude de ce système est de réduire la probabilité de blocage.
- Afin d'atteindre cet objectif, on décide de réduire la taille des cellules dans la zone pour réduire la probabilité de blocage.
- Avec une taille de cellule plus petite, la probabilité de blocage diminue à mesure que le taux d'appels entrants est réduit.
- Mais comment sélectionner le rayon de cellule réduit? Pour répondre à la question, on développe un modèle analytique de ce système.

## 3.4 Modélisation

Quels sont les facteurs importants qui affectent la probabilité de blocage ?

- Le taux d'arrivée des appels,
- La durée moyenne des appels,
- Le nombre de canaux disponibles.

Nous pouvons collecter les informations sur ces facteurs à partir de données historiques.

### 3.4.1 L'étude de système

L'étude de système peut se faire par **simulation** ou par **un modèle analytique**. Afin de développer un modèle analytique de nombreuses **hypothèses** simplificatrices sont nécessaires.

### 3.4.2 Hypothèses

Nous faisons les hypothèses suivantes:

- La durée d'une conversation téléphonique est distribuée de façon exponentielle avec une moyenne de $1/\mu$. (**loi exponentielle** de paramètre $\mu$)
- Le temps entre les appels téléphoniques entrant au système est distribué de façon exponentielle avec une moyenne de $1/\lambda$, même lorsque tous les canaux sont occupés. (**loi de poisson** de paramètre $\lambda$)

### 3.4.3 Chaine de Markov à temps continu (CTMC)

Les hypothèses présentées sont essentielles pour développer un modèle CTMC.

- La valeur moyenne des durées d'appel peut être obtenue à partir de données historiques et nous supposons en outre que les durées d'appel sont distribuées de façon exponentielle pour créer un modèle CTMC.
- La deuxième hypothèse est une autre façon de dire que les arrivées d'appels forment un processus de Poisson de taux $\lambda$.

**États**
Un modèle CTMC est déterminé par l'ensemble d'états $E$ et les taux de transitions. Dans le modèle du système cellulaire (Figure 3.5), l'espace d'état est:

$$E = 0, 1, 2, ..., K-1, K$$

Tel que $K$ est le nombre de canaux disponibles dans la cellule.
L'état $X(t)$ représente le nombre de canaux occupés au temps $t$ (le nombre d'appels téléphoniques en cours) et $X(t) \in E$.

**Taux de transition**
L'état change à **l'arrivée** d'un nouvel appel et à **la fin** d'un appel, ce qui correspond à un processus **de naissance et de mort**.

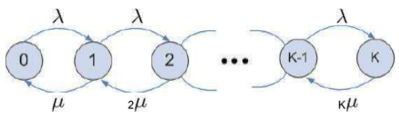


Figure 3.5: Modèle de Markov de réseau cellulaire

### 3.4.4 Solutions

Les équations de balance pour certains états sont comme suit :

$$\pi_0.\lambda = \pi_1.\mu$$

$$\pi_i(\lambda + i\mu) = \pi_{i-1}.\lambda + \pi_{i+1}(i+1)\mu \qquad i = 1, 2, ..., K-1$$

$$\pi_K.K\mu = \pi_{K-1}.\lambda$$

Tel que: $\pi_i = P_i$ est la probabilité d'être sur l'état $i$

Ces équations donnent:

$$\pi_i\lambda = \pi_{i+1}.(i+1)\mu \qquad i = 1, 2, ..., K-1$$

A partir des équations de balance et de théorème: $\sum_i \pi_i = 1$, nous aurons la solution suivante :

$$\pi_0 = \left[\sum_{i=0}^{K} \frac{\rho^i}{i!}\right]^{-1}$$

$$\pi_i = \pi_0 \frac{\rho^i}{i!}, \quad 1 \leq i \leq K$$

Tel que: $\rho = \frac{\lambda}{\mu}$ est le taux d'utilisation de la cellule (appelé aussi $u$)

La probabilité de blocage du système cellulaire est $\pi_k$ ou $P_k$

En effet, la probabilité qu'un appel arrive lorsque tous les cannaux sont occupés est la probabilité $\pi_K$.

$$P_k = P_0 \times \frac{u_k}{k!} \quad (Formule\ d'Erlang)$$

### 3.4.5 Graphe

- La figure 3.6 montre la probabilité de blocage $\pi_k$ du système cellulaire lorsque le nombre $K$ de canaux est de 20.

- Nous supposons que le taux d'arrivée par kilomètre carré est de deux appels par heure et donc le taux d'arrivée par cellule $\lambda = 3,14 \times r^2 \times 2$ appels.
- L'axe $X$ correspond au **rayon** de la cellule et l'axe $Y$ à **la probabilité de blocage**.
- La ligne continue avec une coche carrée provient de **la formule d'Erlang** (qu'on a extrait des équations de balances) et la ligne pointillée avec une coche circulaire provient de **la simulation**.

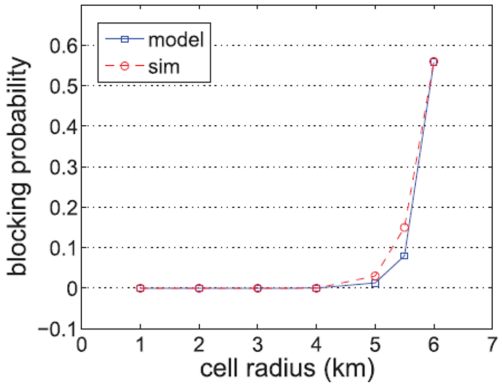


Figure 3.6: Probabilité de blocage avec différents rayons de cellule

### 3.4.6 Observations

- Nous pouvons observer que la probabilité de blocage est presque nulle lorsque le rayon de la cellule est inférieur à 4 $Km$ et qu'elle atteint 9% lorsque le rayon de la cellule devient $5,5$ $Km$.
- Nous pouvons conclure que la taille de la cellule est inférieure à 5 $Km$ pour satisfaire le seuil de probabilité de blocage du modèle (1% *ou* $0,01$).

# 4 Évaluation d'un simple système de transmission

## 4.1 Description du système

Considérant un réseau de campus qui a un groupe de réseaux locaux (LAN) et un routeur d'accès comme passerelle vers internet (Figure 4.1).

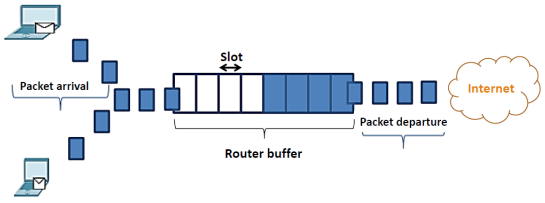


Figure 4.1: Un simple système de transmission

- Les étudiants commencent à remarquer une lenteur du réseau de connexion, alors, le gestionnaire de réseau décide **d'étudier le problème de performances** et constate que le routeur d'accès est un goulot d'étranglement potentiel.
- Pour résoudre le problème, le gestionnaire de réseau n'est pas sûr de remplacer ou de mettre à niveau le routeur.
- Dans les deux cas, le gestionnaire doit **déterminer les caractéristiques de routeur**.
- Pour étudier ces questions, le gestionnaire décide de **développer un modèle de performance** pour évaluer les différentes options.

## 4.2 Modélisation du système

On décide de simplifier le modèle en se concentrant sur les aspects clés :

- Étant donné que les réseaux locaux ne sont pas des goulots d'étranglement, on se concentre sur le routeur comme indiqué dans la figure précédente.
- On ignore la complexité des réseaux locaux et on modélise le trafic total qu'ils génèrent pour le routeur.

On étudie les données historiques et identifie les modèles de trafic d'entrée tels que:

- le nombre de paquets.
- également le CPU et la mémoire du routeur.

On sait déjà qu'on peut étudier ce système soit par **simulation informatique**, soit par **un modèle mathématique**. On opte pour un modèle mathématique car le système n'est pas excessivement complexe :

- Un modèle de chaîne **de Markov** peut fournir une compréhension détaillée **des performances** du routeur.
- La question maintenant est de savoir s'il faut utiliser un DTMC (Markov à temps discret) ou CTMC (Markov à temps continu): avec des slots d'intervalles de temps généralement fixes on décide d'utiliser un modèle **à temps discret**.

### 4.2.1 États

Dans le DTMC, on doit définir l'état $X_n$ à l'instant $n$.
$\rightarrow$ Que devrait être l'état **?** que devrait signifier le temps n **?**

Comme on s'intéresse par les performances du routeur d'accès, pour observer le comportement de **l'occupation du buffer** et celui du CPU, on définit l'état $X_n$ par **le nombre de paquets dans le système au temps** $n$

$$X_n \in E = \{0, 1, 2, 3, ..., N1, N\}$$

Ici, $n$ est l'intervalle de temps suffisamment long pour transmettre un paquet (dit **slot**) et $N$ est **la taille du buffer** (en nombre de paquets).

### 4.2.2 Données historiques

Après avoir étudié les données historiques, on découvre que :

- Les paquets arrivent au routeur avec **une probabilité** $\alpha$ dans l'intervalle de temps $n$, indépendamment de toute autre chose.
- Le paquet qui arrive à l'intervalle de temps $n$ est disponible pour être transféré dans l'intervalle de temps suivant $n + 1$.

- Le CPU du routeur est impliqué dans de nombreuses tâches différentes et alloue seulement une fraction $\beta$ de ses cycles à la tâche de transmission de paquets. En conséquence, le routeur est capable de transmettre un paquet avec **une probabilité** $\beta$ dans un emplacement donné.
- Avec une probabilité de $(1-\beta)$, le CPU effectue une tâche différente.
- On simplifie ainsi le système en supposant que les arrivées et les départs sont indépendants.

### 4.2.3 Probabilités de transition

- À partir de cette compréhension des arrivées et du comportement du routeur, on peut spécifier les probabilités de transition.
- Notant que les transitions d'état se produisent suite à l'arrivée/départ d'un paquet :
  - Lorsqu'un paquet arrive à l'instant $n$, l'occupation du buffer $X_n$ augmente ou reste la même selon si le départ a lieu ou non.
  - S'il y a un départ dans le même slot, alors $X_{n+1} = X_n$; sinon $X_{n+1} = X_n + 1$
- De même, s'il n'y a pas d'arrivée dans l'intervalle de temps $n$, $X_n$ reste le même ou diminue d'une unité, selon si le départ a lieu ou non.
- Ainsi, la matrice de probabilités de transition $P$ est la suivante:

$$P(i,j) = \begin{cases} C_1 = \alpha(1-\beta), & j = i+1, i = 1,2,...; \\ C_2 = (1-\alpha)\beta, & j = i-1, i = 1,2,...; \\ C_3 = \alpha\beta + (1-\alpha)(1-\beta), & j = i, i = 1,2,...; \\ \alpha, & i = 0, j = 1; \\ (1-\alpha), & i = 0, j = 0; \\ 0, & \text{sinon} \end{cases}$$

### 4.2.4 Solution du modèle

Les équations de balance de ce modèle de **Markov** sont comme suit:

$$p_0\alpha = p_1 C_2$$

$$p_1 = p_0\alpha + p_1 C_3 + p_2 C_2$$

$$p_n = p_{n-1}C_1 + p_nC_3 + p_{n+1}C_2 \quad \text{pour } n \geq 2$$

A partir de ces équations et de théorème $\sum p_n = 1$ , on constate que $p_n$ est donnée par:

$$p_n = \frac{\alpha C_1^{n-1}}{C_2^n} p_0$$

$$p_0 = \left(1 + \sum_{n=1}^{N} \frac{\alpha C_1^{n-1}}{C_2^n}\right)^{-1}$$

- À partir de la modélisation faite, le nombre moyen de paquets dans le système (backlog) $E[X]$ (ou $L$) peut être calculé comme suit:

$$E[X] = \sum_{n=1}^{N} n \times p_n$$

- Le backlog moyen est tracé sur la figure 4.2 pour différentes valeurs de $\alpha$ et $\beta$.
- L'axe des $x$ correspond à la valeur de $\alpha$ et l'axe des $y$ correspond au **backlog moyen**.
- Les trois lignes de gauche à droite correspondent aux différentes probabilités de transmission $\beta = 0.3, 0.6$ *et* $0.9$ respectivement.

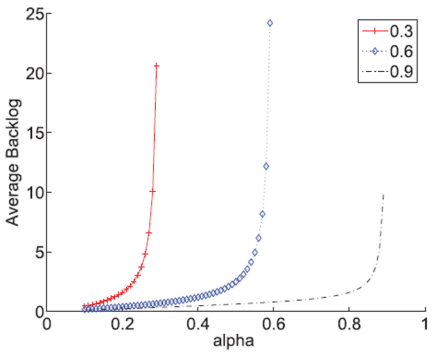


Figure 4.2: le backlog moyen en fonction de la probabilité avec différentes valeurs de $\beta = 0.3, 0.6, 0.9$.

### 4.2.5 Observations

- Selon le graphe, on note que lorsque la probabilité d'arrivée approche de $\beta$, l'occupation moyenne du buffer augmente rapidement.
- On peut calculer le retard moyen $W$ à partir du backlog moyen $L$ en utilisant la formule de Little : $L = \alpha W$.

# 5 Évaluation d'un protocole d'accès aléatoire: Aloha

## 5.1 Présentation de protocole Aloha

ALOHA était un système de **réseau sans fil** novateur développé à l'Université de Hawaï au début des années 1970. Afin de permettre la communication entre les campus séparés de l'Université, le professeur Abramson a développé le réseau à commutation de paquets avec **le premier schéma à accès multiple**.

Comme plusieurs émetteurs partagent **un seul canal de communication**:

- Si deux nœuds ou plus transmettent simultanément, les transmissions échouent.
- Si aucun nœud ne transmet, le canal n'est pas utilisé. (la différence avec le cours précédent c'est qu'ici la communication est sans fil)

Le défi du protocole **d'accès multiple** est de coordonner plusieurs émetteurs pour atteindre l'efficacité et l'équité d'accès.

## 5.2 Description du système

Il existe deux versions du protocole ALOHA: **pur** et slotté (de l'anglais **slotted**).

- Dans le protocole **ALOHA pur**: chaque émetteur peut transmettre des paquets à tout moment
- Dans **slotted Aloha** : la transmission des paquets est limitée au début des slots de temps (intervalles de temps)

Le **slotted Aloha** est plus synchronisé que le protocole pur et il est connu que cette restriction améliore **le débit** (d'un facteur de deux) par rapport au protocole ALOHA pur en réduisant le nombre **de collisions**.

- Le débit c'est la vitesse avec laquelle les paquets sont transmis, ex: nombre de bits par seconde.
- On parle de collision, quand deux émetteurs (ou plus) essaient de transmettre des paquets au même temps (plus précisément : dans le même slot de temps) et puisque dans ce cas il y a un seul canal : ces deux transmissions vont échouer.

## 5.3 Fonctionnement

Le protocole d'accès multiple slotted ALOHA fonctionne comme suit:

- Lorsqu'un paquet arrive sur un nœud **unbacklogged** [1], le nœud transmet simplement le paquet dans le début du slot après l'arrivée
  - risquant ainsi des collisions mais réalisant de très petits retards si les collisions sont rares.
- Lorsqu'une collision se produit, chaque nœud envoyant l'un des paquets entrant en collision découvre la collision quand 'il ne reçoit pas d'accusé de réception
  - Donc ce paquet devient **backlogged** [2]

**REtransmission des paquets backlogged** :

- Le nœud backlogged attend un nombre aléatoire de slots de temps (un délai aléatoire ) pour retransmettre le paquet.
- Si les autres nœuds backlogged devaient réessayer la transmission dans le slot suivant (le même slot que le premier nœud) → une autre collision serait inévitable.
- Un délai aléatoire est inclus dans le protocole pour éviter les collisions répétitives.

## 5.4 Définitions

### 5.4.1 Pure Aloha

- Le client envoie son message à n'importe quel moment sur le canal sans se soucier de savoir si celui-ci est déjà occupé à transmettre ou non (Figure 5.1).
- Si l'émetteur ne reçoit pas d'acquittement pour sa transmission durant un intervalle de temps donné, il réémet celui ci de la même manière.

[1] Un noeud qui ne contient pas un paquet backlogged
[2] Un paquet dont sa transmission a échoué et il est en attente d'être retransmis

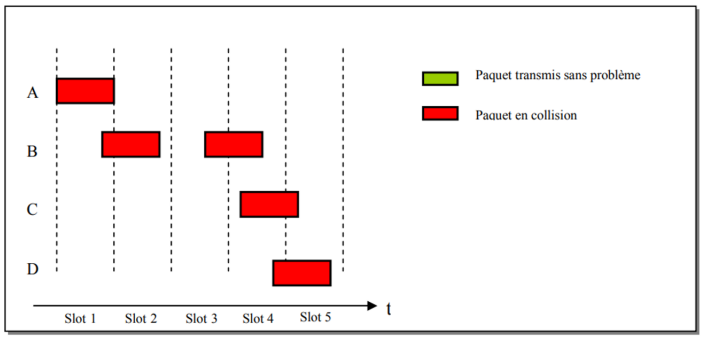


Figure 5.1: Pure Aloha

### 5.4.2 Slotted Aloha

- Ce protocole améliore le pur Aloha, en divisant le temps sur des « slots » de durée 577µs. Tous les émetteurs sont synchronisés sur cette division de temps (même horloge) : chaque nœud sait exactement lorsqu'un slot commence.
- Deux paquets entrent en collision s'ils sont prêts à être émis dans le même slot (Figure 5.2).

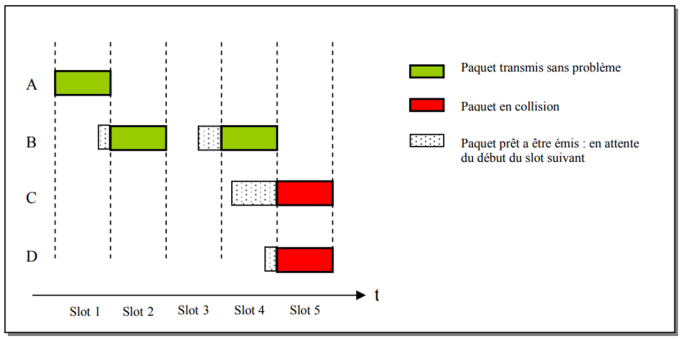


Figure 5.2: Slotted Aloha

## 5.5 Modélisation du Slotted Aloha par Chaine de Markov

### 5.5.1 Modélisation du système

- Le protocole Aloha est une formalisation de l'accès aléatoire (Random access) que l'on rencontre dans de nombreux **systèmes de communication**.
- Pour mieux comprendre **les performances** de protocole **slotted Aloha**, on développe un modèle de chaine de Markov.

- Comme le temps est une séquence de **slots** dans slotted ALOHA, c'est logique de considérer un modèle de chaine de **Markov à temps discret**.

### 5.5.2 Évènements

Les événements possibles qui modifient le nombre de nœuds backlogged sont:

(1) les arrivées de paquets vers des nœuds unbacklogged

(si un paquet arrive à un nœud unbacklogged celui-ci devient backlogged si sa première transmission est ratée)

(2) Une transmission réussie et

(3) Une transmission ratée.

### 5.5.3 Hypothèses

- Il y a $N$ nœuds dans le système; soit $n$ le nombre de nœuds backlogged au début d'un slot donné.
- Chaque nœud backlogged transmet un paquet avec une probabilité $p$, indépendamment des autres nœuds.
- Chacun des nœuds $(N - n)$ unbacklogged transmet un paquet qui est arrivé dans le slot précédent.
- Les paquets arrivent au système selon un processus de poisson avec un taux $\lambda$ dans un slot.
- Étant donné que le nombre de ces arrivées dans une unité de temps est distribué par un processus de Poisson :
  - La probabilité qu'aucun paquet n'arrive est : $e^{-\lambda}$
  - Ainsi, la probabilité qu'un nœud unbacklogged transmet son paquet dans un slot donnée est : $q = 1 - e^{-\lambda}$ (s'il y a un paquet arrivant à un nœud unbacklogged, il sera transmis certainement)

### 5.5.4 États

Soit $X_t$ le nombre de nœuds backlogged dans le système au début de slot de temps $t$. Alors, **l'espace d'états** $E$ du système est :

$$E = 0, 1, 2, ..., N$$

Où $N$ est le nombre de nœuds dans le système.

Une transmission réussie peut se produire si il y a :

- **Un paquet arrivant** aux nœuds **unbacklogged** (une seule transmission) et **aucune tentative** de transmission des nœuds **backlogged** ou
- **Aucun paquet arrivant** aux nœuds **unbacklogged** et **une seule tentative** de transmission des nœuds **backlogged**.

### 5.5.5 Probabilités de transition

- Soit $r_u(i, n)$ la probabilité que $i$ nœuds unbacklogged transmettent des paquets dans un **slot** donné.
- De même, soit $r_b(i, n)$ la probabilité que $i$ nœuds backlogged transmettent des paquets dans un **slot** donné.
- Avec ces notations, et pour tout état $n$ $(1 < n < N - 1)$ la probabilité de transition peut être écrite comme suit:

$$P_{n,n+1} = \begin{cases} r_u(0,n)r_b(1,n), & i = -1; \\ r_u(1,n)r_b(0,n) + r_u(0,n)[1 - r_b(1,n)], & i = 0; \\ r_u(1,n)[1 - r_b(0,n)], & i = 1; \\ r_u(i,n), & 2 \leq i \leq N - n \end{cases}$$

**De l'état $n$ à $n-1$:**

L'état $n$ passe à $n - 1$ si un seul nœud backlogged transmet un paquet et qu'aucun nœud unbacklogged fait une transmission.

**De l'état $n$ à $n$:**

L'état $n$ reste le même:

(1) lorsqu'il y a une tentative de transmission des nœuds unbacklogged et zéro tentative des nœuds backlogged ou

(2) lorsqu'il n'y a aucune tentative de transmission des nœuds unbacklogged et (zéro tentative des nœuds backlogged ou une collision des nœuds backlogged).

**De l'état $n$ à $n+1$:**

$n$ passe à $n+1$ lorsqu'une tentative des nœuds unbacklogged entre en collision avec des tentatives des nœuds backlogged.

**De l'état $n$ à $(n+2, n+3, ...n+N-n)$:**

L'état augmente de $i$ ($\geq 2$) lorsqu'il y a $i$ tentatives à partir des nœuds unbacklogged, indépendamment des nœuds backlogged.

## 5.6 Les Performances d'Aloha par Analyse Heuristique

### 5.6.1 Analyse heuristique

Il est connu que **l'efficacité** du protocole slotted ALOHA est de 36% sous certaines hypothèses idéalisées. Cela signifie que sur 100 slots de temps, seuls 36 emplacements peuvent être utilisés pour la transmission de données et que les 64 emplacements restants sont gaspillés. Le gaspillage provient des collisions.
Pour comprendre l'efficacité de 36%, dans cette deuxième partie de cours, nous évaluons les performances de protocole Aloha de ses deux types : Pure et Slotted, en se basant sur une analyse heuristique.

### 5.6.2 Offered channel traffic

On considère un ensemble de sources de trafic générant $\lambda$ paquets par secondes et $Tp$ la durée moyenne d'un paquet en secondes.
On introduit alors le trafic normalisé (ou **offered channel traffic**) :

$$G = \lambda \times Tp$$

$G$ représente ainsi **le nombre moyen de tentatives de transmission** de paquets pendant la durée d'un paquet (dans un slot de temps).

### 5.6.3 Nombre de paquets générés

On considère alors que la probabilité que $k$ paquets soient **générés pendant la durée d'un paquet** suit une distribution de **Poisson** et s'écrit :

$$Pr(k) = \frac{G^k e^{-G}}{k!}$$

### 5.6.4 Probabilité de succès : Pure ALoha

Si on identifie alors un paquet particulier on peut dire que la transmission sera effectuée avec **succès** si :

- Aucun paquet n'a été émis dans une "fenêtre temporelle" (slot de temps) de $Tp$ **avant l'émission de ce paquet** identifié
- et que ce dernier a été le seul a être émis **durant son temps d'émission** $Tp$ (Figure 5.3).

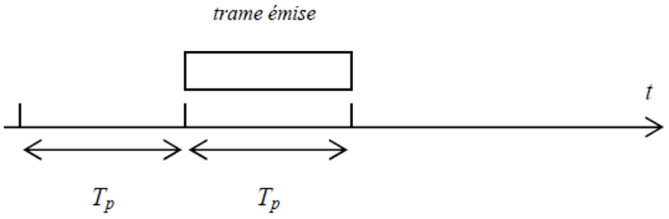


Figure 5.3: Occupation des slots

La probabilité de succès que l'on note $S$ s'écrit donc:

$$S = Pr[X = 0] \times Pr[X = 1] \implies S = G.e^{-2G}$$

Cette probabilité est la probabilité qu'un paquet soit émis et qu'il passe (soit transmis), c'est donc aussi le nombre moyen de paquets qui passent avec succès durant un slot et bien entendu ce nombre ne peut pas être plus grand que 1.

**- Débit effectif :**
Pour remonter au débit effectif en $bits/sec$, il faut regarder la taille du paquet **en nombre de bits** $Np$ et on obtient alors :

$$\text{Débit} = \left(\frac{Np}{Tp}\right) Ge^{-2G}$$

### 5.6.5 Probabilité de succès : Slotted ALoha

Si on se place dans le cadre d'un système de communications utilisant une structure de trame temporelle avec **des times slots définis**, on parle alors de Slotted ALOHA La probabilité de transmission avec succès est alors simplement égale à la probabilité d'émettre **un paquet** pendant **un slot**. On a alors :

$$S = Pr[k = 1] \implies S = Ge^{-G}$$

### 5.6.6 Points d'efficacité maximale

En fonction de $G$, on obtient les points d'efficacité maximale:

- Pour le pure ALOHA:

  - $G_{max} = 0.5$ paquet émis par temps paquet
  - $S_{max} = 0.18$ paquet émis et passé par temps paquet (en utilisant la formule de S)

- Pour le Slotted ALOHA:

  - $G_{max} = 1$ paquet émis par temps paquet
  - $S_{max} = 0.36$ paquet émis et passé par temps paquet

Les courbes de $S = f(G)$ pour l'ALOHA et le Slotted ALOHA sont représentées sur la Figure 5.4.

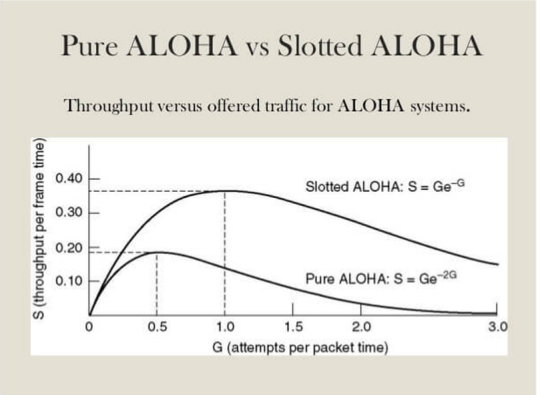


Figure 5.4: Pure Aloha vs Slotted Aloha

# 6 Évaluation d'un Réseau Wi-Fi – CSMA/CA

## 6.1 Description du système

- Un groupe d'appareils Wi-Fi utilisent le protocole CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance) pour partager un canal radio.
- C'est un protocole d'accès multiple avec écoute de canal et évitement de collision.
- Un appareil qui souhaite transmettre des données doit d'abord écouter le canal pour un temps prédéterminé pour vérifier si un autre nœud est en transmission sur le canal. Si le canal est détecté "libre", le nœud est autorisé à entamer le processus de transmission. Si le canal est ressenti «occupé», le nœud reporte sa transmission (back off) pour une période de temps aléatoire.
- L'appareil avec le temps d'attente aléatoire le plus court retransmet son paquet.
- Si deux émetteurs sont suffisamment proches (dans la même portée radio), ils ne peuvent pas émettre simultanément sous le protocole CSMA / CA car l'appareil ressent que le canal radio est occupé quand l'autre transmet.
- Nous nous intéressons au débit normalisé des émetteurs du réseau Wi-Fi et donc le débit normalisé des liens entre chaque source et son destinataire.
- Ce cours présente l'évaluation des performances de ce système dans l'objectif de pouvoir calculer à la fin le débit normalisé de chaque lien de réseau.

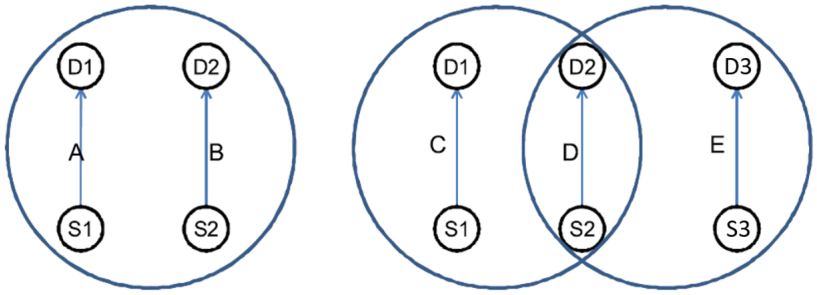


Figure 6.1: Exemple de deux domaines de collision

- Dans le modèle de domaine de collision multiple, un lien[1] n'interfère pas avec

[1]Un lien sans-fil (a wireless link): présente une transmission sans fil entre un nœud source (émetteur) et un nœud de destination (récepteur).

tous les liens mais avec certains d'eux. Dans la Figure 6.1(b), le lien $C$ interfère avec le lien $D$, mais pas avec le lien $E$.

- Dans Figure 6.1, les liens $A$ et $B$ ne peuvent pas être utilisés simultanément car ils sont voisins les uns des autres. Cependant, les liens $C$ et $E$ peuvent être utilisés simultanément.

## 6.2 Graphe de conflit

- Le modèle de Markov de CSMA utilise un modèle d'interférence appelé graphe de conflit.
- **Les sommets** de ce graphe sont les liens du réseau et **ses arêtes** connectent les liens interférents.
- Les liens **interférents** sont les liens qui ne peuvent pas transmettre **simultanément**.
- La Figure 6.2 montre les graphes de conflit qui correspondent aux deux réseaux de la Figure 6.1.
- Par exemple, le graphe de la Figure 6.2 (b) montre que les liens $C$ et $D$ sont en conflit, tout comme les liens $D$ et $E$. Ce graphe indique également que les liens $C$ et $E$ ne sont pas en conflit puisqu'ils ne sont pas connectés par une arête dans ce graphe.


a. Domaine de collision unique

b. Domaine de collision multiple

Figure 6.2: Graphe de conflit: le sommet dans ce graphe correspond à un lien dans le réseau; une arête dans ce graphe représente une relation d'interférence entre deux liens du réseau.

## 6.3 Les Performances du système par Modélisation de Markov

### 6.3.1 Hypothèses

- Chaque lien a des temps d'attente distribués selon une loi exponentielle;
  PS: quand on parle de temps d'attente d'un lien on signifie ainsi le temps d'inter-arrivée des données à transmettre sur ce lien.

- Les temps de transmission de donnée sont distribués selon une loi exponentielle;
- Les conflits entre les liens sont capturés par un graphe de conflit.
- La distribution exponentielle dans les deux premières hypothèses est nécessaire pour modéliser les activités des liens par un CTMC.
- Avec ces hypothèses, nous définissons un CTMC dans la suite de ce cours.

### 6.3.2 Espace d'états

- Le modèle de Markov de ce système présente l'état des liens $L$ dans le réseau.
- Soit $x_l(t)$ état de lien $l$ au temps, qui peut être soit actif (occupé, transmission en cours) soit inactif (libre).
- Quand le lien est libre $x_l(t) = 0$; sinon, $x_l(t) = 1$
- Soit $X(t) = [x_l(t), l \in L]$ le vecteur des états de tous les liens du réseaux et $E$ l'ensemble des états possibles de $X(t)$.
- En raison de conflits entre les liens, $E$ a un nombre d'éléments inférieur à $2^k$ où $k$ est le nombre de liens du réseau.
- Considérant le réseau de la Figure 6.1 (a): Puisque les liens $A$ et $B$ interfèrent l'un avec l'autre, l'état $(1, 1)$ n'est pas possible.
- Pour cet exemple de réseau, L'espace d'états est : $E = \{(0, 0), (1, 0), (0, 1)\}$
- De même, pour le réseau de la Figure 6.1 (b), l'espace d'états est: $E = \{(0, 0, 0), (1, 0, 0), (0, 1, 0), (0, 0, 1), (1, 0, 1)\}$

### 6.3.3 Transitions

Dans le modèle de Markov de ce système, il existe deux types de transitions:

- le début de transmission
- et la fin d'une transmission.

$T_{tr}$ est le temps de transmission de données et $T_{cd}$ est le temps de countdown, d'inter-arrivée des données à transmettre et le temps d'attente avant de transmettre.

- $\lambda = \frac{1}{E[T_{cd}]}$ tel que: $E[T_{cd}]$ la durée moyenne d'inter-arrivée.
- $\mu = \frac{1}{E[T_{tr}]}$ tel que $E[T_{tr}]$ est la durée moyenne de transmission.

Alors, pour tous deux états connectés, la transition de l'état gauche à l'état droit se produit avec le taux $\lambda$ la transition de l'état **droit** à l'état **gauche** avec le taux $\mu$.

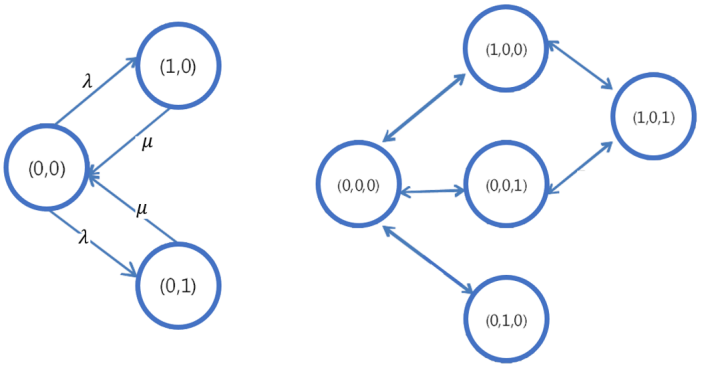


Figure 6.3: Modèle de Markov de transition d'états pour les réseaux de Figure 6.1

La Figure 6.3 montre le modèle de transition d'états du modèle de Markov CSMA pour les deux réseaux illustrés dans la Figure 6.1. Dans le modèle de gauche, il n'y a que trois états $(0, 0), (1, 0), (0, 1)$, car les deux liens sont en conflit.

- L'état (0, 0) passe à l'état (1, 0) avec le taux $\lambda$ et de (1, 0) à (0, 0) avec le taux $\mu$.

Le modèle de droite correspond au réseau à trois liens de la Figure 6.1.

- Dans ce modèle, nous ne montrons pas les taux de transition, par souci de simplicité.

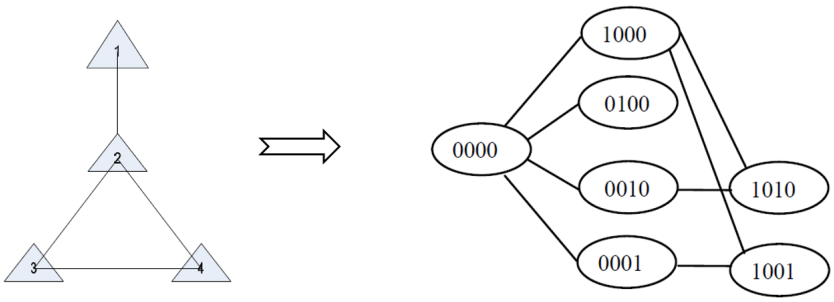


Figure 6.4: Le graphe de conflit d'un réseau et son modèle de Markov correspondant

Figure 6.4 présente un graphe de conflit d'un réseau (à gauche) et son modèle de transition d'états (à droite). Par exemple, le passage de 1000 à 1010 est dû au début d'une transmission sur le lien 3 pendant que le lien 1 est en transmission; tandis que la transition inverse de 1010 à 1000 est due à la fin d'une transmission sur le lien 3 tandis que la transmission de lien 1 se poursuit.

### 6.3.4 Calcul du débit normalisé (Normalized throuhput)

- $P_s$ est la probabilité que le réseau soit dans l'état $s$ (appelée aussi la fraction de temps pendant laquelle le réseau est dans l'état $s$).
- $s$ est un vecteur qui présente l'état de tout lien dans le réseau (1 si occupé, 0 si libre).
  - $s = s_1 s_2 s_3 ... s_L$
  - $L$ est le nombre total de liens dans le réseau
  - $s_3$ vaut 1 si le lien 3 est occupé ou 0 si le lien 3 est libre.
- Par exemple: dans la figure 6.4, l'état $s = (0, 0, 0, 1)$ signifie que tous les liens 1, 2, 3 sont libres et le lien 4 est occupé (en transmission).
  - $P_s = P_{s_1 s_2 s_3 ... s_L}$
- **Le débit normalisé** d'un lien présente **l'utilisation de ce lien** (occupé, en transmission)
- Le débit normalisé de lien $i$ est:

$$x_i = \sum_{s:s_i=1} P_s \tag{1}$$

  - C'est la somme des probabilités de tous les états $s$ où $s_i = 1$, autrement dit, de tous les états $s$ où le lien $i$ est occupé.
  - Par exemple: dans le réseau de la Figure 6.4, le débit de lien 3 est :

$$x_3 = P_{0010} + P_{1010}$$

- Une observation dit que tous les états avec le même nombre de liens occupés (c'est-à-dire les états dans la même colonne dans le modèle de Markov) ont la même probabilité.
- Par exemple: dans la Figure 6.4, $P_{1010} = P_{1001}$ (les états de la troisième colonne de modèle)

- Pour pouvoir calculer $P_s$, il faut appliquer les équations de balance sur le modèle de Markov de réseau. La formule de $P_s$ à la fin, doit être présentée en fonction de $\lambda$ et $\mu$.

### 6.3.5 Solution: Exemple de Figure 6.4

- **Probabilités de transition**

  En appliquant les équations de balance sur le modèle de Markov de la Figure 6.4, on obtient:

$$P_{0000} = \left(1 + 4\frac{\lambda}{\mu} + 2\left(\frac{\lambda}{\mu}\right)^2\right)^{-1}$$

$$P_{1000} = P_{0100} = P_{0010} = P_{0001} = \left(\frac{\mu}{\lambda} + 4 + 2\frac{\lambda}{\mu}\right)^{-1}$$

$$P_{1010} = P_{1001} = \left(\left(\frac{\mu}{\lambda}\right)^2 + 4\frac{\mu}{\lambda} + 2\right)^{-1}$$

- **Le Débit Normalisé de chaque lien**

  En appliquant la formule 1 sur le modèle de Markov de la Figure 6.4, on obtient:

$$\begin{cases} x_1 = P_{1000} = P_{1010} = P_{1001} = \left(\frac{\mu}{\lambda} + 4 + 2\frac{\lambda}{\mu}\right)^{-1} + 2\left(\left(\frac{\mu}{\lambda}\right)^2 + 4\frac{\mu}{\lambda} + 2\right)^{-1} \\ x_2 = P_{0100} \\ x_3 = P_{0010} + P_{1010} \\ x_4 = P_{0001} + P_{1001} \end{cases}$$

## 6.4 Les Performances du système par Analyse heuristique

### 6.4.1 Méthode de BoE (Back of the Envelope)

Le concept de la méthode BoE est que pour de nombreux états de réseau, des résultats de débits peuvent être obtenus par un simple calcul manuel. Les débits normalisés (throughput) des liens sont présentés par un vecteur:

Ex. Le vecteur des débits normalisés des liens 1,2,3 et 4 est ($Th_1$ $Th_2$ $Th_3$ $Th_4$)

- Etape 1: À partir de graphe de conflit, trouver les états possibles de réseau pour lesquelles un nombre maximal de liens peuvent être actifs simultanément;
- Etape 2: Additionner les vecteurs et diviser la somme par le nombre d'états possibles trouvés.

Ex. Dans le réseau de Figure 6.4, il y a deux états possibles: (1 0 1 0) et (1 0 0 1) et donc le vecteur des débits est (1 0 0.5 0.5).

- Cette procédure est purement algorithmique et il n'y a pas d'analyse stochastique complexe.
- Cette méthode est d'origine proposée à partir de l'observation de la simulation et des résultats expérimentaux du réseau réel plutôt que de la construction théorique.
- Les résultats de débits de la méthode précédente (Markov) se rapprochent des résultats de BoE lorsque $\frac{\mu}{\lambda}$ approche de zéro.

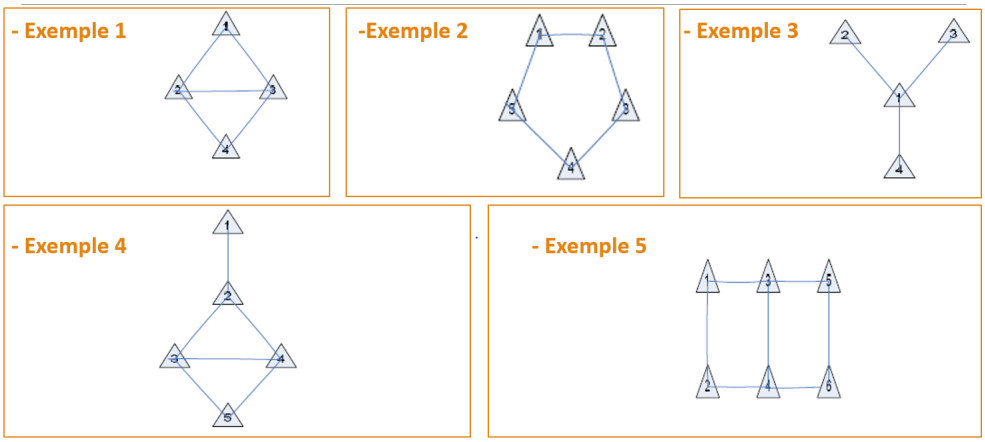


Figure 6.5: Exemples de graphes de conflits

**Exemple :** Calculer le débit des liens dans chaque graphe de conflit présenté dans la Figure 6.5.

# Travaux Pratiques (Simulation)

# 7 Évaluation d'un système cellulaire

## 7.1 Exercice TP n°01: (Modèle analytique)

### 7.1.1 Énoncé

(A) **Étape 1** (Figure 7.1)

- Écrire un programme Java qui permet de calculer la probabilité de blocage (formule d'Erlang de cours) en modifiant les valeurs de rayon de 1 à 7 avec un pas de 0,5.
- Les données sont:
  - $\mu = 20$
  - Le nombre de canaux disponibles pour une cellule $K = 20$
- À partir de ces calculs, tracer la courbe présentant la probabilité de blocage en fonction de rayon de la cellule en utilisant **JFreeChart**.

(B) **Étape 2** (Figure 7.2)

- Sur le même graphique, tracer une deuxième courbe pour $\mu = 15$

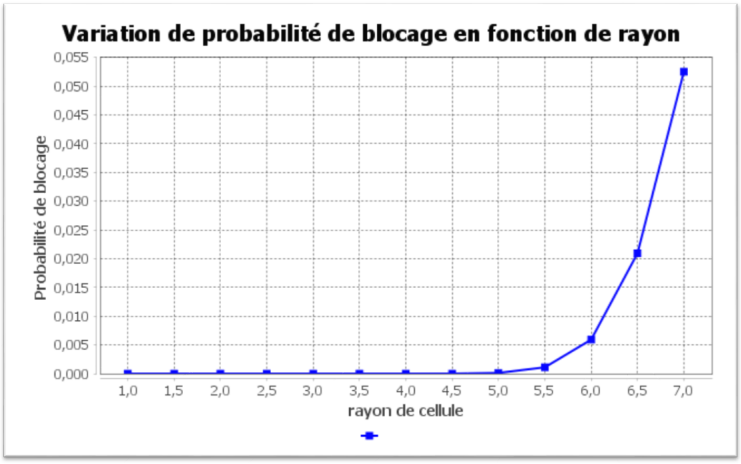


Figure 7.1: La probabilité de blocage avec différents rayons de cellule (mu=20)

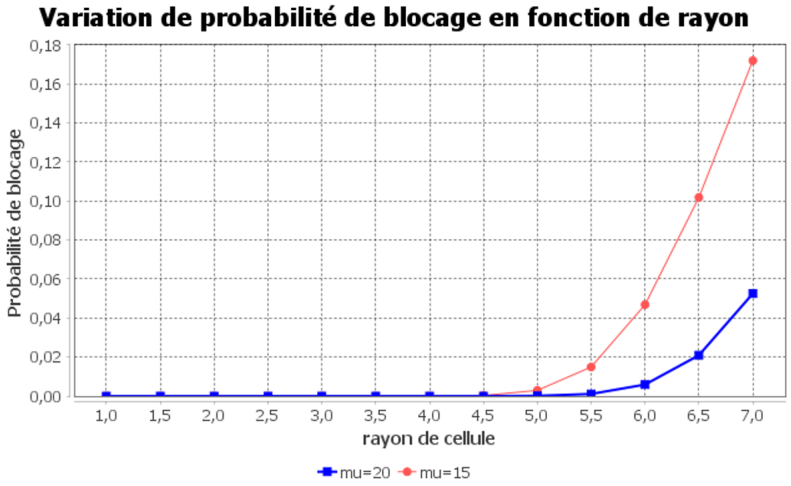


Figure 7.2: La probabilité de blocage avec différents rayons de cellule (mu=20 et mu=15)

### 7.1.2 Java Build Path - JFreeChart

- Le Java build path est utilisé lors de la compilation d'un projet Java pour découvrir les classes dépendantes
- Le Java Build Path peut être vu et modifié à l'aide de la page Java Build Path de la boîte de dialogue des propriétés du projet Java.
- Eclipse : Clic droit sur le projet → **Build Path** → **Configure Build Path** → **Libraries** → **Add External Jars** (car les Jars se trouvent dans un fichier séparé)
- Netbeans : Clic droit sur le projet → Properties (ou autres) → **Libraries** → **Add Jar/folder**
- Rajouter les paths suivants (selon l'emplacement de votre fichier JfreeChart):
  - C:jfreechart-1.0.19-lib-jfreechart-1.0.19;
  - C:jfreechart-1.0.19-lib-jcommon-1.0.23;

## 7.2 Exercice TP n°02: (Simulation)

L'objectif de cet exercice est de faire évaluer les performances d'un système cellulaire avec blocage en utilisant la SIMULATION.
Simuler ce système en Java, dont:

- Le rayon prend les valeurs de 1 à 7 avec un pas de 0,5
- Le nombre de canaux disponibles $k = 20$

### 7.2.1 Génération des variables aléatoires

Méthode de la fonction inverse :

- Temps d'inter-arrivée = $\frac{-ln(u)}{\lambda}$
- Durée de service = $\frac{-ln(u)}{\mu}$

Utiliser la méthode de génération des variables aléatoires pour définir :

- Le temps d'inter arrivée (pour trouver le temps d'arrivée de chaque appel) dont le taux $\lambda$ dépend de rayon $r$ ($\lambda = 3.14 \times r^2 \times 2$)
- La durée d'appel (durée de service) dont le taux est $\mu = 20$

### 7.2.2 Évènements

Les évènements qui peuvent changer l'état de ce système sont :

- L'arrivée d'un nouvel appel,
- La fin d'appel (sortie du système),

Dans le code Java, chaque évènement possède au moins :

- Le type : 0 pour arrivée, 1 pour fin.
- Le temps de l'événement:
  - Le temps d'arrivée si le type est 0

– Le temps de sortie si le type est 1

L'évènement de départ doit contenir un troisième paramètre qui présente le numéro de canal occupé par le client correspondant. La structure générale de la classe Event est présentée dans la Figure 7.3.

```
9  public class Event {
10     protected int type; // 0 = arrivee, 1 = depart
11     protected double instant;
12     int numC;
13
14     public Event(int type, double instant, int numC){
15         this.type = type;
16         this.instant = instant;
17         this.numC=numC;
18     }
19     public Event(int type, double instant){
20         this.type = type;
21         this.instant = instant;
22     }
23     public double getInstant() {
24         return instant;
25     }
26     public int getType() {
27         return type;
28     }
29     public int getNumC() {
30         return numC;
31     }
32
33 }
```

Figure 7.3: La classe Event

### 7.2.3 Probabilité de blocage & courbe

Dans le début de la simulation:

- Tous les canaux sont libres, et ils sont placés dans un tableau «Canaux» d'une taille $k$ et chaque case contient 0 si le canal est libre et 1 si le canal est occupé.
- La valeur de «blocage» est 0
  Ps: cette valeur est incrémentée de 1 quand un client ne trouve aucun canal libre dans le tableau «canaux»
- Le temps d'arrivée du premier appel est 0 et donc le temps correspondant au premier évènement est 0

A travers cette simulation, pour chaque appel (client) on peut déterminer :

- Le temps d'arrivée,
- Le numéro de canal utilisé (de 1 à 20), si il y a un canal libre sinon le client sera bloqué et donc la probabilité de blocage est incrémenté par 1.
- La durée de service

- La fin de service

Le client cherche un canal libre dans le tableau «canaux» depuis la première case jusqu'à la dernière. Si par exemple, le client n° 10 prend le canal n°5, il libère ce même canal juste après la fin de son service.

La Probabilité de blocage est la valeur finale de «blocage» (incrémentée de 1 à chaque blocage) divisée par le nombre total d'appels (de clients):

$$Pb = \frac{\sum_{i=1}^{N} p_i}{N}$$

Tels que: $p_i$ est la valeur de blocage de chaque client : 1 si il y a un blocage , 0 sinon. $N$ est le nombre de clients (appels) exécutés.

→ La simulation prend fin après un temps de simulation limité (ex. 30) ou un nombre d'appels limité (ex. 100).

À partir de cette simulation, tracer la courbe présentant la probabilité de blocage en fonction de rayon de la cellule en utilisant la classe JFreeChart.

→ La courbe trouvée ressemble à celle du modèle analytique du 1er TP. La Figure 7.4 présente les graphes résultant de plusieurs exécutions de cette simulation.

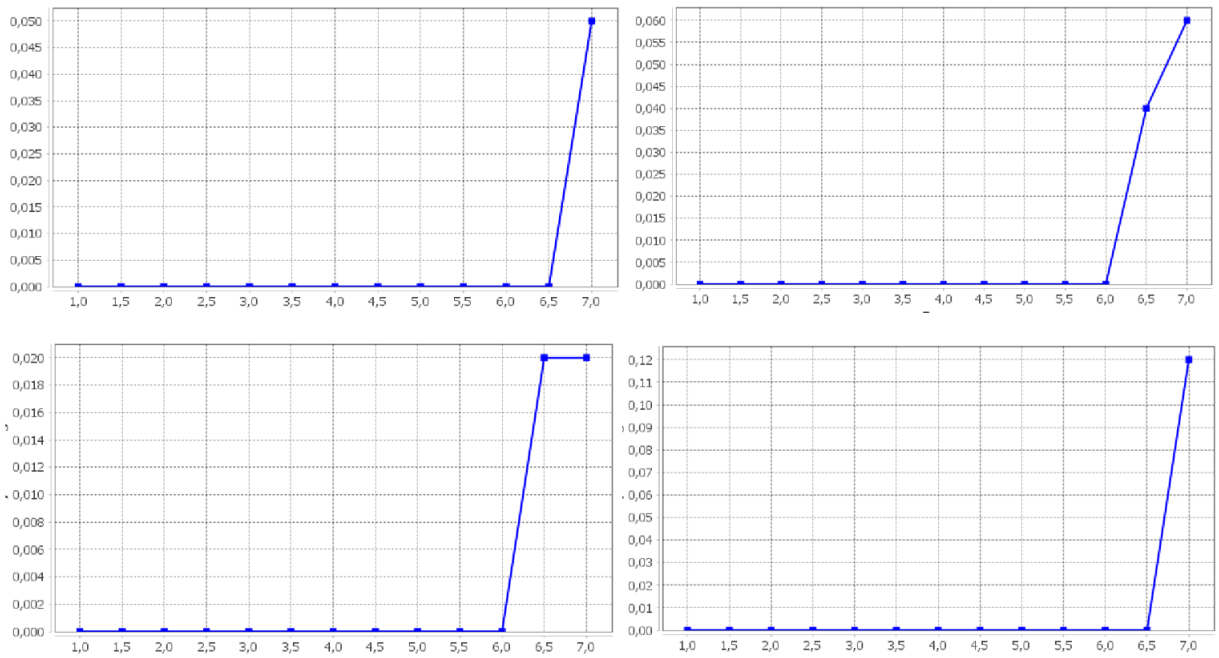

Figure 7.4: Probabilité de blocage en fonction de rayon à partir d'une simulation.

# 8 Simulateur Omnet++

## 8.1 Introduction

Ce document présente comment installer le simulateur réseau Omnet++ sur Windows et puis comment installer la plateforme INET, en plus des étapes de configuration et de compilation.

## 8.2 Installation Omnet++

Omnet++ est un environnement de simulation à évènements discrets basé sur le langage C++, une application open source et sous licence GNU. Il est programmable, paramétrable et modulaire ainsi grâce à son architecture flexible et générique, il a été utilisé avec succès dans divers domaines, notamment la simulation :

- des réseaux de file d'attente,
- des protocoles de communication,
- des réseaux filaires et sans-fil,
- et donc l'évaluation des performances pour des systèmes simples et complexes.

### 8.2.1 Téléchargement

(1) **Version Windows**

- OMNeT ++ (version 6.0) est prise en charge uniquement sur les versions 64 bits de Windows.
- Si vous avez besoin de versions Omnet++ pour 32 bits sous Windows, vous pouvez utiliser **OMNeT++ 5.0**.
- Téléchargez OMNeT++ depuis https://omnetpp.org/download/. Assurez-vous de sélectionner l'archive spécifique à Windows.
- Sur le site web : (Figure 8.1)

  → **Previews** : pour Omnet++ 6.0
  → **Older versions** : pour autres.

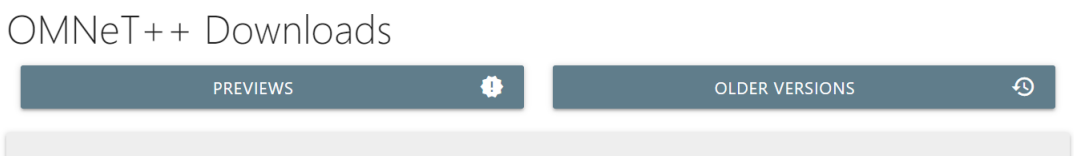


Figure 8.1: Interface de site de téléchargement

En plus des fichiers OMNeT ++, ce package téléchargé comprend :

- un compilateur C ++,
- un environnement de construction en ligne de commande,
- et toutes les bibliothèques et programmes requis par OMNeT ++.

(2) **Un autre système d'exploitation**
Pour l'utilisation d'Omnet++ sur un autre système d'exploitation, vous pouvez consulter les liens suivants :

- Téléchargement : https://omnetpp.org/download/
- Installation : https://doc.omnetpp.org/omnetpp/InstallGuide.pdf

### 8.2.2 Après téléchargement

- Extraire le fichier .zip
- Placer le dossier dans un répertoire: (ex. C:-Sim-omnetpp-6.0pre10-src-windows)

(1) **Configurer et compiler Omnet++**

- Dans le répertoire **omnetpp-6.0pre10**, démarrez **mingwenv.cmd** en double-cliquant dessus.
- Il affichera une console avec le shell bash MSYS (figure 8.2), où le chemin est déjà défini pour inclure le répertoire **omnetpp-6.0pre10/bin**.
- Sur cette console de shell bash, entrez la commande : **./configure** et attendez que la configuration d'Omnet++ se termine.
- Puis, entrer la commande : **make** et attendez que la compilation d'Omnet++ se termine (cette compilation va prendre plus d'une heure tout dépend les performances de PC)

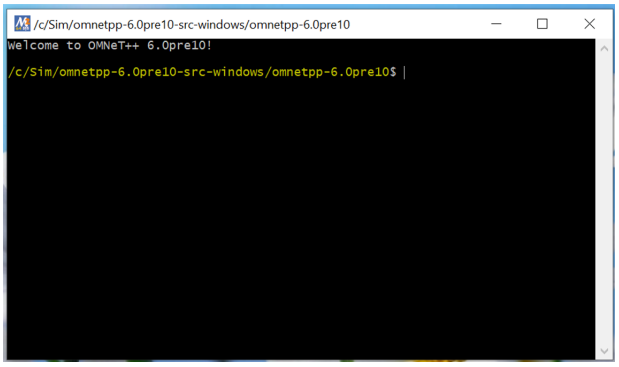


Figure 8.2: Console Shell bash Omnet++

(2) **Démarrage de l'IDE (Integrated Development Environment)**

OMNeT++ est livré avec un IDE de simulation basé sur Eclipse. Vous devriez pouvoir démarrer l'IDE à partir de la ligne de commande (figure 8.3) en tapant: **« omnetpp »**

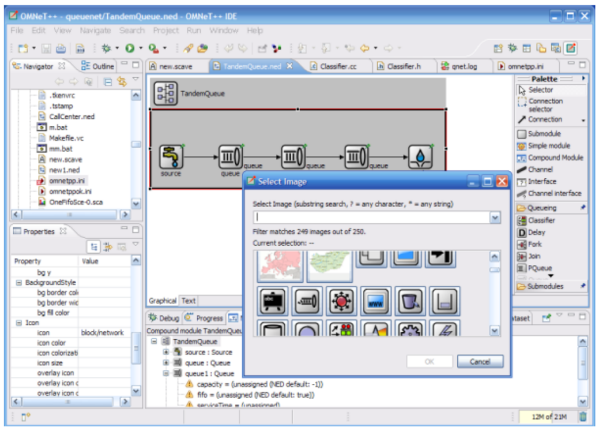

Figure 8.3: Omnetpp IDE

## 8.3 Installation INET

La plateforme INET (INET Framework) est une bibliothèque open source pour l'environnement de simulation OMNeT++. INET fournit des protocoles, des agents et d'autres modèles aux chercheurs et aux étudiants travaillant avec des réseaux de communication.
INET contient des modèles Internet (TCP, UDP, IPv4, IPv6, OSPF, BGP, etc.), les protocoles filaire et sans fil de la couche de liaison (Ethernet, PPP, IEEE 802.11, etc.), la prise en charge de la mobilité, des protocoles MANET, plusieurs modèles d'application et de nombreux autres protocoles.

Plusieurs plateformes de simulation prennent INET comme base et l'utilisent dans des directions spécifiques, telles que les réseaux véhiculaires, les réseaux de peer-to-peer ou LTE.

→ **Note** : Pour le manuel d'installation de INET (Installation automatique et installation manuelle) vous pouvez consulter le lien suivant : https://inet.omnetpp.org/Installation.html, ou simplement lisez la suite de ce document.

### 8.3.1 Téléchargement

- Téléchargez INET depuis le site : https://inet.omnetpp.org/Download.html
- Assurez-vous de sélectionner la version correspondante à votre version Omnet++ déjà installée.
- Pour la version **Omnet++6.0** la version compatible d'INET est celle de la Figure 8.4 ci-dessous.

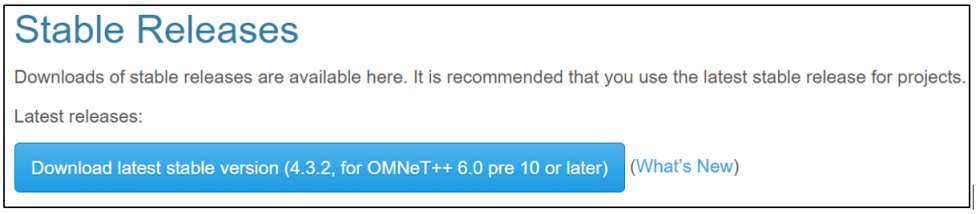


Figure 8.4: Version INET compatible avec Omnet++6.0.

### 8.3.2 Installation manuelle

- Après avoir téléchargé INET, Extraire le fichier **.Zip** dans le répertoire de votre choix, par exemple, le même répertoire où se trouve Omnet++.
- Lancer l'IDE Omnet++, allez sur : **File -> Import -> Projects** from Folder or Archive et puis dans **directory**, vous sélectionnez le projet **Inet**.
- Une fois qu'Inet est importé, il va apparaitre sur la partie gauche de votre **IDE Omnet++** (Figure 8.5).

- Pour compiler le projet Inet, clique droit dessus et **Build Project** (la compilation va prendre quelques minutes) Vous devriez maintenant pouvoir lancer des exemples de simulations existants dans le Projet Inet (Examples, Tutorials, ... etc).

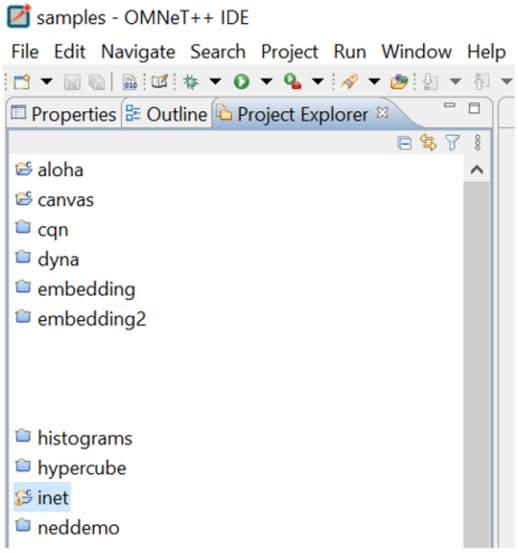


Figure 8.5: Project Explorer de l'IDE de Omnet++.

- Figure 8.6 montre la méthode la plus simple pour lancer la simulation d'un type de réseau, par exemple : **ad-hoc**.

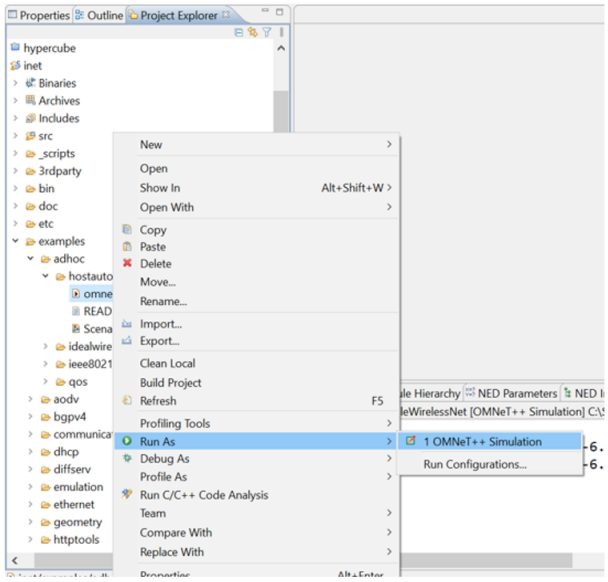


Figure 8.6: Lancer une simulation sur Omnet++.

## 8.4 Simulation

### 8.4.1 Modèle

Commençant par un "réseau" composé de deux nœuds. Les nœuds feront quelque chose de simple: l'un des nœuds créera un paquet, et les deux nœuds continueront

à transmettre le même paquet dans les deux sens. Nous appellerons les nœuds **tic** et **toc**.
Voici les étapes à suivre pour mettre en œuvre votre première simulation à partir de zéro.

### 8.4.2 Mise en place du projet

Démarrez l'IDE OMNeT++ en tapant **omnetpp** dans votre **terminal**. Une fois dans l'IDE, choisissez **new -> OMNeT++ Project** dans le menu.

Une boîte de dialogue d'assistant apparaîtra. Entrez **tictoc** comme nom de projet, choisissez *Projet vide* lorsque vous êtes interrogé sur le contenu initial du projet, puis cliquez sur **Terminer**. Un projet vide sera créé, comme vous pouvez le voir dans l'explorateur de projet. (Remarque: certaines versions d'OMNeT++ généreront un fichier **package.ned** dans le projet. Nous n'en avons pas besoin maintenant).

Le projet contiendra tous les fichiers appartenant à notre simulation. Dans notre exemple, le projet se compose d'un seul répertoire. Pour les simulations plus importantes, le contenu du projet est généralement trié dans les dossiers **src**/ et **simulations**/ , et éventuellement dans des sous-dossiers en dessous.

### 8.4.3 Ajout du fichier NED

OMNeT++ utilise des fichiers NED pour définir des composants et les assembler en unités plus grandes comme **des réseaux**. Nous commençons à implémenter notre modèle en ajoutant un fichier NED. Pour ajouter le fichier au projet, cliquez avec le bouton droit sur le répertoire du projet dans le panneau Explorateur de projets à gauche, puis choisissez **Nouveau -> Fichier** de description de réseau (NED) dans le menu. Entrez **tictoc1.ned** lorsque vous êtes invité à entrer le nom du fichier.

Une fois créé, le fichier peut être modifié dans la zone Éditeur de l'IDE OMNeT++. L'éditeur NED de l'OMNeT++ IDE a deux modes, **Design** et **Source** ; on peut basculer entre eux en utilisant les onglets en bas de l'éditeur.

- En mode **Design**, la topologie peut être éditée graphiquement, à l'aide de la souris et de la palette de droite de l'IDE.
- En mode **Source**, le code source NED peut être directement édité sous forme de texte.

→ Passez en mode **source** et entrez le code présenté dans la Figure 8.7.

```
13  simple Txc1
14  {
15      gates:
16          input in;
17          output out;
18  }
19
20  //
21  // Two instances (tic and toc) of Txc1 connected both ways.
22  // Tic and toc will pass messages to one another.
23  //
24  network Tictoc1
25  {
26      submodules:
27          tic: Txc1;
28          toc: Txc1;
29      connections:
30          tic.out --> {  delay = 100ms; } --> toc.in;
31          tic.in <-- {  delay = 100ms; } <-- toc.out;
32  }
```

Figure 8.7: Partie Source de fichier NED.

→ Lorsque vous avez terminé, revenez au mode **Design**. Vous devriez voir une conception comme celle de Figure 8.8.

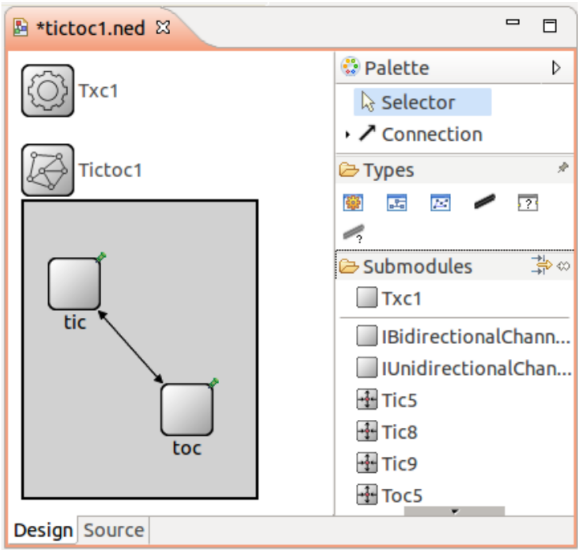


Figure 8.8: Partie Design de fichier NED.

Le premier bloc du fichier déclare $Txc1$ comme un type de **module simple**. Les modules simples sont atomiques au niveau NED. Ce sont également des composants actifs et leur comportement est implémenté en **C++**. La déclaration indique également que $Txc1$ a une **porte d'entrée** nommée **in** et une **porte de sortie** nommée **out**.

Le deuxième bloc déclare *Tictoc*1 en tant que **réseau**. *Tictoc*1 est assemblé à partir de **deux sous-modules**, tic et toc, les deux instances du type de module *Txc*1. La porte de sortie de tic est connectée à la porte d'entrée de toc, et vice versa. Il y aura **un délai** de propagation de **100 ms** dans les deux sens.

### 8.4.4 Ajout des fichiers C++

Nous devons maintenant implémenter la fonctionnalité du module simple *Txc*1 en C++. Créez un fichier nommé **txc1.cc** en choisissant **Nouveau -> Fichier source** dans le menu contextuel du projet (ou **Fichier -> Nouveau -> Fichier** dans le menu principal de l'EDI), et entrez le contenu de Code 1.

### 8.4.5 Ajout de fichier omnetpp.ini

Pour pouvoir exécuter la simulation, nous devons créer un fichier **omnetpp.ini** indiquant au programme de simulation le réseau que vous souhaitez simuler (car les fichiers NED peuvent contenir plusieurs réseaux).

Créez un fichier omnetpp.ini à l'aide de l'élément de menu **Fichier -> Nouveau -> Fichier d'initialisation (INI)**. Le nouveau fichier s'ouvrira dans un éditeur Inifile. Comme dans l'éditeur NED, l'éditeur Inifile a également deux modes, **Form** et **Source**, qui éditent le même contenu.

Pour l'instant, passez simplement en mode **Source** et entrez ce qui suit:

**[General]**
**network = Tictoc1**

**Algorithm 1** (txc1.cc)

```
# include <string.h>
# include <omnetpp.h>
using namespace omnetpp;
class Txc1 : public cSimpleModule
{
protected:
virtual void initialize() override;
virtual void handleMessage(cMessage *msg)
override;
};
// The module class needs to be registered with OMNeT++
Define_Module(Txc1);
void Txc1::initialize()
{
// Initialize is called at the beginning of the simulation.
// To bootstrap the tic-toc-tic-toc process, one of the modules needs
// to send the first message. Let this be ‘tic’.
// Am I Tic or Toc?
(strcmp("tic", getName()) == 0)
// create and send first message on gate "out". "tictocMsg" is an
// arbitrary string which will be the name of the message object.
cMessage *msg = new
cMessage("tictocMsg");
send(msg, "out");
}
}
void Txc1::handleMessage(cMessage *msg)
{
// The handleMessage() method is called whenever a message arrives
// at the module. Here, we just send it to the other module, through
// gate ‘out’. Because both ‘tic’ and ‘toc’ does the same, the message
// will bounce between the two.
send(msg, "out"); // send out the message
}
```

# Références


- MO, Jeonghoon. "Performance modeling of communication networks with Markov chains". Synthesis Lectures on Data Management, 2010, vol. 3, no 1, p. 1-90.
- Gunter Bolch, et al. "Queueing Networks and Markov Chains: Modeling and Performance Evaluation with Computer Science Applications". Wiley-Interscience (2006).
- YeQiong SONG, "Evaluation de performances", LORIA – Université de Lorraine, 2013.
- S. C. Liew, C. Kai, J. Leung and B. Wong, "Back-of-the-Envelope Computation of Throughput Distributions in CSMA Wireless Networks," 2009 IEEE International Conference on Communications, 2009, pp. 1-6.
- A. Varga and R. Hornig. "An overview of the OMNeT++ simulation environment". In Proceedings of the First International Conference on Simulation Tools and Techniques for Communications, Networks and Systems (SIMUTools 2008'), March 2008.