Rapport de stage DEA INFORMATIQUE 2004

La simulation à événements discrets

Validité d'un modèle et analyse statistique des données en sortie

Pascal Merindol

sous la direction de

S.Cateloin
Maître de Conférence en informatique
RP/LSIIT ULP Strasbourg

C.Michel
Professeur en informatique
Bioinformatique/LSIIT ULP Strasbourg

TABLE DES MATIERES :

TABLE DES FIGURES:

TABLE DES TABLEAUX :

Introduction

Lorsque l'on souhaite analyser le comportement d'un système dont l'expérimentation « grandeur nature » est coûteuse voire impossible, les méthodes de simulation sur ordinateur constituent une démarche de plus en plus courante, notamment grâce à la puissance sans cesse croissante des processeurs actuels. Cet outil d'expérimentation, que constitue la simulation et qui se doit d'être souple et extensible, se révèle donc être une opportunité très appréciable dans le cadre de réseaux de communication à grande échelle. En effet, un simulateur de réseaux de type IP permet le développement et la comparaison de différents protocoles dont le déploiement, pour sa phase de test et de validation, est impossible à l'échelle d'Internet.
Pourtant, la complexité de tels systèmes peut, dans l'étude de son comportement, conduire à la formation d'hypothèses éloignées de la réalité du système dans la mesure où la modélisation des nombreux paramètres entrant en jeu (et de leurs relations) est simplifiée.
Afin de produire des hypothèses comportementales valables et vérifiables, il est donc primordial de définir une méthodologie de simulation se basant sur un cadre précis, non ambigu et extensible, pour éviter d'éventuels effets de bord qui fausseraient les observations récoltées en sortie.
Par ailleurs, les mesures effectuées en sortie doivent être sujettes à un certain nombre d'indicateurs statistiques de confiance pour en valider la pertinence.
Nous allons donc, après avoir défini dans une première partie les caractéristiques générales d'une simulation, étudier dans une seconde partie les principaux facteurs de crédibilité (ou plutôt d'incertitude) dont doit faire preuve un simulateur.
La troisième partie dressera un état de l'art de la méthodologie de simulation, ainsi qu'un certain nombre d'hypothèses comportementales spécifiques au domaine des communications IP.
Le quatrième chapitre, quant à lui, définira les principaux outils de confiance qu'on peut associer aux mesures en sortie, ainsi que le contexte dans lequel ces outils sont valables.
Pour finir, le dernier chapitre développera sur un exemple, QoSim [35], une méthodologie de simulation pour des réseaux grande échelle, et une proposition d'analyse de données en sortie appropriée pour des réseaux à commutation de paquets.

1. LA SIMULATION : PRESENTATION GENERALE

1.1. LE SYSTEME ET SES OBJECTIFS D'ETUDE

La simulation est l'imitation du comportement et des opérations d'un processus ou d'un système réel dans le temps. Elle constitue une alternative à la reproduction physique du système. Pour étudier un système, il est nécessaire d'en dégager un certain nombre d'observations et d'analyser les caractéristiques comportementales de celui-ci afin de produire des hypothèses. Ces hypothèses sont réunies afin de construire un modèle ayant pour but de reproduire de manière simplifiée le système imité.
Si les relations introduites dans le modèle sont suffisamment simples on peut envisager une solution analytique pour obtenir des informations exactes sur le comportement du système, sinon la voie numérique s'impose. Elle passe par le développement d'un modèle qui va être utilisé le plus souvent au moyen d'un ordinateur (mais elle peut être manuelle) dans le but d'estimer les caractéristiques correspondant aux objectifs de l'étude.
Par conséquent, il est primordial de bien définir le système, son environnement et les lois qui régissent son fonctionnement pour construire un modèle valide.
On peut décrire un système en procédant de la manière suivante :

- identifier les éléments qui le composent (composant d'intérêt ou entité).
- déterminer les attributs (propriété d'intérêt) de ces éléments.
- spécifier les activités qui constituent le comportement du système (il s'agit des changements provoqués par les éléments et leurs attributs).

Notons que les activités peuvent être endogènes ou exogènes selon leurs interactions avec l'environnement, déterministes ou stochastiques selon la nature (aléatoire ou non) des changements d'états, nous en étudierons ces caractéristiques dans la suite de ce document (Voir chapitre 3).
Un système se définit par son état au cours du temps, celui ci étant la somme des états de tout ses attributs à un instant donné.Le schéma ci-dessous illustre le fonctionnement d'un système qu'il soit simple ou complexe, il est dans ce cas divisible en sous systèmes simples interconnectables.Il s'agit d'atteindre un but à partir de données réelles ou estimées, qui vont être converties selon des opérations se déroulant suivant un plan.

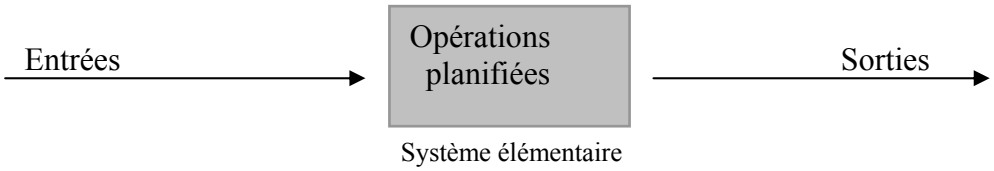


Figure 1: ***Schéma du fonctionnement d'un système simple.***

Dans l'analyse d'un système, d'une part il sera impératif de définir clairement les sorties qui correspondent aux objectifs de l'étude, les limites c'est-à-dire l'ampleur du système selon sa complexité, ses contraintes et son milieu, puis les entrées et les opérations qui s'y appliquent pour obtenir les sorties recherchés. De la même façon, dans un système complexe, on analysera chacun des sous-systèmes selon le triplet : sorties, entrées, opérations et on

portera une attention toute particulière aux facteurs affectants le fonctionnement de l'ensemble du système.

Il faut, d'autre part, étudier les notions d'état et d'événement qui sont fondamentales pour analyser le comportement d'un système dans la mesure où ils permettent sa représentation dans le temps. Dans le cadre de la simulation à événements discrets, un état est une collection de variables, qui peut être ou non exhaustive, décrivant le système à tout moment alors qu'un événement correspond à l'occurrence instantanée d'une activité modifiant l'état du système. Nous allons à présent définir les principales voies que peut prendre la simulation pour représenter l'évolution des activités dans le temps, et les différentes formes d'implémentation correspondantes.

1.2. LES DIFFERENTS TYPES DE SIMULATION

La simulation sur ordinateur peut prendre diverses formes, notamment en fonction de la représentation du temps. On parle d'horloge discrète ou continue selon que le déroulement de la simulation se fait de façon événementielle ou par intervalle de temps régulier[1]. Dans le premier cas (celui que nous allons aborder dans cet état de l'art) l'horloge est modifiée par l'occurrence d'un événement alors que dans le cas d'une horloge continue on simule pour un « grain de temps » spécifique (par exemple pour une continuité machine, on prendra un petit temps machine de moins d'un jiffy, 1 jiffy= 10ms), c'est-à-dire que la progression de la simulation se déroule selon un certain laps de temps constant et défini pour imiter la notion de continuité. La Figure 2 illustre les deux trajectoires que peuvent prendre l'analyse des états du système.

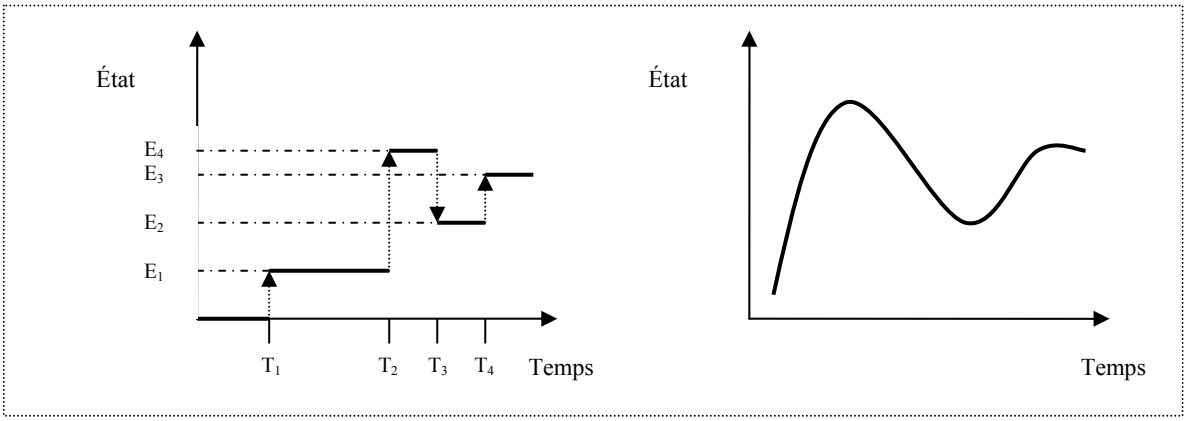


Figure 2 : ***Trajectoire discrète ou trajectoire continue.***

Si l'horloge de simulation est continue, on aura alors une approche fonctionnelle très pratique dans certains domaines, cependant ce type de simulation peut poser problème dans l'ordonnancement des actions et peut être coûteuse en terme d'exécution (effectivement même si aucun événement n'a lieu durant le laps de temps défini, des calculs sont générés). L'approche événementielle garantit, quant à elle, la causalité (le futur et le présent ne peuvent influencer le passé) et l'implémentation d'une telle simulation est très simple à mettre en

[1] Le domaine des variables d'intérêt peut être discret ou continu (par exemple on définira l'état d'un serveur par occupé ou libre- discret- alors que la mesure de probabilité que ce serveur soit occupée appartient à [0, 1] –continu) ; à ne pas confondre avec la définition de l'horloge.

œuvre. En effet il suffit de fragmenter le code en routines rattachées aux événements définis dans le modèle.
On parle de simulation équationnelle lorsque le code est basé sur des équations de récurrences et que par conséquent l'évolution du modèle se fait par une suite matricielle accélérant ainsi le temps d'exécution.
La simulation peut également être orienté processus, le code se présente ici comme une structure compact, et il s'agit en fait d'un mécanisme de co-routines (ou processus légers) exécutés quasiment en parallèle avec l'utilisation de sémaphores pour les interactions entre routines, et de primitives pour le noyau de simulation.
L'approche orientée processus correspond le plus souvent à une simulation distribuée (ou parallèle), se déroulant sur plusieurs processeurs ou machines, qui permet de traiter des modèles plus larges grâce à une exécution plus rapide.
La simulation distribuée (SD) peut se révéler extrêmement intéressante pour des systèmes eux mêmes répartis, on peut imaginer dans le cadre de réseau à l'échelle Internet que chaque machine impliquée dans la SD modéliserait un système autonome (AS) et que le protocole BGP (Border Gateway Protocol) serait « réel » entre les AS simulés. L'avantage d'une SD n'est donc plus uniquement sur le jeu d'exécution mais également sur la qualité des résultats recueillis grâce à une modélisation plus en adéquation avec le système étudié.
Dans le cadre de cet état de l'art nous étudierons la simulation séquentielle à événements discrets.

1.3. DOMAINES D'APPLICATION ET ALTERNATIVES

La simulation est un outil qui, notamment grâce à la puissance de calcul des ordinateurs actuels, est de plus en plus utilisé et dont les possibilités en terme d'applications deviennent réellement intéressantes, en particulier pour des systèmes évolués, c'est-à-dire dont le comportement est caractéristique d'une multitude de paramètres. En effet elle ne se limite plus aujourd'hui à de simples modèles, mais à des modèles représentant des systèmes complexes dont l'expression analytique est difficile, voire impossible, en raison du grand nombre de paramètres à prendre en compte dans le système étudié.
Les domaines d'application de la simulation sont donc extrêmement variés. Ils s'étendent de l'économie aux applications militaires, en passant par l'analyse de système informatique.
Dans ce document nous apporterons un intérêt tout particulier au domaine des communications sur des réseaux de type IP, sujet sur lequel il existe de nombreux simulateurs comme VINT/nS [17], PROSIT [5] (Voir chapitre 5)

Pourtant, la simulation est souvent critiquée par rapport aux approximations qu'elle engendre dans le processus de modélisation, et surtout pour les procédures statistiques approximatives à mettre en place pour réévaluer les mesures effectuées en sortie.
Il existe un certain nombre d'alternatives comme les chaînes de Markov, dont les qualités théoriques, notamment en terme de calcul de moyenne, lui sont supérieur, mais dont les applications sur des systèmes complexes restent limitées.
En effet, il suffit d'étudier la complexité de telles techniques pour exprimer le comportement, à priori simple, d'une file d'attente de type FIFO [4] ou le comportement d'un flux TCP [3] pour comprendre que cette voie analytique ne peut pas s'appliquer sur des systèmes complexes. Par exemple, pour la modélisation de réseaux de commutation de paquets, cette solution analytique n'est pas envisageable dans la mesure où ces deux modèles s'inscrivent justement dans un modèle beaucoup plus large, lui même composé de multiples files

d'attentes et sur lequel on peut appliquer des mécanismes de flots type TCP. On peut donc plutôt parler de complémentarité que d'alternative à la simulation.
Par ailleurs, l'émulation, ou la reproduction semi physique du modèle (au sens où on incorpore des éléments réels au modèle), peut également présenté des qualités de réalisme en terme de comportement.
Le document [16] définit le fonctionnement d'un émulateur pour réseau filaire et sans fil qui présente l'avantage d'avoir une architecture distribuée (au sens où elle est construite sur plusieurs machines reliées par un switch, les paquets transitants ainsi sur de « véritables liens »). Cependant les modèles de trafic de fond suivent des processus aléatoires (ce qui demande, on le verra plus tard dans le chapitre 4, que les mesures en sortie soient affinées) et l'émulation des communications sans fil nécessite l'utilisation de scénarii prédéfinis comparables à un processus de simulation à événements discrets.

La voie de la simulation est donc inévitable (à moins de reproduire entièrement le système étudié), bien qu'il faille mettre en œuvre des précautions à 3 niveaux :

- Qualité de la génération de nombre aléatoire pour les systèmes stochastiques.
- Qualité du modèle employé, c'est-à-dire les données en entrée.
- Qualité du traitement statistique à mettre en place sur les données en sortie.

C'est pourquoi nous allons détailler dans le chapitre qui suit comment ces facteurs de qualité entrent en jeu dans une simulation à événements discrets.

2. LA CREDIBILITE DE LA SIMULATION MISE EN DOUTE

La simulation stochastique à événements discrets est devenue aujourd'hui un outil incontournable pour les scientifiques et les ingénieurs, à tel point que plus de 50% des publications, dans la communauté réseau par exemple, font appel aux résultats d'une telle simulation. [1]

Pourtant cet enthousiasme n'est pas partagé par tous. En effet, l'utilisation de simulateurs stochastiques nécessite la génération de nombres pseudo aléatoires dont il faut s'assurer que l'uniformité de la distribution soit acceptable et surtout que la longueur du cycle soit suffisante pour la simulation.

Par ailleurs, les résultats obtenus doivent être analysés de façon appropriée.

Or dans la plupart des articles de recherche, ces deux facteurs ne sont absolument pas détaillés alors même qu'ils constituent pour une grande part la validité d'une simulation.

L'étendue du phénomène est si large qu'on ne peut nier les conséquences de ce problème dans ce domaine. Nous allons donc discuter des propriétés d'un « bon » générateur de nombres aléatoires puis de la qualité du modèle (usant de cette pseudo génération) et des démarches à mettre en œuvre afin de garantir la crédibilité d'une simulation stochastique.

2.1. SOURCES ELEMENTAIRES DE DISTRIBUTION UNIFORME

L'utilisation de générateurs de nombres pseudo aléatoires comme source de distribution uniforme est communément admise. Le type de générateur le plus populaire est sûrement la classe des MLC-PRNGs(*multiplicative Linear Congruential Pseudo Number Random Generators*) basé sur la congruence multiplicative linéaire et utilisant donc un modulo entier M. Pour des ordinateurs 32-bit, le modulo $M=2^{31}-1$ a été la cible d'attention particulière et est considéré comme générant une distribution uniforme (dont les nombres générés sont indépendants et aléatoires) acceptable. Ces générateurs ont été utilisés, par exemple, dans des simulateurs comme GPSS, SIMSCRIPT II.5, SIMAN et SLAM II.[21]

Cependant avec l'avancée des technologies actuelles, il suffit d'un processeur de quelques Mégahertz pour que en quelques minutes la suite de congruence M d'un tel générateur soit entièrement parcourue, et donc que la simulation suive des cycles identiques de nombres aléatoires. Dans le cadre des simulations nécessitant une passe relativement longue, dans la mesure où la récolte des résultats devra être représentative et donc probablement longue, le cycle du générateur va être dépassé, ce qui va mettre en péril l'ensemble des résultats. [1] (c'est le cas par exemple d'une simulation collectant des données sur un réseau dont le trafic est déterminé par une distribution spécifique inter arrivé)

En outre, les simulateurs stochastiques ont souvent plusieurs paramètres décrits par des distributions usant de nombres pseudo aléatoires et cela restreint d'autant plus l'utilisation de générateurs MLC-PRNGs à petit cycle.

Il existe à présent, grâce aux travaux récents dans ce domaine, des générateurs dont la durée du cycle est suffisamment longue pour supporter des simulations demandant un temps d'exécution processeur très long. Il s'agit de générateurs multiples récursifs LC-PRNGs et multiple récursif LC-PRNGs (*Linear Congruential*) combinés dont le cycle est entre 2^{185} et 2^{377} [22] qui satisfont des simulateurs à 32 dimensions (c'est-à-dire à 32 paramètres aléatoires de simulation).
Une autre classe de générateurs basés sur la récursivité en polynôme arithmétique et connu sous le nom de GFSR PRNGs (*Generalized Feedback Shift Register*) peut atteindre des cycles d'une durée de 2^{19937} et ainsi satisfaire des simulations à 623 dimensions ![23]
Avec de telles générateurs de qualité on peut atteindre des cycles d'une durée importante, néanmoins il faut se méfier des corrélations potentielles entre sous séries disjointes de nombres consécutifs que peut engendrer l'utilisation d'un même PRNG dans une simulation distribuée et/ou parallèle.
Dans le cas d'une simulation séquentielle sur un seul processeur il faut veiller à la qualité du générateur utilisé et par conséquent préférer des générateurs reconnus pour leurs propriétés à une distribution incontrôlé qui aboutira à des résultats périodiques et faussées.

2.2. DONNEES EN ENTREE, DONNEE EN SORTIE

Une issue fondamentale de la pertinence des résultats obtenus sur certains objectifs d'étude lors d'une simulation est la validité du modèle, en d'autres termes les données soumises en entrée du simulateur. En effet il faut que les hypothèses de simulation sur le système étudié soient réalistes. Dans le cadre de la simulation de réseau de télécommunications il faut notamment s'assurer que les mécanismes internes de routage soient correctement modélisés, que les caractéristiques des processus stochastiques soient reconnues (voir section 3.2.1), et que de manière générale les simplifications duent à la modélisation du système ne produisent pas un modèle éloigné de la réalité du système étudié.
Dans la partie 3, nous allons définir les propriétés d'un bon cadre de développement pour modèle encourageant la validation et les vérifications sur celui-ci. Cependant il s'agit juste d'outils de travail, or la qualité du modèle réside surtout dans son aptitude à reproduire de façon crédible le système sur lequel on souhaite réaliser des expériences.

D'autre part les résultats obtenus sur ces expériences ne peuvent être analysés directement, il est impératif de réaliser un traitement statistique approprié selon le type (aléatoire ou non) de données utilisées en entrée.
Von Neumann [34] a d'ailleurs noté la similitude entre une simulation stochastique sur ordinateur et le jeu de la roulette, d'où le nom de *Simulation Monte Carlo* pour désigner un simulateur stochastique.
Afin de définir l'inévitable erreur statistique (on parlera également d'erreur relative définie dans la section 4.1) portée par les résultats d'une expérience sur une simulation Monte Carlo, la section 4 a pour but de mettre en avant les principaux outils de mesure de confiance. La Figure 3 tirée de [1] illustre l'importance de la mesure de l'erreur sur la qualité et la pertinence des résultats.

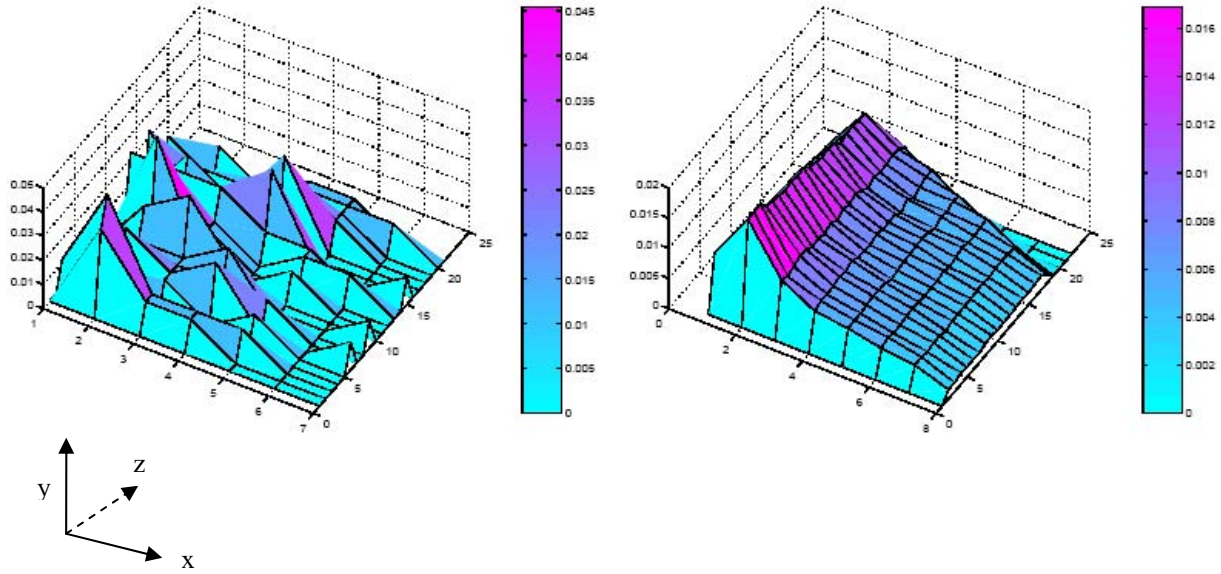


Figure 3 : ***L'erreur relative influe sur les résultats.***

Avec :
x :gigue ; y :probabilités ; z :longueur des segments(paquets MAC).
EVALUATION protocole MAC sur un réseau de communication mobile.

a. (A gauche)Résultats avec une erreur relative de 25% ou moins.
b. (A droite)Résultats avec une erreur relative de 1% ou moins.

Niveau de confiance choisi =90%.(voir section 4 pour les définitions).

On peut constater de manière significative l'importance de la durée de simulation dans la mesure où celle-ci dépend directement de l'erreur relative qui diminue puis ce stabilise au cours du temps de simulation. (On peut donc en déduire que la figure a est une passe de simulation plus courte que b.)

2.3. LA SIMULATION DE RESEAUX DE COMMUNICATION

Bien que la plupart des chercheurs en informatique et des ingénieurs en télécommunications aient de réelles notions statistiques pour contrôler et minimiser l' inévitable erreur associée aux mesures due à la nature stochastique d'une simulation de réseaux de communications, une large proportion des publications récentes dans ce domaine ne tient absolument pas compte, dans le report de leurs résultats issus de telle simulation, de ces traitements statistiques spécifiques indispensables à leur crédibilité.

En effet selon [1], on peut observer que 76.6, 79.05 ,71.6, et 68.6 % des papiers relatant de résultats basés sur des simulations et respectivement publié dans *Proceedings of* IEEE INFOCOM, IEEE *Transactions on Communications*, IEEE/ACM *Transactions on Networking*, et *Performance Evolution Journal* ne nous informe pas sur les outils de confiance employés dans l'analyse de leurs résultats. D'autre part la grande majorité de ces publications ne précisent même pas la nature (finie ou non) de la simulation utilisée pour aboutir à ces résultats.

De manière générale la Figure 4 met en évidence que plus de 75% des papiers publiés ne semblent pas concernés par la nature aléatoire de la simulation sur laquelle ils fondent leurs observations [1]!

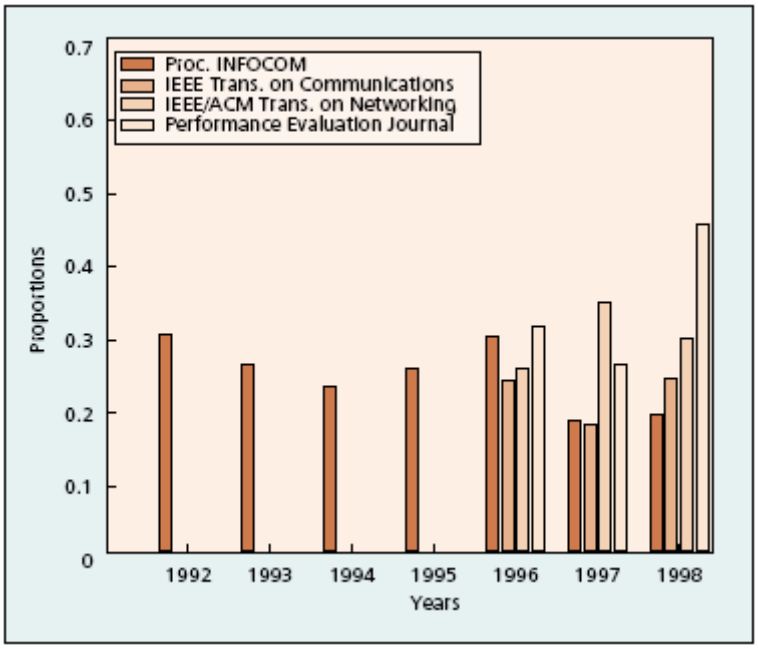


Figure 4 : ***Taux de publications où mention est faite de l'analyse des données en sortie par rapport à l'ensemble des publications usant de simulations stochastique.***

On peut espérer qu'il s'agisse juste d'un manque d'attention et que leur simulation soit tout de même sujette à des traitements statistiques appropriés bien qu' aucune mention n'en soit faite, cependant cette pratique est inacceptable dans la mesure où la description des techniques de simulation devrait être systématique. La complexité intrinsèque de telle simulation, non seulement en terme de récoltes de données mais aussi en terme de qualité de modélisation, en fait des systèmes extrêmement fragiles dans le sens où les effets de bord sont fréquents. Par conséquent il est du devoir d'un auteur d'être très précis dans la description du type et de la méthodologie de simulation employée (52 % des articles ne précisent pas si la simulation est à horizon fini ou non) et de mettre en avant la nature du traitement statistique utilisé (séquentielle ou parallèle si la quantité d'informations recueillis le permet) ainsi que la manière dont il a été mis en oeuvre.

Il est à noter que cette crise de crédibilité ne se limite pas aux domaines des télécommunications mais s'étend à toute l'informatique aussi bien en électronique qu'en ingénierie alors qu'en physique ou en biologie ces principes de confiance sont respectés.
La conséquence d'un tel laxisme peut se révéler désastreuse malgré le signal d'alarme lancé dans les années 90 par B.Gaither [24], rédacteur en chef du ACM *Performance Evolution Journal*; l'exemple de AT&T's à travers l'échec de leurs protocoles réseaux longue distance en donne la mesure.

Avant d'être pris au sérieux, les résultats d'une simulation doivent être soumis à un examen minutieux de la part de leur concepteur tant au niveau de l'implémentation même du simulateur:

- validité du modèle (génération de trafic, processus d'arrivée …).
- qualité du PRNG employé (particulièrement si l'on simule un système à horizon infini car la durée de simulation tend à être conséquente au vu des méthodes statistiques à déployées).

qu'au niveau du traitement statistique:

- méthode utilisée (réplications, moyenne des lots…).
- intervalle et niveau de confiance choisi.

L'auteur d'un article traitant des résultats d'une simulation stochastique se doit de faire partager à ses lecteurs ces deux principaux facteurs de crédibilité du fait de leur impact sur la qualité et la précision des résultats présentés.

3. DU SYSTEME AU MODELE: FORMALISME ET METHODOLOGIE

3.1. EVALUATION DES METHODES

Lorsque que l'on souhaite analyser un système complexe, dans le cadre de la simulation à événements discrets, la méthode consiste à construire un modèle représentant, souvent de façon simplifiée, le système étudié. Dans ce chapitre l'objectif est de mettre en avant dans un premier temps les conditions nécessaires à un bon cadre de modélisation puis de décrire les principales approches formelles respectant les critères ainsi définis et évaluer leurs performances pour conceptualiser, représenter, analyser et pour finir, implémenter ce type de modèle.

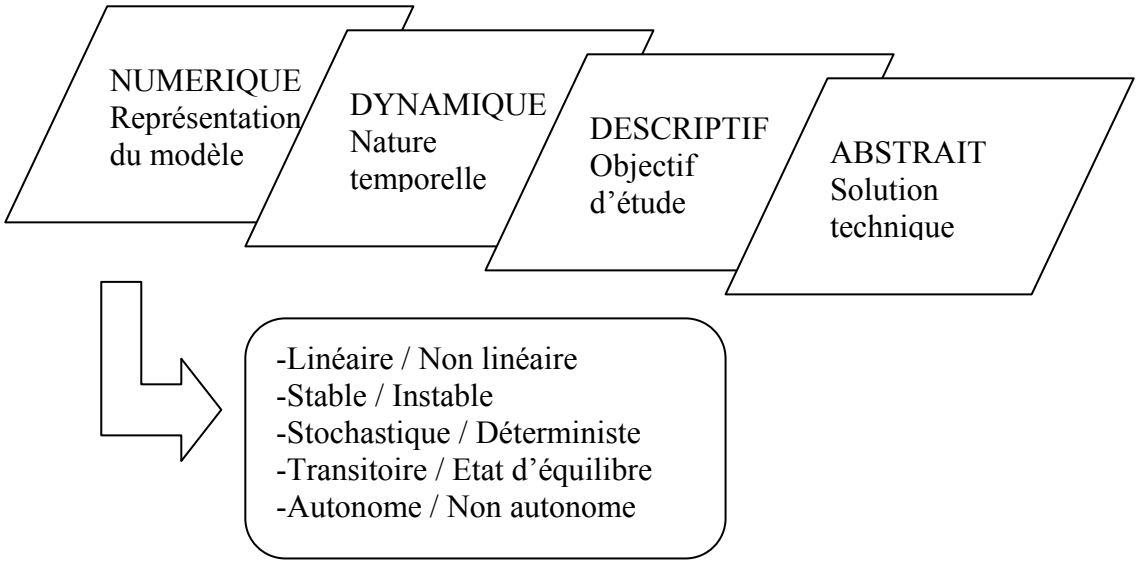


Figure 5 : ***schéma descriptif d'un simulateur à événements discrets dérivé de [2]***

La Figure 5 ci-dessus décrit une classification en 4 dimensions de la modélisation d'un système.

Un modèle abstrait se caractérise par une description en langage naturel ou par le symbolisme des termes mathématiques à la différence d'un modèle physique qui est un « clone » du système d'origine.

La seconde dimension décrit les objectifs de l'étude sous jacente au modèle : un modèle descriptif décrit le comportement du système sans critère de jugement alors qu'un modèle prescriptif décrit le comportement du système en termes qualitatifs.

Une troisième dimension identifie la nature temporelle du modèle, celle-ci peut être dynamique si le système évolue au cours du temps ou statique si le modèle décrit un système immuable.

La dernière dimension spécifie le coté technique du modèle. Elle est soit analytique dans le cas de déductions mathématiques soit numérique dans le cadre de procédures de calculs sur ordinateur.

La simulation à événements discrets se situe donc dans la classe des modèles abstraits, dynamiques, descriptifs et numériques.

D'autre part comme illustré dans la Figure 5 la simulation à événements discrets est définie également par la combinaison de certaines caractéristiques :

- un modèle linéaire est décrit par des relations linéaires (à l'opposé d'un modèle non linéaire).
- un modèle stable tend à retourner dans son état initial si il est perturbé (à l'inverse d'un modèle instable).
- un modèle à état d'équilibre a le même comportement sur une période donnée qu'à toute autre période alors qu'un modèle transitoire à un comportement qui change avec le temps.
- un modèle stochastique admet en entrée au moins une variable dont le comportement est aléatoire sinon le modèle est déterministe.
- un modèle autonome ne nécessite ou n'accepte pas de données en entrée de l'environnement en cours de simulation tandis qu'un modèle non autonome peut recevoir de nouvelles entrées en dehors de la période d'initiation.

Nous allons également étudier dans ce chapitre une méthodologie complète pour la modélisation des composants et du comportement d'un système qui intègre directement des outils abordant des notions de vérification et de validation du modèle généré.

3.1.1. Conditions pour un bon cadre de travail

Dans sa thèse E.Page [2] rassemble un certain nombre de critères dégagés par Nance et Sargent ([41] et [42]) afin de définir les conditions essentielles à un développement adéquat du modèle représentant le système. Le tableau ci-dessous met en avant ces différents critères.

	Le cadre de travail doit :
1.	encourager et faciliter la production de modèles et l'étude de documents, particulièrement au niveau des définitions, hypothèses et objectifs.
2.	permettre de décrire le modèle à partir du niveau le plus haut jusqu'au niveau le plus bas.
3.	rester fidèle au modèle du niveau le plus haut au niveau le plus bas.
4.	être discret, et/ou le support doit être fourni pour de multiples cadres conceptuels.
5.	structurer le développement du modèle et gérer la description, le niveau de fidélité et le choix du cadre conceptuel adapté au modèle.
6.	répondre à un large domaine d'applications.
7.	permettre au modèle d'être indépendant de l'architecture et du langage d'implémentation.
8.	définir le support de l'environnement et encourager la conversion automatique vers le code utilisé.
9.	fournir un large support technique de vérification et de validation pour les modèles étudiés.
10.	faciliter l'utilisation des composants (réutilisables) et des expériences sur le système.

Tableau 1 : Conditions nécessaires pour un bon cadre de développement de modèle.

Le développement d'un « bon modèle » nécessite donc un environnement d'abstraction adapté au problème. Ceci nous conduit à définir trois niveaux de représentation du modèle :

- spécifications générées par le modélisateur. Il s'agit de la partie où les définitions, hypothèses, comportements et objectifs du système pour une étude donnée sont décrits. La nature de ce niveau de représentation est à la guise du modélisateur.
- spécifications transformées. Ce niveau est généré de manière automatique ou semi-automatique pour permettre l'analyse et la translation vers le code selon des critères spécifiques.
- implémentation. Il s'agit ici du plus bas niveau de spécifications transformées c'est-à-dire le code satisfaisant les différents niveaux de contraintes spécifiques d'une simulation donnée.

3.1.2. Formalisme pour modèles à événements discrets

Dans cette section nous passerons en revue différentes approches permettant de représenter un système à événements discrets. Les approches existantes dans ce domaine sont par nature formelles dans la mesure où cela permet de travailler dans un cadre précis et non ambigu en accord avec les conditions définies précédemment. Cependant, ces approches peuvent prendre des formes très diverses et variées allant d'un formalisme très intuitif (souvent sous forme de graphe) à un formalisme purement mathématique. Nous allons aborder deux formalismes représentatifs de ces variations. Dans un premier temps nous allons étudier le formalisme de spécification de système à événements discrets dans la mesure où il semble le plus valable au regard des critères définis dans le Tableau 2, et dans une seconde partie, les techniques graphiques orientées événements dont nous étudierons quelques applications dans le chapitre 5.2.

3.1.2.1. Formalisme de spécification de système à événements discrets

De nombreuses approches et méthodes (dont celle de Lackner [25] avec le « Change Calculus ») pour la modélisation trouve leurs origines dans la *théorie générale des systèmes* (Voir [2]). Celle ci affirme que les systèmes réels obéissent aux mêmes lois et expriment des similitudes dans leurs comportements même s'ils sont physiquement différents.
Basé sur cette théorie, Zeigler [26] a créé un formalisme de spécification de système à événements discrets (DEVS) qui est aujourd'hui le plus répandu dans le cadre de la simulation à événements discrets.

Cette théorie lui a permis d'établir une hiérarchie de spécifications du système :

1. spécification des relations entrée-sortie.
2. spécification des fonctions entrée-sortie.
3. spécification générale.
4. spécification structurée.
5. spécification en terme de réseau.

Chacun de ces niveaux peut être automatiquement mis en relief à partir de DEVS et d'une collection de morphisme d'équivalence.
DEVS identifie trois concepts élémentaires majeurs pour la simulation à événements discrets :

1. le système.
2. le modèle.
3. l'ordinateur.

Deux relations sont également décrites :

1. la modélisation : relation système/ modèle.
2. la simulation : relation modèle /ordinateur.

Un système est une source potentielle de « données comportementales » consistant en des marqueurs XT, où X est une variable d'intérêt et T représente le temps. Le modèle est vu comme un ensemble d'instruction décrivant ces « données comportementales » qui seront finalement générées par l'ordinateur.

La théorie générale des systèmes nous fournit cinq catégories de modèles :

A. modèle discret / modèle continu par rapport à la base de temps.
B. modèle discret / modèle continu par rapport aux variables d'état descriptives.
C. modèle déterministe / modèle stochastique.
D. modèle autonome / modèle non autonome par rapport aux effets potentiels de l'environnement.
E. modèle à temps variable / modèle à temps invariant.

D'autre part la relation de modélisation contribue à la validité du modèle. Ainsi la nature du modèle obtenu peut être:

- Valable en réplication, si le modèle correspond aux données déjà acquises du système.
- Valable en prédiction, si le modèle correspond aux données précédents celles générées par le système réel.
- Valable structurellement, si le modèle ne reproduit pas seulement le comportement du système mais reflète vraiment la voie par laquelle le système réel opère pour produire ce comportement.

La relation de simulation quant à elle concerne la façon dont l'ordinateur traite les instructions fournies par le modèle.

Définition du modèle

La définition du modèle est basée en partie sur les affirmations suivantes :

a- la simulation doit être gérée par une liste d'événements (échéancier) contenant les horaires auxquels les composants du système subissent des changements d'état.
b- L'échéance de certains événements est prévisible sur la base des résultats de l'occurrence d'autres événements, il est ainsi placé sur l'échéancier.

c- Si la prochaine échéance événementielle d'un composant n'est pas prévisible à l'avance, il ne peut subir de changements d'états jusqu'à ce que et à moins qu'un tel changement soit causé par une transition d'état d'un composant qui a été préprogrammé.

Modèle atomique

Un modèle atomique M est donnée par :
$M=<X, S, Y, \delta_{int}, \delta_{ext}, \lambda, t_a>$
où
X : ensemble d'événements en entrée.
S : ensemble d'états séquentiels.
Y : ensemble d'événements en sortie.

δ_{int} : $S \rightarrow S$ fonction de transition interne.
δ_{ext} : $Q \times X \rightarrow S$ fonction de transition externe.
λ : $S \rightarrow Y$ fonction de sortie.
t_a : $S \rightarrow R^+$ fonction « de saut dans le temps ».

Modèle couplé

Les systèmes complexes nécessitent pour leur modélisation d'interconnecter ensemble plusieurs modèles atomiques créant ainsi un modèle couplé.

Un modèle couplé DN est donnée par :
$DN= <D, \{M_i\}, \{I_i\}, \{Z_{i,j}\}, SELECT >$

où
D ensemble des noms des composants.
Pour chaque i dans D (1) M_i = modèle atomique pour le composant i dans D.
(2) I_i = ensemble d'influence de i (attributs sur lesquels il agit).
Pour chaque j dans I_i $Z_{i,j} : Y_i \rightarrow X_i$; fonction de translation en sortie de i vers j.
SELECT soit $E \subset D \rightarrow D$, alors pour tout ensemble $E \neq \varnothing$, $SELECT(E) \in E$; sélecteur de rupture/attachement (pour joindre ou disjoindre deux modèles atomiques).

Le fait de connecter les ports E/S de plusieurs modèles atomiques pour spécifier un modèle complexe se nomme *accouplement modulaire*. De même on peut également interconnecter des modèles couplés par leurs ports E/S pour créer un modèle couplé encore plus large faisant référence à une modélisation hiérarchique.
Praehofer et Pree [27] ont mis en avant que dans le formalisme DEVS la notion d'*état* est proéminente à la différence des langages et approches traditionnels pour la simulation à événements discrets qui se basent sur les concepts d'événement, d'activité ou de processus. Le comportement dynamique du système est organisé autour de la notion de variable de *phase* dénotant l'état global courant du système, ainsi selon la phase actuelle le système réagira

différemment aux événements internes ou externes. Dans la modélisation DEVS, la phase courante définit alors une partition de l'espace des états possibles du modèle.

Il est à noter que l'approche DEVS trouve des applications dans des domaines de recherche différents de la simulation à événements discrets comme l'intelligence artificielle ou la simulation continue.

Cependant il existe des variations orientées objets de cette approche comme HON (Hierarchical Objet Nets, voir [2]) qui sont destinées uniquement au domaine discret à la différence d'extensions de DEVS comme DEVandDESS [27] qui s'appliquent aussi bien à des simulations continues, discrètes ou combinées.

3.1.2.2. Techniques graphiques orientées événements

Graphe d'événements

Les techniques graphiques développées pour la simulation à événements discrets basées sur l'analyse d'activité ou l'interaction de processus ne propose pas d'outils pour modéliser le déroulement des événements, Schruben [28] a donc introduit un formalisme de *graphe d'événements*. Il s'agit d'un graphe orienté décrivant les relations inter événements (ils sont numérotés) et dont les éléments cruciaux sont les *variables d'états* qui décrivent l'état du système, les *événements* qui modifient ces mêmes variables, et les *relations logiques* et *temporelles* qui relient ces événements. Les sommets du graphe représentent les événements et les arcs indiquent la façon dont les événements influencent l'occurrence des autres.

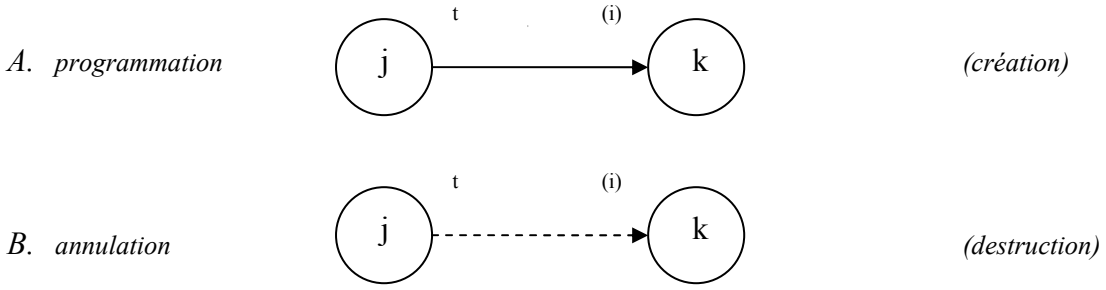


Figure 6 : ***Deux types de relation inter événements.***

Il existe deux types d'arcs :

La figure A exprime le fait que après t unité de temps de l'occurrence de l'événement j, l'événement k va être programmé pour se déclencher à la condition (i) et sous réserve que (j) se soit déroulé.

La figure B indique qu'après t unité de temps de l'occurrence de l'événement (j), toute occurrence actuellement programmée de l'événement (k) sera annulée à la condition (i) et sous réserve que j se soit déroulé.

D'autres propriétés d'un graphe d'événement sont :

1. deux sommets peuvent être joint par plusieurs arcs, et les boucles sont permises.
2. les arcs sans label de condition (i) sont des arcs inconditionnels.
3. les labels de temps et conditions peuvent invoquer des variables aléatoires.
4. les sommets et les labels de conditions peuvent être paramétrable pour simplifier le graphe.
5. un graphe d'événements n'est pas nécessairement connecté.

(Voir 5.2.4 pour un exemple d'application).

Graphe de simulation

Yücesan [29] a mis en avant la possibilité de généraliser les graphes d'événements créant ainsi un graphe de simulation, variante d'un graphe d'événements à cela près qu'il constitue un mécanisme de représentation indépendant du monde réel.
Un graphe de simulation G est donné par un quadruplet :

$G=<V(G), E_s(G), E_c(G), \Psi_G>$
où
$V(G)$ ensemble de sommets/événements.
$E_s(G)$ ensemble d'arcs de programmation.
$E_c(G)$ ensemble d'arcs d'annulation.
Ψ_G fonction d'incidence.

Les données définies dans le graphe sont données par l'ensemble d'indexes suivants :
-$F = \{f_v : \text{ETATS} \rightarrow \text{ETATS} \mid v \in V(G)\}$; ensemble de fonctions de transitions d'états associées au vecteur v.
-$C = \{C_e : \text{ETATS} \rightarrow \{0,1\} \mid e \in E_s(G) \bigcup E_c(G)\}$; ensemble des arcs conditionnels.
-$T = \{\gamma_e : \text{ETATS} \rightarrow R^+ \mid e \in E_s(G)\}$; ensemble des arcs de délais.
-$\Gamma = \{\gamma_e : \text{ETATS} \rightarrow R^+ \mid e \in E_s(G)\}$; ensemble des priorités des exécutions événementielles.

ETATS étant défini comme dans le développement de Zeigler [26].
Un modèle de graphe de simulation (SGM) est donné par :

$$S = (F, C, T, \Gamma, G)$$

G, dans la définition de S, joue un rôle analogue à Ψ dans la définition d'un graphe orienté, c'est-à-dire qu'il organise et spécifie les relations entre les différents éléments des ensembles F, C, T et Γ.

3.1.3. Evaluation et autres Formalismes

Afin d'évaluer les formalismes décrits précédemment, dans sa thèse [2] E.page a définit une échelle de note allant de 1 à 4 avec comme système de notation (Tableau 1):
1 = non reconnu.
2 = reconnu mais pas démontré.
3 = partiellement démontré.
4 = complètement démontré.

Le tableau ci-dessous rassemble les notes obtenues pour le formalisme de Lackner et sa théorie « change calculus » (CC) dans la mesure où il fut l'un des pionniers dans ce domaine, ainsi que la spécification de systèmes à événements discrets (DEVS), les techniques graphiques orientées événements (TGOE) et la méthodologie conique associée à la spécification de condition (CM/CS, voir section 3.1.4). Pour le détail des critères, il faut se reporter au Tableau 1.

Critères	**CC**	**DEVS**	**TGOE**	**CM/CS**
Documentation	**2**	**2**	**2**	**3**
Description	**2**	**2**	**3**	**2**
Fidélité	**1**	**2**	**3**	**3**
Discrétion	**2**	**1**	**2**	**2**
Structurelle	**2**	**3**	**2**	**3**
Domaines d'applications	**2**	**4**	**2**	**3**
Indépendant du langage de programmation	**3**	**3**	**3**	**3**
Facilité de la conversion automatique	**2**	**3**	**3**	**3**
Validation et vérification	**2**	**3**	**3**	**4**
Réutilisable	**1**	**4**	**3**	**3**

Tableau 2 : Evaluation des formalismes.

Il existe bien entendu d'autres formalismes de type graphique comme les diagrammes de cycles d'activité, les réseaux de Petri ou les graphes de contrôle de flots (Voir [2]) cependant ils obtiennent de moins bons résultats que les techniques graphiques orientées événements. De même des formalismes purement mathématiques comme l'approche basé logique ou les processus de Markov semi généralisés (à ne pas confondre avec des processus de Markov purs qui sont une alternative à la simulation) obtiennent des résultats moindres que la théorie générale des systèmes avec DEVS. ([2])
Il est à noter qu'il existe une variante des réseaux de Petri : les réseaux de Petri stochastique décrit dans [2], qui semblent toute particulièrement adaptée au simulation comme QoSim dans la mesure où les temps de transition (entre flots pour QoSim) sont distribués de façon exponentielle, pourtant ce formalisme semble faible aux égards de nombreux critères ([2]).

3.1.4. Une méthode complète de modélisation

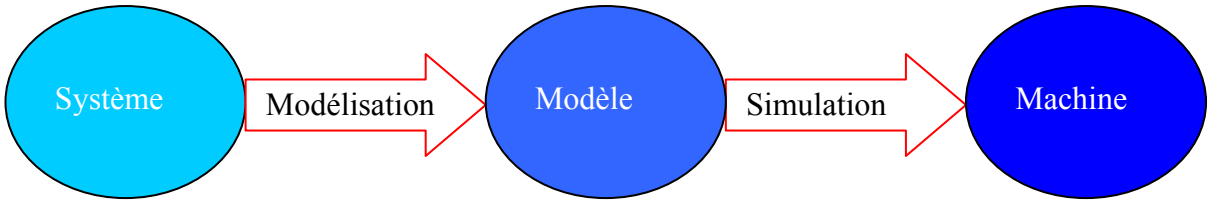


Figure 7*: schéma d'une procédure de simulation dérivé de la théorie générale des systèmes.*

Construction d'un modèle :

Dans cette section nous allons nous intéresser à deux aspects fondamentaux dans le développement du prototype d'un modèle :

- La méthodologie conique, qui fournit un support pour identifier l'ensemble des composants d'un système.
- La spécification de condition, qui permet de dégager les relations inters composants.

3.1.4.1. La méthodologie conique (« Conical Methodology », *CM*)

Cette méthode a été imaginée par Zeigler [26] et Nance [30] pour répondre aux problèmes techniques de modélisation. Il s'agit d'un cadre de travail basé sur la Théorie Générale des Systèmes (voir section 3.1.2) qui permet de décrire de façon conceptuelle un système dans l'intention de le simuler de manière discrète.
En pratique le développement d'un modèle, dans la CM, commence par un ensemble de définitions qui a pour but de réaliser une description complète, précise et non ambigu du système à simuler. La CM est en fait une aide à la modélisation qui se veut généralisable et qui a pour objectif :

a- d'assister le modélisateur pour structurer et organiser le modèle conceptuel.
b- d'imposer un développement axiomatique sous une apparence libre et non restrictive.
c- d'utiliser le diagnostique du modèle pour lui assurer une crédibilité par le biais de mesures de consistance et de complétude ainsi qu'une bonne planification à travers des mesures de complexité relative.
d- de produire une documentation majeur pour permettre d'étendre les descriptions du modèle.
e- de promouvoir et d'organiser des expériences et des tests sur un modèle expérimental.

La CM identifie deux niveaux dans le développement d'un modèle :

- la phase de définition (bas niveau).

- la phase de spécification (haut niveau).

Le schéma ci-dessous illustre la relation cyclique entre ces deux phases au cours de la construction du modèle.

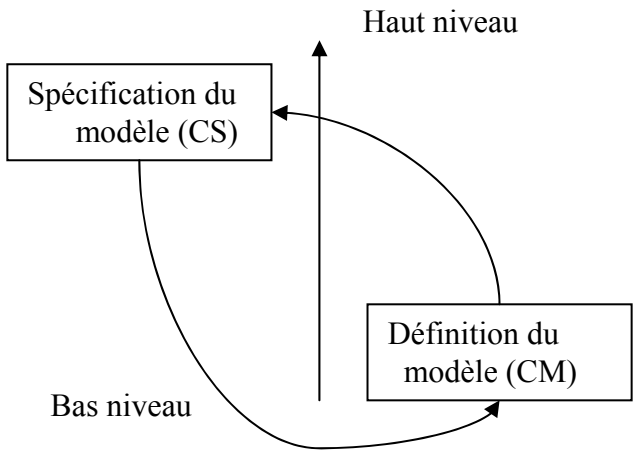


Figure 8 ***: Développement d'un modèle avec CM/CS.***

(Au cours de la modélisation on passe d'une phase à l'autre).

Définition du modèle

Durant cette phase, il s'agit de décomposer le système étudié en objets et sous objets puis nommer et typer leurs attributs. On peut représenter cette décomposition par un arbre dont la racine serait le modèle lui-même et les nœuds les sous objets à différents niveaux de détail. Un attribut contribue à la description d'un objet, et peut être de différentes natures :

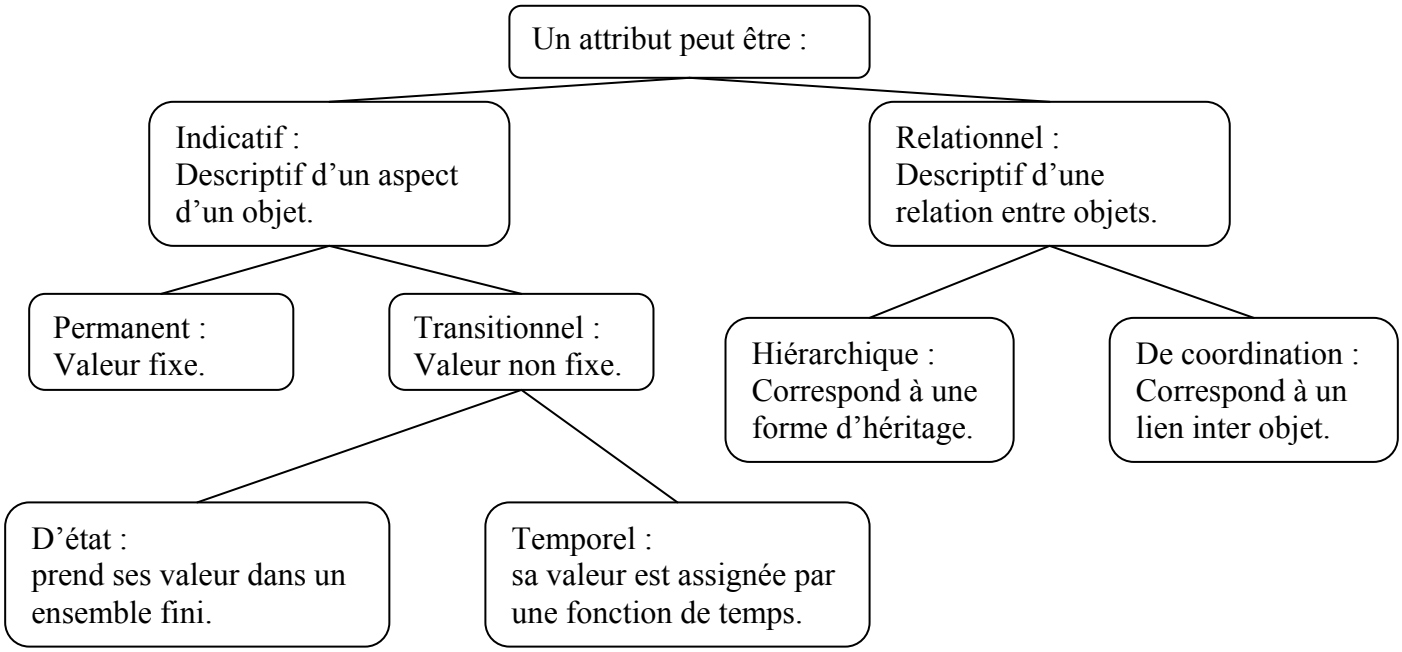


Figure 9 : ***Arborescence de la nature d'un attribut de simulation.***

La CM insiste sur un typage explicite des attributs dans la mesure où cela permet une analyse de spécifications extensible, de même que la dimension et la portée de celui ci permettent une meilleure description. Au minimum un attribut d'indexage est associé à tout modèle, typiquement il s'agit du temps au travers de l'attribut *system time*.
Dans une décomposition issue de CM la notion d'ensemble (*sets*) est capitale, il existe deux types *sets* qui vont permettre au modélisateur de définir les relations entre objets, les primitifs *(primitive set)* et les définis *(defined set).*
Tous les objets membres d'une *primitive set* ont exactement les mêmes attributs, ils sont donc de nature statique, alors que les objets constituant un *defined set* n'ont pas nécessairement les mêmes attributs et correspondent à l'évaluation d'une expression au cours de l'exécution du modèle et sont par conséquent dynamiques. Pour nommer un *set* ayant un ou plusieurs attributs relationnels qui permettent d'accéder à la valeur de l'attribut de tous les objets membre on parle de *set creation.* De la même façon *set deletion* efface la relation établie par au moins un attribut relationnel.

Nance considère cette méthodologie comme étant seulement une pièce -certes importante - dans un puzzle plus large. La CM propose juste un ensemble de principes dans des termes très généraux, chaque domaine d'application aura ses spécificités qui lui sont propres. Pour obtenir un modèle réellement représentatif et significatif l'environnement propre à la simulation utilisant des langages de spécifications et de documentations particuliers (SMSMDLs) aura un rôle important à jouer dans la modélisation. La CM ne se veut ni restrictive ni complètement explicite afin d'éviter une approche rigide.
Un exemple de CM est donné en 5.2.1.

Spécification du modèle

Il s'agit ici de décrire les aspects dynamiques du modèle, c'est-à-dire son comportement. Une fois que les objets sont définis sous forme d'arbre, le modélisateur doit étudier l'effet du changement de valeur d'un attribut appartenant à un objet sur la valeur des attributs d'autres objets.
La CM stipule que le SMSMDL doit :

a) Permettre l'indépendance entre les spécifications et les contraintes d'implémentations liées au langage de programmation utilisé pour la simulation.
b) Autoriser l'expression de propriétés statiques et dynamiques du modèle.
c) Faciliter la validation et les vérifications sur le modèle.
d) Produire de la documentation comme un sous-produit.

Ces conditions sont respectées par la méthode de spécification décrite dans la section qui suit.

3.1.4.2. La spécification de condition (*CS*)

Overstreet dans [31] définit un formalisme pour la spécification de modèle de simulation dans lequel le comportement du modèle a des propriétés utiles et désirables :

a) Il est indépendant des représentations traditionnelles du monde réel.
b) Il peut être analysé pour identifier les composants naturels mesurant la complexité et identifie les problèmes potentiels d'une spécification.
c) Certains aspects du modèle peuvent être négligés (ou plutôt moins bien spécifiés) sans pour autant entraver les analyses et transformations vu précédemment.
d) La définition du modèle n'est pas restreinte à un mécanisme d'implémentation particulier comme par exemple le *time flow mechanism.*

Le but de ce formalisme CS est de fournir une représentation sous forme de modèle indépendant du monde assez expressif pour représenter tout modèle et assez concis pour faciliter un diagnostique automatique de celui-ci.
Pour un CS donné on peut aisément identifier la nature dynamique d'un modèle qu'elle soit basé sur le temps (*time based*) ou sur des états (*state based*).

Overstreet favorise le diagnostic automatique du modèle au détriment de la souplesse de la syntaxe du CS ce qui entraîne que ce formalisme n'est pas utilisé directement par le modélisateur, mais qu'il faut intercaler un générateur de modèle entre le modélisateur et le bas niveau de syntaxe du CS.

Un *modèle de simulation* est un processus dans lequel une série de changements ordonnés interviennent au cours du temps, il est dit *modèle événementiel discret* si tout les attributs exceptés le *time based signal* change de valeur un nombre fini et dénombrable de fois.
Une *passe de simulation* correspond à une exécution unique d'un modèle de simulation produisant des données représentatives du comportement du modèle, tandis qu'une *expérience de simulation* est une collection de passe de simulation produisant un ensemble de données comportementales.
Le développement d'un modèle de simulation est selon l'approche d'Overstreet décomposable en deux phases :

- Spécification du modèle de simulation.
- Implémentation du modèle de simulation.

Spécification du modèle de simulation.

Une *spécification de modèle* est un quintuple MS : <Φ, Ω, Γ, τ, Θ> où :

Φ est la *spécification d'entrée*, c'est-à-dire les informations que le modèle reçoit de son environnement.

Ω est la *spécification de sortie*, il s'agit des informations que l'environnement reçoit du modèle. Ces attributs de spécification permettent de coordonner les différents composants d'un modèle M si le modèle spécifié appartient au modèle M plus large et/ou fournir un support pour les objectifs et la validation du modèle.

Γ est l'*ensemble de définition d'objet.*

La définition d'un objet est une paire ordonnée, <O, A (O)>, où O est l'objet et A (O) est *l'ensemble des attributs*. En cours de simulation plusieurs instances d'un même type d'objet peuvent exister, cependant on peut les distinguer au moins par une valeur d'un attribut.
Un *modèle d'ensemble d'attributs* A (M, t) est l'union de tous les attributs de tous les objets existants d'un modèle M au temps t.
L'*état d'un objet,* S (O, t) se défini par la valeur de tous ses attributs au temps t.
De la même façon l'*état du modèle* S (M, t) se définit par la valeur de tous les attributs existants dans A (M, t).
A (M, t) ne décrit pourtant pas de manière exhaustive l'ensemble des variables d'états, définies comme étant suffisamment complètes en terme d'information pour décrire entièrement le comportement du modèle.

L'ensemble A (M, t) doit être augmenté d'un certain nombre de « variables système » comme les horaires évènementiels pour constituer un véritable ensemble de variables d'états.

τ est l'*attribut d'indexage*. En général il s'agit du temps du système (*system time*). Celui ci permet d'ordonner partiellement les changements d'états durant une passe de simulation.

Θ est la *fonction de transition*, elle se contient trois phases :

1. L'*état initial* du modèle. Soit $t=t_0$ la valeur initiale du temps alors il s'agit de A (M, t_0) et d'au moins un événement déjà « préprogrammé ».
2. Une *condition de terminaison*.
3. La définition comportementale du modèle décrivant l'effet de chaque composant sur les autres, la réaction du modèle aux entrées ainsi que la manière dont les sorties sont générées. Ces descriptions dépendent du langage utilisé mais doivent être en tout les cas non ambigu.

Implémentation du modèle de simulation.

Un modèle de spécification MS est un modèle d'implémentation si :

- A (M, t) contient à tout temps t (*system time*) un ensemble de variables d'états.
- Θ décrit tout les changements de valeur de ces attributs.

Sinon, une fonction primaire du langage de programmation pour simulation (SPL) est d'augmenter les attributs du modèle A (M, t) jusqu'à constituer une réelle description des variables d'état. Le SPL doit également modifier Θ autant que nécessaire en rajoutant les transitions liées aux attributs additionnels.
Un modèle d'implémentation fournit donc la base désirée pour l'exécution du modèle, c'est à dire la mise en applications des actions, typiquement sur un ordinateur, décrit par le modèle d'implémentation.

Composants d'une spécification de conditions

Une CS peut être divisée en quatre composants distincts comme l'illustre le schéma suivant :

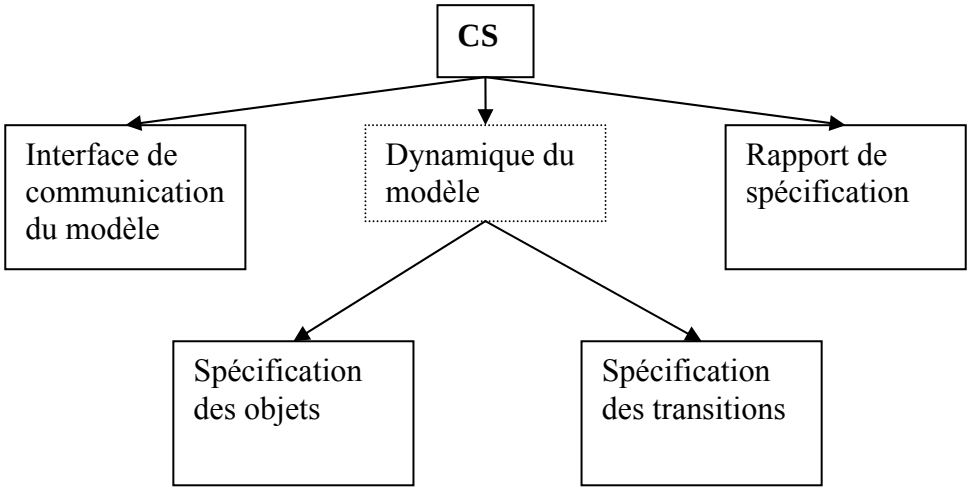


Figure 10 ***: Schéma compositionnel du CS.***

Spécification de l'interface

Celle-ci identifie tous les attributs par leurs noms, leur type de données et leur type de communication (entrée ou sortie). Cette interface peut-être dérivée de la dynamique interne du modèle et peut être générée par le système. Toute CS a au moins un attribut de sortie.

Spécification des objets

Il s'agit d'une liste contenant tous les éléments (objets) du modèle et leurs attributs correctement typés (booléen, entier, liste de valeurs discrètes…).
Il existe un type supplémentaire le *time-based signal* qui, en programmant des horaires de changement de valeur des attributs listés, permet de relier le temps de simulation à l'état du modèle.

Spécification des transitions

Cela consiste en un ensemble ordonné de paires *condition-action*.
Une *condition* est une expression booléenne composée d'attributs du modèle et des primitives CS : *WHEN ALARM* et *AFTER ALARM*.
Une *action* peut être divisée en cinq classes :

- description d'un changement de valeur.
- une action de séquençage du temps.
- génération ou destruction d'un objet.
- communication environnementale (en entrée ou en sortie).
- un état de terminaison pour la passe de simulation.

La spécification de transition peut être augmentée par des fonctions propres au modélisateur pour simplifier la représentation comportementale du modèle.
Les paires *condition-action* (CAPs) peuvent être réunies en *action clusters* (ACs) si les conditions entre paires sont équivalentes.
La CS fournit également bon nombre de primitives (autres que *WHEN ALARM* et *AFTER ALARM) :*

- *SET ALARM* et *CANCEL* manipulant la valeur des attributs typés *time-based signal.*
- *CREATE* et *DESTROY* manipulant l'instance d'objet « temporaire ».
- *INPUT* et *OUTPUT* qui fournissent un support de communication avec l'environnement.

Deux conditions apparaissent dans tout CS :

1. *initialisation* : celle-ci est vrai avant le premier changement de valeur du temps du système, faux sinon.
2. *terminaison* : cette expression est dépendante du modèle et peut être basée sur le temps, basée sur un état ou sur une combinaison des deux.

Rapport de spécification

La syntaxe de ce rapport est indéfinie, mais doit contenir un certain nombre d'indicateurs statistiques qui ne reflètent en eux mêmes pas le comportement du système mais peuvent être désirable pour l'analyse à posteriori. (Voir 4.)

La Thèse de E.Page [2] illustre les différentes étapes de cette méthodologie à travers l'exemple d'une file d'attente M/M/1.

Analyse du modèle dans la spécification de condition

La notion d'équivalence *en terme de spécification du modèle* est la clé de voûte de ce chapitre. Intuitivement, deux modèles sont équivalents si ils sont interchangeables, c'est-à-dire que pour des données en entrée identiques les deux modèles vont réagir de la même façon en produisant des données identiques en sortie. Pourtant on peut aussi considérer que deux modèles sont interchangeables si ils satisfont tout deux les objectifs d'étude bien que leurs comportements externes diffèrent.

Oversstreet identifie [31] deux types d'équivalence sur un ensemble d'attributs des modèles concernés:

- équivalence structurelle : les deux modèles ont des conditions équivalentes (1) et les actions associées (si les modèle sont stochastiques, les variables doivent suivre la même distribution) sont identiques (2).
- équivalence externe : les deux modèles spécifient des sorties identiques lorsqu'ils sont soumis à des entrées identiques.

Décomposition du modèle issu de CS

Le CS, bien qu'il soit conçu pour encourager un diagnostique automatique, doit pouvoir être analysé par un « humain » pour répondre à certaines questions plus spécifiques. Par conséquent pour un modèle donné il est primordial d'organiser les paires CAPs, en ACs comme dit dans la section *Spécification des transitions*, et de définir également une description du modèle décomposable en plusieurs niveaux.
Pour commencer on peut organiser les attributs de la manière suivante :

Attributs de contrôle .Il s'agit des attributs fournissant l'information nécessaire pour déterminer quand l'action doit intervenir, ils appartiennent à l'expression d'une condition.

Attributs d'entrée. Leur rôle est de fournir les données qui vont être utilisés soit pour changer la valeur des attributs de sortie soit pour programmer l'occurrence d'une future action.

Attributs de sortie. Ceux sont les attributs dont la valeur change à cause d'une action.

Tout les CAPs ont des attributs de contrôle, en revanche ils n'ont pas nécessairement d'attributs d'entrée ou de sortie.

Décomposition de l'objet. On souhaite réaliser des associations entre objet et CAPs par le biais des attributs qui les composent. Ainsi les CAPs associés à un objet décrit le comportement de celui-ci. Cette description basée objet peut se faire sous deux formes :

- toutes les actions du modèle affectent l'objet. → objet passif.
- toutes les actions du modèle sont effectuées par l'objet. → objet actif.

Overstreet adopte la seconde approche.

Une décomposition de l'objet doit être construite par identification, pour tout objet O, de deux types de CAPs : (1) les CAPs avec un attribut de contrôle de O et les (2) CAPs d'initialisation avec un attribut de sortie de O. Cette technique permet de séparer la spécification du modèle en une collection de petites unités facilement gérables, mais une partie de l'information sera redondante (car les mêmes CAPs peut apparaître dans la spécification de plusieurs objets) et la notion de séquençage dans les actions du modèle est difficile à détecter.

Décomposition en sous modèles. On cherche ici à identifier les sous modèles connectés de « façon minimale » dans le sens où l'on mesure les interactions entre groupe de ACs. Ce type de décomposition nécessite donc l'utilisation d'un *graphe d'interaction entre ACs* qui est un graphe orienté où les nœuds sont les ACs et les arcs les attributs les connectant. Cette représentation graphique a pour but de déterminer le nombre minimal d'arc reliant les sous modèles interactif.

Un fois le CS explicitement défini, il doit pouvoir être « traduit » en une représentation plus traditionnelle adoptant le concept de *localité* du monde réel. C'est pourquoi afin de produire une représentation orientée *échéance des événements,* les CAPs sont organisés autour des primitives *WHEN ALARM* (cf. *Spécification des transitions*), en générant à partir d'un *graphe d'incidence entre ACs* (ACIG), contenant un seul AC *time-based* et tout les ACs *state-based* construit sur cette base de temps, un ensemble de sous graphes décrivant ainsi un modèle exhibant la *localité de temps*. En parallèle, une *analyse de l'activité* peut être effectuée en créant des sous graphes orienté autour des ACs *state- based* dans l'ACIG (→ *localité d'état*), et une *interaction entre processus* peut être générer à partir de l'ACIG en créant des sous graphes qui soient des actions qui relient les objets entre eux comme dans le CS (→ *localité d'objet).*
C'est pourquoi nous allons étudier à présent différentes formes de représentations graphiques pour le diagnostique du modèle.

Graphe AC – attribut (ACAG)

Le ACAG représente les interactions entre ACs et attributs dans la CS, d'une part le potentiel d'action d'un AC de changer la valeur d'un attribut et d'autre part l'influence d'un attribut sur l'exécution d'un AC.
Soit une CS avec k time-based signal, m autres attributs et n ACs,
alors G est un graphe orienté avec k+m+n sommets(nœuds) et est construit comme ceci :
G a un arc orienté et labellisé du sommet i au sommet j si :

- le nœud i est un attribut de contrôle ou d'entrée pour le nœud j, un AC,
- le nœud j est un attribut de sortie pour le nœud i, un AC.

Un exemple de ACAG est donné dans la thèse de E.Page pour la CS d'un modèle M/M/1.
Le ACAG est un graphe biparti, par conséquent il nécessite deux matrices de booléen pour sa représentation :

- La matrice attribut→AC (AACM *attribute-action cluster matrix*).
- La matrice AC→attribut (ACAM *action cluster - attribute matrix*).

Pour une CS avec m ACs(AC_1, AC_2, …, AC_m), et n attributs(a_1, a_2, …, a_n) :

Le AACM est une matrice n par m $A_{n, m}$ telle que :

$a(i, j)=$ 1 si (a_i, AC_j) existe dans le ACAG; 0 sinon.

Le AACM est une matrice m par n $B_{m, n}$ telle que :

$b(i, j)=$ 1 si (AC_i, a_j) existe dans le ACAG; 0 sinon.

A partir de ces deux matrices, deux autres peuvent être formées, la matrice d'interaction attribut (AIM *attribute interaction matrix*),

AIM=AACM*ACAM

et la matrice d'interaction AC(ACIM *action cluster interaction matrix*)

ACIM=ACAM*AACM.

Graphe d'incidence inter ACs (ACIG)

Un ACIG est un graphe orienté où chaque sommet correspond à un AC du CS. Si l'action d'un AC, noté AC_i , provoque la réalisation de la condition d'un autre AC, noté AC_j , (c'est-à-dire que la condition passe à *TRUE* au même instant de simulation où est exécutée l'action AC_i ou par la mise en place d'une alarme) alors il existe un arc direct entre le nœud i et j.Par convention cet arc est en pointillé si AC_i pose une alarme utilisée par AC_j, solide sinon.

Il existe trois types de succession entre AC reliés par un arc :

1. successeur *time-based* si *WHEN ALARM.*
2. successeur *mixé* si *AFTER ALARM.*
3. successeur *stated-based* dans les autres cas.

Formellement, pour construire un ACIG à partir d'une CS composée d'un ensemble ACs (AC_1, AC_2, …, AC_n) il existe un algorithme décrit dans [2] qui décrit complètement les sphères d'influence de chacun des AC_i dans la CS.

En général, un grand nombre de ces interactions n'intervient jamais dans l'exécution du modèle car l'algorithme ne tient pas compte de « l'impact » (implique t'elle *p* ou ¬*p* si p est l'expression de la condition de l'AC récepteur?) de la post condition engendrée par l'AC initiateur.

Overstreet a montré qu'il n'existait pas d'algorithme pour simplifier complètement ce type de graphe, cependant Puthoff a décrit dans [194] un système expert capable de simplifier de façon presque optimale l'ACIG. Un exemple d'ACIG pour une file d'attente M/M/1 est donné dans [2].

3.1.4.3. Limite Théorique de l'analyse du modèle

Définitions.

- Deux séquences d'actions du modèle dans deux implémentations d'une spécification du modèle sont *équivalentes* si l'exécution de chacune d'elle à toute moment dans l'instanciation produit les mêmes résultats.
- Deux séquences d'actions, A et B, sont *indépendantes en ordre* si AB est équivalent à BA. (Commutatif).
- Une CS est *triviale* si la condition de terminaison est « rencontrée » en même temps que l'initialisation du modèle.

Propriétés.

- Une CS est *finie* si pour toute instanciation, pour des données valides en entrée, seulement un nombre fini d'action intervient avant que la condition de terminaison soit rencontrée.
- Une CS (dont toutes les actions sont sujettes à des influences stochastiques) est *ambiguë* si deux implémentations de celui-ci produisent des résultats différents en sortie bien que les entrées et le comportement stochastique soient identiques sur les deux implémentations.
- Une CS est *complète* si à tout moment de son instanciation, soit on rencontre la condition de terminaison soit il existe au moins une instance supplémentaire d'action « en attente ».
- Une CS est *accessible* si toute action qu'elle décrit peut intervenir dans une instanciation d'elle même.
- Une CS est *connectée* si le graphe ACIG complètement simplifié auquel on supprime le nœud d'initialisation est connecté.
- Une CS est *ambiguë en état* si l'indépendance en ordre n'est pas vérifiée pour deux AC contingent dont les conditions sont vraies simultanément pour toute instanciation de la CS.
- Une CS est *ambiguë en temps* si pour deux ACs déterminés dont l'occurrence peut être programmée au même instant dans une passe de simulation, ne sont ni indépendants en ordre ni suffisamment ordonnés (grâce à des informations de la CS) pour établir une priorité.

Résultats.

- Tout modèle de spécification fini est complet.
- Toute spécification de machine de Turing peut être transformée en CS.
- Il n'existe aucun algorithme pour déterminer si une CS est finie, complète, accessible, connecté ou ambiguë.
- Il n'existe pas d'algorithme pour déterminer l'indépendance en ordre de deux modèles actions.
- Il n'existe pas d'algorithme pour déterminer si une CS a la propriété d'être ambiguë en temps ou en état.
- Soit X une CS contenant seulement les ACs accessibles d'une CS Y.
- X est équivalent à Y avec respect pour tout les attributs de X.
- Pour toute CS finie, chaque instance d'action contingente est provoquée directement ou indirectement par la phase d'initialisation ou par une instance d'action coïncidant en temps avec l'instance d'action contingente.
- Une CS non triviale contient au moins un attribut *time-based signal*.

- Il n’existe pas d’algorithme pour déterminer si deux CS sont équivalentes de façon externe.
- Il n’existe pas d’algorithme pour savoir si deux conditions vont être simultanément vraies dans toute passe de simulation basée sur une CS.
- Il n’existe pas d’algorithme pour transformer une CS basée sur des CAPs en une CS basée sur des ACs avec un nombre minimum de ACs.
- Il n’existe pas d’algorithme pour générer un ACIG complètement simplifié.
- Il n’existe pas d’algorithme pour déterminer les successeurs effectifs d’un AC dans une CS.
- L’équivalence structurelle implique l’équivalence externe.

3.2. MODELISATION D'UN TRAFIC « IP »

3.2.1. Distribution de poisson et processus d'arrivés

3.2.1.1. Processus de poisson

La distribution de poisson donne la probabilité qu'il y ait exactement x occurrences indépendantes pendant une période donnée de temps (ou d'espace) si les événements prennent place indépendamment et à taux constant λ (λ>0).
La distribution de poisson est donc caractérisée par un paramètre λ qui correspond à l'occurrence la plus représentée en terme de probabilité, la moyenne λ.

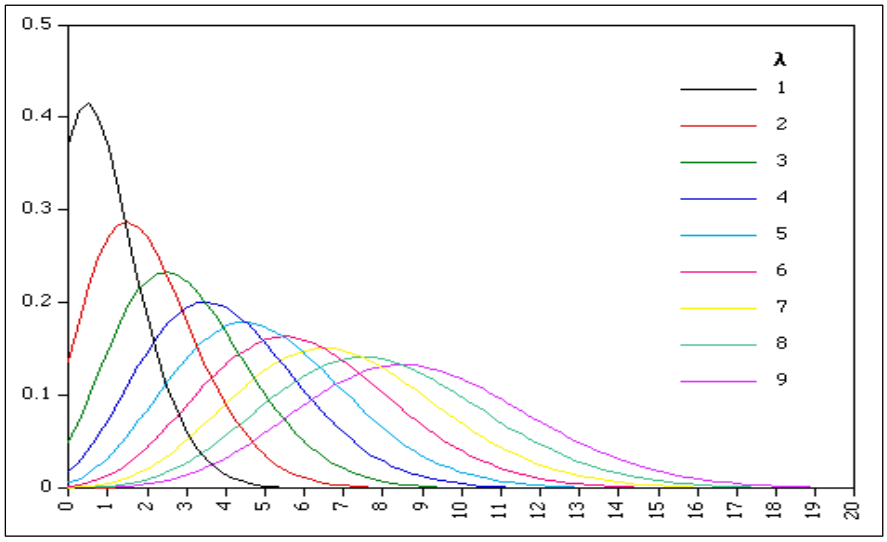


Figure 11*: Distribution de poisson selon la valeur de λ.*

Prenons l'exemple d'une branche d'arbre où se pose en moyenne λ oiseaux par heure, alors la probabilité de trouver exactement x oiseaux dans une heure est donnée par la formule,

$$f(x) = \frac{\lambda e^{x}}{x!}.$$

qui correspond à la distribution de poisson.

Dans une application comme celle de réseaux de communications IP la distribution de poisson est utilisée pour décrire l'arrivée des clients dans une file d'attente avec λ clients par seconde en moyenne (λ à priori relativement grand, par conséquent comme on peut le voir sur la Figure 11 la distribution de poisson a tendance à ressembler à une distribution normale N (λ, σ/√n), voir section 4.1.3).
En pratique si l'on souhaite générer un tel processus d'arrivée la démarche à suivre est la suivante :

1. tirer un chiffre x aléatoirement dans $U(0,1)$ (distribution uniforme sur l'intervalle [0,1]).
2. construire une suite u_n tel que $u_0=0$ et $u_n=u_{n-1} - \frac{\ln(x)}{\lambda}$.

3. la date de départ des paquets ou des flots (dans l'exemple QoSim) est alors déterminée par la suite u_n i.e le départ du n^{ieme} paquet est programmé à la date u_n (selon l'unité de temps choisie).

On peut alors remarquer que la variable aléatoire $\ln(x)/\lambda$ est bien celle désirée car exponentielle et de moyenne $1/\lambda$, et d'autre part le débit théorique pour une source qui émet un processus de poisson est bien λ pour n assez grand.

Une variante de cette distribution très populaire est la distribution de poisson non stationnaire, où le taux d'arrivé n'est plus constant mais correspond à une fonction $\lambda(t)$ avec $1/\lambda(t)$ la moyenne de la distribution exponentielle correspondant aux écarts inter clients au temps t.
Cette distribution permet de simuler des pics d'activité comme pour l'arrivée des clients dans une cafétéria.

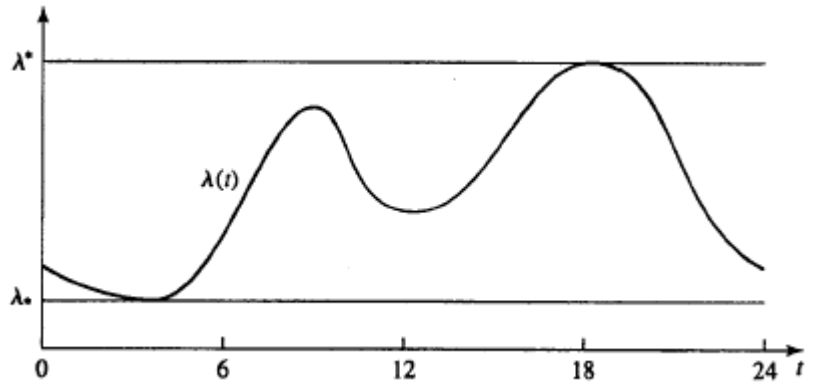


Figure 12 : ***Illustration d'un processus de poisson non stationnaire.***

La génération d'un processus de poisson non stationnaire est plus complexe mais consiste aussi à créer une suite u_n d'arrivées.

Etapes de construction [9]:

1. soit $u_{temp}=u_{n-1}$et $\lambda^* = \max_t(\lambda(t))$.
2. générer x1 et x2 indépendamment sur U (0,1).
3. remplacer u_{temp} par $u_{temp}-\ln(x1)/\lambda^*$.
4. si $x2 \leq \lambda(t)/\lambda^*$, alors $u_n=u_{temp}$ et on passe au suivant, sinon on retourne à l'étape 2.

Cependant la distribution de poisson semble échoué pour la représentation de trafic de fond sur des réseaux à grande échelle (voir [7]) car celui-ci semble trop régulier. C'est pourquoi nous allons étudier à présent d'autres processus d'arrivée. La Figure 13 met en avant la différence entre trafic « poissonien » et « self-similar » selon la période de temps sur laquelle on observe la quantité de paquets reçus. La définition d'un trafic self-similar et un exemple sont donnés dans la section modèle ON/OFF.

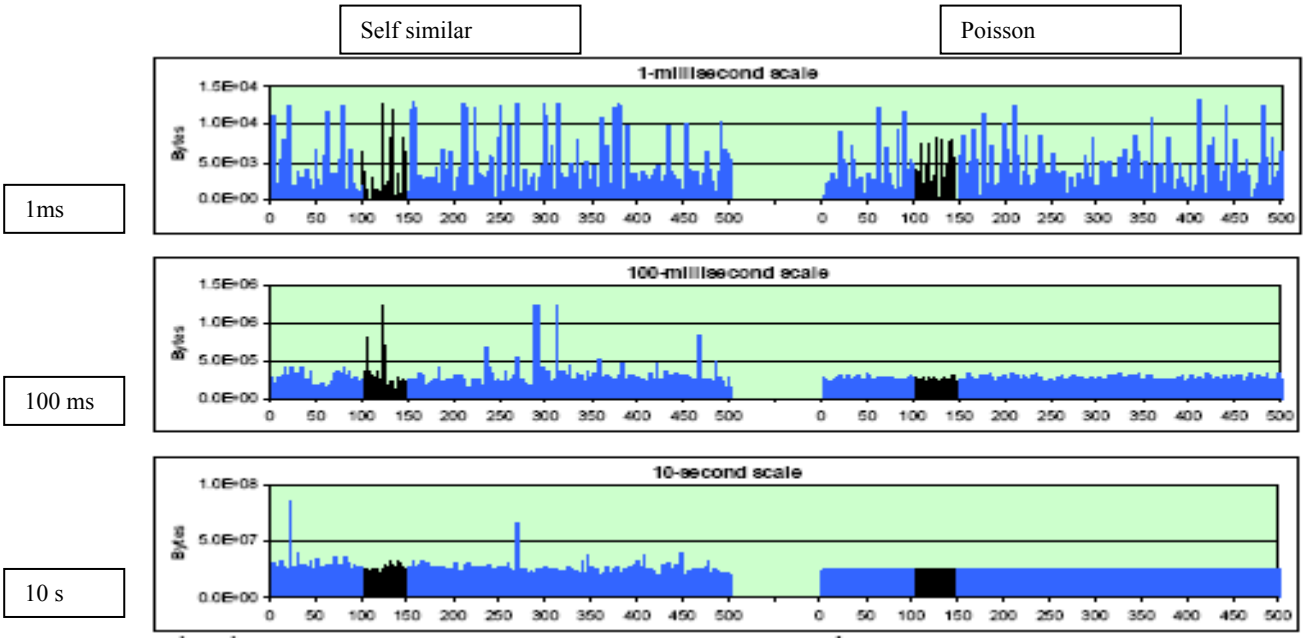


Figure 13 : ***Trafic « poissonien » face au trafic « self similar ».***

3.2.1.2. Distribution discrète

Ce processus d'arrivée peut se révéler intéressant dans des cas simples et si l'on connaît (par expérience à priori) les probabilités de l'occurrence de chaque événement.
Soit X l'événement nous concernant (et dans ce cas l'écart entre le départ ou l'arrivée de deux clients), et $P(X=X1)=a_1$, $P(X=X2)=a_2$,…, $P(X=Xn)= a_n$. $(a_1 + a_2+\ldots+ a_n=1)$.
Alors si l'on souhaite générer un tel processus il faut:

1. tirer une variable aléatoire x dans U (0, 1).
2. construire une suite b_i tel que b_i=somme des a_i de 1 à i.
3. si $b_i < x \leq b_j$ alors $u_n = u_{n-1}+Xj$.

3.2.1.3. Modèle ON/OFF ou processus interrompu

Il s'agit ici du modèle le plus intéressant pour la simulation de réseau de type IP, et il en existe plusieurs variantes.
Ce modèle est basé sur la superposition de périodes actives et passives (voire silencieuses pour les processus interrompus), c'est à dire que la source émet pendant un certain laps de temps t_1 des requêtes à un taux élevé puis à un taux plus faible (ou nulle) durant une autre période t_2 et ainsi de suite…
Une source interrompue se forme de la façon suivante: la source alterne entre des phases actives ("on") et silencieuses ("off") dont les durées t_i sont aléatoires.
Pendant les périodes "off", le processus d'arrivée est gelé et aucun client n'arrive. À la prochaine période "on", il reprend où il en était resté.

Par conséquent si durant les périodes "on" la génération de trafic est « poissonienne », en enlevant les périodes "off" et en recollant les périodes "on" on obtient à nouveau un processus de poisson.
On peut également imaginer un processus de poisson (ou autre) de paramètre λ entrecoupé de périodes "on" et "off" ayant des distributions exponentielle avec respectivement α et β comme paramètres caractéristiques (et donc 1/α et 1/β comme moyenne respective, avec α>β) pour obtenir un débit théorique

$$d = \frac{\lambda\beta}{\alpha + \beta}.$$

Cependant le modèle "ON/OFF" le plus courant se base sur des distributions de Pareto ayant deux paramètres caractéristiques.
Le premier, a (1<a<2 pour un trafic self similar), paramètre la courbure de la distribution et le second, b, caractérise la « position de départ » de la courbe.

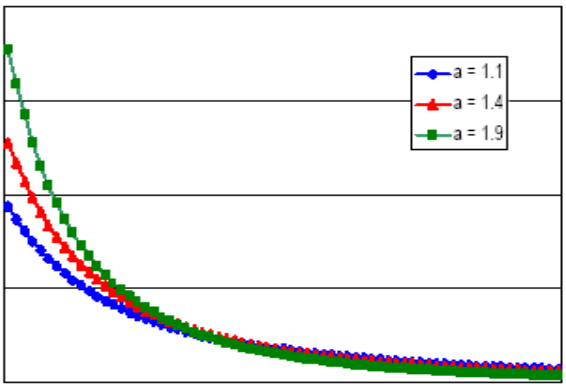


Figure 14 ***: courbe de Pareto selon le paramètre a.***

La distribution de Pareto possède la fonction probabiliste de densité suivante :

$$f(x) = \frac{ab^a}{x^{a+1}}$$

pour $x \geq b$,
et sa moyenne noté E(x) est :

$$E(x) = \frac{ab}{a-1}.$$

Pour générer une distribution de Pareto il faut :

1. tirer un chiffre u aléatoirement dans $U\ (0,\ 1)$.
2. générer un ensemble de variables $X_{PARETO} = b/u^{1/a}$.

A noter que l'utilisation de pseudo générateur dans une simulation sur ordinateur nous conduit à des résultats légèrement différents. En effet si l'on note m la plus petite valeur que peut produire le pseudo générateur utilisé (ainsi $M = b/m^{1/a}$ sera la plus grande valeur que peut prendre X_{PARETO}) alors la moyenne E(x) vaut :

$$E(x) = \frac{ab}{a-1} \times \frac{1 - m^{\frac{a-1}{a}}}{1-m},$$

car l'intégrale des probabilités est tronquée et ne vaut pas 1.

De nombreuses études ([7] et [8]) s'accordent à dire que le trafic Internet est « self-similar » (traduisible en français par auto similaire ou similaire à soi même) et ne suit pas un processus de poisson.

Nous allons donc brièvement définir la nature d'un tel trafic puis étudier sur un exemple sa génération et ses propriétés.

Il faut au préalable définir la notion de processus cumulatif Y (t) et de processus incrémental X_t.

Dans l'exemple QoSim, Y (t) sera par exemple le nombre total de paquets reçus sur un certain nœud jusqu'au temps t alors que $X_t = Y(t+1) - Y(t)$ sera le nombre de paquets reçus au temps « instantané » t.

Il existe plusieurs définitions- pas forcément équivalentes et souvent trop restrictives- de la notion de self similarité, celle qui nous intéresse, qu'on dit de *second ordre pour la variable incrémental*, est la suivante :

Pour tout entier m,

$$X^{(m)} = m^{H-1} X_o.$$

Avec H, paramètre de *Hurst*, compris entre 0 et 1, et $X^{(m)}$ la moyenne des X_t sur m mesures.

On dit également que le processus est asymptotiquement self similar (de second ordre) si :

$$\lim m \to \infty \; X^{(m)} = m^{H-1} X_o.$$

En étudiant la fonction d'auto corrélation (voir section 4.1.1) sur de tels processus on peut observer que ceux-ci sont à la fois self similar et dépendants sur de grands intervalles pour ½<H<1. (NB : Self-similar ne veut pas forcément dire qu'il existe une dépendance entre intervalles larges).

Ainsi pour générer un processus à la fois self-similar et dépendant sur de grands intervalles, on peut utiliser le modèle ON/OFF basé sur des distributions de Pareto et procéder comme suit pour une source i (Voir [33]):

Pour la période ON, on choisit a_{ON}=1.4, (comme H= (3-a)/2 → H=0.8) et b_{ON}=1, la taille minimum d'un paquet.

Pour la période OFF, on pose a_{OFF}=1.2, et on calcule b_{OFF} comme suit :

Ce calcul correspond à la charge par source qu'on désire obtenir, si l'on note L_i la charge générée par le nœud i alors on a $L_i = E(ON_i)/(E(ON_i)+E(OFF_i))$ et ainsi :

$$b_{OFF} = b_{ON} \times \frac{a_{ON}}{a_{OFF}} \times \frac{a_{OFF} - 1}{a_{ON} - 1} \times (1/L_i + 1)$$

selon les formules de moyenne établies précédemment (en considérant que toutes les sources sont définies de la même manière).

Soit toute les sources génèrent la même charge, possèdent les mêmes caractéristiques et alors $L_i = L_j$ pour tout i et j et ainsi ce calcul est le même pour toutes les sources, soit on peut considérer que chaque source émet une charge différente et alors il faut calculer b_{OFF} selon les trois autres valeurs caractéristiques (a_{ON}, b_{ON}, et a_{OFF}) de la source en question.

Lorsqu'on utilise un pseudo générateur b_{OFF} doit être recalculé selon les nouvelles moyennes E(x). (Comme vu précédemment).

3.2.2. Application de la méthodologie

Etant donné la souplesse de la CM/CS décrite dans 3.1.4 et la complexité d'un modèle comme celui sur lequel l'exemple dans la section 5 est basé, il est impossible de décrire un modèle unique et parfait pour des réseaux de type IP, cependant des éléments de réponse et des exemples sont donnés dans le chapitre 5 section 1 (Dans sa thèse E.Page [2] détaille également un certain nombre d'exemples de manière exhaustive sur des thèmes proches d'un réseau de communications). De plus, une issue fondamentale de la modélisation d'un réseau est la génération de trafic. Les principaux modèles en sont donnés dans la section précédente ainsi que dans le chapitre 5.2.2. Or les hypothèses de simulation sont validées par l'aptitude du modèle à reproduire la réalité du système étudié, et dans le cas d'un modèle de trafic, il semble que le modèle ON/OFF basé sur des distributions de Pareto soit le plus réaliste actuellement reconnu (grâce à sa propriété de dépendance sur de longs espaces de mesures selon le paramètre de Hurst, Figure 13).

4. INTERPRETATION DES RESULTATS D'UNE SIMULATION

4.1. ANALYSE STATISTIQUE DES DONNEES EN SORTIE

4.1.1. Estimateurs classiques et termes génériques

Le plus souvent dans une simulation, les paramètres mesurés en sortie correspondent à des moyennes de variables d'état si celle-ci sont d'ordre numérique. Par exemple dans le cadre d'une simulation stochastique comme décrit en 5., il peut s'agir du nombre moyen de flots vivants, du délai moyen d'acheminement d'un paquet ou encore du nombre moyen de clients dans une file d'attente.
C'est pourquoi nous allons nous intéresser tout particulièrement aux calculs de moyenne, variance et autres outils statistiques qui vont être décrits dans cette partie.
Cette partie a été rédigée en s'appuyant, de manière générale, sur les documents [6] et [9] pour les fondements statistiques.

Notations utilisées :

- μ : moyenne théorique.
- σ : écart type théorique, σ^2 : variance théorique.
- Soit $\{X_1, X_2, \ldots, X_n\}$ l'échantillon de valeurs récoltées sur un paramètre X au cours de la simulation.
  - $\overline{X(n)}$: moyenne mesurée sur n observations.
  - $S(n)$: écart type à la moyenne mesurée, $S^2(n)$: variance mesurée sur n observations.

- $U(a, b)$: distribution uniforme sur l'intervalle [a, b].
- $N(\mu, \sigma/\sqrt{n})$: distribution normale de moyenne μ et de variance σ^2 (sur n mesures).
- IID : indépendants et identiquement distribués.
- $|I/2|$: désigne la demi norme d'un intervalle I= [a, b] ($|I/2| = (b-a)/2$).
- $\lfloor x \rfloor$: partie entière du réel x.

Estimateurs sans biais :

Si la famille d'éléments étudiés est composée de membres indépendants et identiquement distribués (IID) un estimateur sans biais pour la moyenne est :

$$\overline{X(n)} = \sum_{i=1}^{n} \frac{1}{n} X_i$$

, et pour la variance :

$$S^2(n) = \sum_{i=1}^{n} \frac{\left(X_i - \overline{X(n)}\right)^2}{n-1}.$$

Si ce n'est pas le cas, les méthodes usuelles de statistique ne peuvent être appliquées directement car ces estimateurs sont approximatifs et donc avec biais, c'est-à-dire qu'il surestime la variance.

Nous allons donc étudier dans la suite de ce chapitre certaines méthodes particulières qui s'appliquent sur les données collectées en sortie d'un simulateur stochastique et qui ne respectent généralement pas ces conditions IID.

D'autres indicateurs comme la covariance ou la corrélation entre intervalles peuvent se révéler intéressante pour mesurer la dépendance entres échantillons.

Soit X et Y deux intervalles distincts de même cardinal prenant les valeurs $\{X_1, X_2, \ldots, X_n\}$ et $\{Y_1, Y_2, \ldots, Y_n\}$ et si l'on note $\mathrm{cov}(X, Y)$ la covariance entre ces deux échantillons alors,

$$\mathrm{cov}(X,Y) = \frac{\sum_{i=1}^{n}(X_i - \overline{X(n)})(Y_i - \overline{Y(n)})}{n}.$$

On a également les propriétés suivantes :

- $\mathrm{cov}(X, Y)=\mathrm{cov}(Y, X)$
- $\mathrm{cov}(X, X) \approx S_x^2(n)$
- si X et Y sont indépendants alors $\mathrm{cov}(X, Y)=0$, la réciproque étant fausse.

Si l'on note $\mathrm{corr}(X, Y)$ le coefficient de corrélation entre X et Y alors celui-ci se définit par,

$$\mathrm{corr}(X,Y) = \frac{\mathrm{cov}(X,Y)}{(S_x(n))(S_Y))} \in [-1,1]$$

.

Cette valeur est sans unité et désigne la corrélation linéaire entre X et Y, si bien que si $\mathrm{corr}(X, Y)$ est suffisamment proche de 0 on peut en déduire que X et Y n'ont pas de relation linéaire mais pas de notion d'indépendance plus générale. On parle de corrélation positive (les éléments étudiés sont positivement et linéairement dépendant, par exemple X=7Y) si cette valeur est comprise entre O et 1, négative sinon.

4.1.2. Le traitement statistique dépend du type de simulation

Selon le type de simulation, les données en sortie peuvent être assujetties à des manipulations statistiques afin de valider les résultats de la simulation. En effet lorsque les événements décrits en entrée ne sont pas entièrement déterminés à l'avance, c'est-à-dire lorsqu'on utilise des générateurs de nombres aléatoires (distribution uniforme U) ou des distributions classiques (issues de générateurs aléatoires PRNGs) comme celles présentées dans la section 3.2.1, la pertinence et la précision des résultats est mise en doute par le caractère aléatoire des données en entrée d'autant plus que sur ordinateur on utilise des générateurs pseudo aléatoires (voir section 2.1). La Figure 15 illustre le type de simulation pour lesquelles les mesures en sortie sont sujettes aux outils qui vont être décrits dans la suite de ce chapitre.

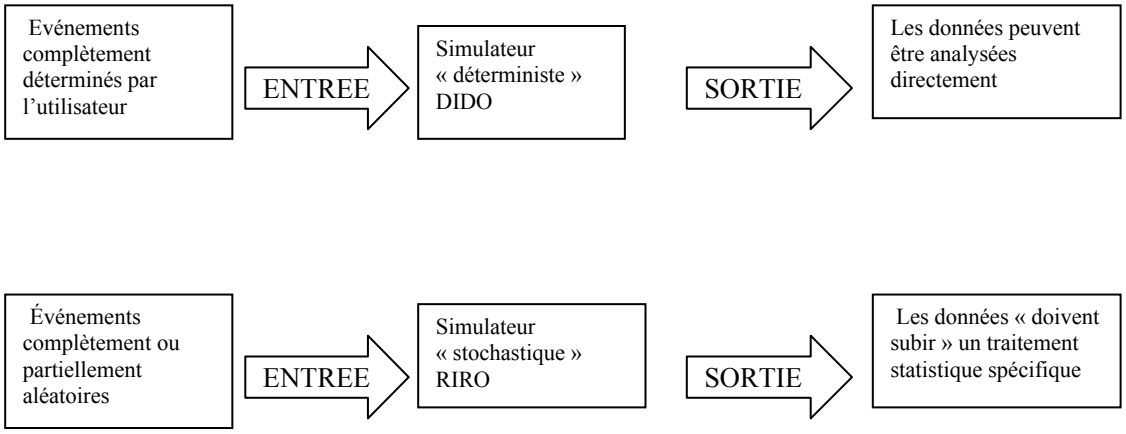


Figure 15 : ***Deux types de simulation.***

DIDO :Data In Data out
RIRO :Random In Random out

Les données ou résultats que l'on souhaite mesurer et analyser en sortie correspondent généralement aux moyennes des divers paramètres évoluant au cours de la simulation – mais de nombreuses autres mesures peuvent se révéler intéressantes selon le modèle étudié. Pour évaluer la confiance qu'on peut accorder à ces moyennes mesurées il faut introduire la notion de niveau et d'intervalle de confiance. Ces deux notions sont directement liées à la loi centrale centrée réduite $N(0,1)$ que nous allons étudier dans la section suivante.

4.1.3. Loi normale, intervalle de confiance et niveau de confiance

Soit une population de taille n dont les observations $\{X_1, X_2, \ldots, X_n\}$, sur un paramètre X de moyenne μ et de variance σ^2, sont indépendantes et distribuées normalement, alors la variable,

$$\overline{X(n)} = \sum_{i=1}^{n} \frac{1}{n} X_i$$

suit une loi normale $N(\mu, \sigma\sqrt{n})$.

Si l'hypothèse de normalité n'est pas satisfaite mais que la population est composée d'observations IID et si la taille de la population est supérieure à 30 alors, selon le *théorème centrale limite (TCL)* [6] [1], la variable

$$Y = \frac{(\overline{X(n)} - \mu)}{\sigma / \sqrt{n}}$$

est distribuée selon une normale $N(0,1)$. (cf. Figure 16)

[1] Il est à noter que la TCL doit vérifier un certain nombre d'hypothèses basées sur la nature stochastique du processus étudié. Alexopoulos dans [10] définit les deux hypothèses (ASA et AWA), qu'on jugera assez généralement satisfaites, nécessaires à l'application du TLC, elles sont construites sur la base de processus standard de Brownian [10].

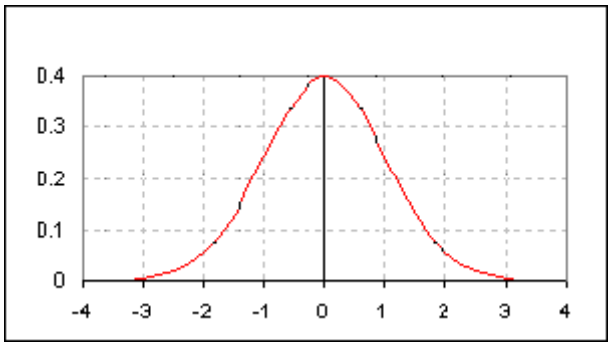


Figure 16 : ***Distribution N (0,1) ou loi centré réduite.***

Il s'agit d'une distribution gaussienne dont la densité est donné par $f(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2}$

qui va nous permettre d'évaluer, dans la mesure où l'on connaît sa fonction de répartition, la confiance que l'on peut accorder aux mesures de moyenne grâce au théorème centrale limite.

On peut à présent introduire la notion d'intervalle de confiance associé à un certain niveau de confiance, et la notion d'erreur relative qui en découle.

Définitions :

- Le niveau de confiance 1-α décrit la probabilité P, ($0 \leq P \leq 1$), que la vraie valeur d'un paramètre γ soit située entre les bornes a et b de l'intervalle de confiance I=[a,b].
  Intervalle de confiance [a, b] et niveau de confiance α alors :
  $P\{a \leq \text{vrai valeur de } \gamma \leq b\} = 1- \alpha.$

ou $P\{\text{vrai valeur de } \gamma \in I\} = 1- \alpha$.

On doit interpréter de la manière suivante :
Il a (1- α)*100 % de chance que l'intervalle I couvre la réelle valeur du paramètre γ ou bien il y a un risque de 100* α que l'intervalle I ne couvre pas la vraie valeur de γ.

Et grâce au théorème centrale limite (*TCL*), on a pour la moyenne théorique μ,

$$P\left\{\overline{X(n)} - z_{\alpha/2} \times \sigma\sqrt{n} \leq \mu \leq \overline{X(n)} + z_{\alpha/2} \times \sigma\sqrt{n}\right\} = \frac{1}{2\pi} \int_{-z_{\alpha/2}}^{z_{\alpha/2}} e^{-\frac{1}{2}x^2} dx = 1 - \alpha$$

à condition que la famille d'éléments étudiés $\{X_n\}$ soit IID et composé de plus de 30 individus, et on peut donc déterminer les bornes de l'intervalle de confiance selon le calcul de cette intégrale, cependant on préférera utiliser les tables récapitulatives de la loi normale ou de la loi de student donnée dans le Tableau 3 et 4.

- L'erreur relative décrit le rapport entre la demi largeur de l'intervalle de confiance et la moyenne mesurée.
  Erreur relative E (n) :

$$E(n) = \frac{|b - a|/2}{\overline{X(n)}}$$

car les intervalles de confiance sont symétriques (comme $N(0, 1)$) autour de la moyenne.

- Le t-ratio se définit comme le rapport entre la différence de la moyenne mesurée à la moyenne théorique et l'écart type divisé par le nombre d'individus de la population étudiée, il s'agit de la variable Y définie ci-dessus et on a:

$$t_{ratio} = \frac{|\mu - \overline{X(n)}|}{\sigma/\sqrt{n}} \xrightarrow[n\to\infty]{} N(0,1)$$

Le t-ratio peut aussi, dans le cas où n est petit (n<30), être distribué selon la distribution t de student avec n-1 degré de liberté :

$$t_{ratio} \approx t_{n-1}$$

L'intervalle de confiance I pour la moyenne pour une population supérieure à 30 individus non distribué normalement (mais IID) est alors, pour un risque α,

$$I = \left[\overline{X(n)} - z_{\alpha/2} \times \sigma/\sqrt{n}, \overline{X(n)} + z_{\alpha/2} \times \sigma/\sqrt{n}\right]$$

Z est une valeur à déterminer en fonction de la table 20, qui se contente de récapituler la densité de répartition pour des valeurs classiques de niveau de confiance α de la distribution normale N (0, 1) (en général on prend 0.05 ou 0.1 pour un niveau de confiance élevé).

α	0,00	0,01	0,02	0,03	0,04	0,05	0,06	0,07	0,08	0,09
0,00	-	2,576	2,326	2,170	2,054	1,960	1,881	1,812	1,751	1,695
0,10	1,645	1,598	1,555	1,514	1,476	1,440	1,405	1,372	1,341	1,311
0,20	1,282	1,254	1,227	1,200	1,175	1,150	1,126	1,103	1,080	1,058
0,30	1,036	1,015	0,994	0,974	0,954	0,935	0,915	0,896	0,878	0,860
0,40	0,842	0,824	0,806	0,789	0,772	0,755	0,739	0,722	0,706	0,690
0,50	0,674	0,659	0,643	0,628	0,613	0,598	0,583	0,568	0,553	0,539
0,60	0,524	0,510	0,496	0,482	0,468	0,454	0,440	0,426	0,412	0,399
0,70	0,385	0,372	0,358	0,345	0,332	0,319	0,305	0,292	0,279	0,266
0,80	0,253	0,240	0,228	0,215	0,202	0,189	0,176	0,164	0,151	0,138
0,90	0,126	0,113	0,100	0,088	0,075	0,063	0,050	0,038	0,025	0,013

..... : pour 95 % (exemple a)
——— : pour 77% (exemple b)

Tableau 3 *:* Table de la loi centrée réduite.

Exemple :

Il faut lire cette table la manière suivante :

a. *Z*=1.96 pour α=0.05= 0 (en ligne) + 0.05 (en colonne). ->95 % de confiance.

b. Z=1.200 pour α=0.23= 0.20 (en ligne) + 0.03 (en colonne). ->77 % de confiance.

Dans l'exemple décrit plus haut pour un niveau de confiance de 95% on a un intervalle de confiance :

$$I = [\overline{X(n)} - 1.96 \times \sigma / \sqrt{n}, \overline{X(n)} + 1.96 \times \sigma / \sqrt{n}] \ .$$

Cependant en général, la taille de l'échantillon (ou de la population dans l'exemple) peut être inférieure à 30 et d'autre part la variance est habituellement inconnue.

Il est donc nécessaire d'étudier la distribution de student qui s'adapte aux petits échantillons et d'utiliser un estimateur de variance le moins biaisé possible.

La distribution de student est comparable à la distribution normale plus ou moins « amortie » selon la taille de l'échantillon (cf. Tableau 4). Cette loi convient tout particulièrement aux expériences où le nombre d'observations est faible comme en médecine, mais aussi dans la simulation à événements discrets dans la mesure où dans une même passe de simulation les événements sont souvent corrélés (donc pas indépendants) et par conséquent il va falloir répéter plusieurs fois la simulation pour obtenir des passes de simulation indépendantes (cela peut se révéler très coûteux et donc le nombre de réplications sera généralement inférieur à 30, sinon on peut utiliser la loi normale avec le même estimateur de variance). Ainsi, il va falloir utiliser cette loi de student propice aux échantillons restreints.

En ce qui concerne l'estimation de la variance on utilisera l'estimateur classique,

$$S^2(n) = \sum_{i=1}^{n} \frac{\left(X_i - \overline{X(n)}\right)^2}{n-1}$$

et sous réserve que les conditions IID soit respectées, la quantité

$$Y = \frac{(\overline{X(n)} - \mu)}{S(n)/\sqrt{n}}$$

est distribuée selon la loi t de Student avec n-1 degré de liberté.

Ainsi les limites de l'intervalle de confiance [a, b] seront :

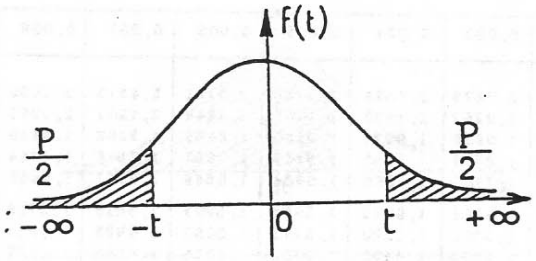
$$a = \overline{X(n)} - t_{\alpha/2,n-1} \times S(n)/\sqrt{n} \text{ et } b = \overline{X(n)} + t_{\alpha/2,n-1} \times S(n)/\sqrt{n} \ .$$

Exemple :

Pour n=10 et un niveau de confiance de 95%, $a = \overline{X(n)} - 2.262 \times S(n)/\sqrt{n}$ et $b = \overline{X(n)} + 2.262 \times S(n)/\sqrt{n}$ selon la table 28. (Respectivement : « n → v » ; « P : 95% → 0.05% (=1-0.95) », l'exemple est illustré en jaune pale sur la table de student à la page suivante).

TABLE DE DISTRIBUTION DE T (LOI DE STUDENT)

Valeurs de T ayant la probabilité P d'être dépassées en valeur absolue



ν \ P	0,90	0,80	0,70	0,60	0,50	0,40	0,30	0,20	0,10	0,05	0,02	0,01	0,001
1	0,158	0,325	0,510	0,727	1,000	1,376	1,963	3,078	6,314	12,706	31,821	63,657	636,619
2	0,142	0,289	0,445	0,617	0,816	1,061	1,386	1,886	2,920	4,303	6,965	9,925	31,598
3	0,137	0,277	0,424	0,584	0,765	0,978	1,250	1,638	2,353	3,182	4,541	5,841	12,929
4	0,134	0,271	0,414	0,569	0,741	0,941	1,190	1,533	2,132	2,776	3,747	4,604	8,610
5	0,132	0,267	0,408	0,559	0,727	0,920	1,156	1,476	2,015	2,571	3,365	4,032	6,869
6	0,131	0,265	0,404	0,553	0,718	0,906	1,134	1,440	1,943	2,447	3,143	3,707	5,959
7	0,130	0,263	0,402	0,549	0,711	0,896	1,119	1,415	1,895	2,365	2,998	3,499	5,408
8	0,130	0,262	0,399	0,546	0,706	0,889	1,108	1,397	1,860	2,306	2,896	3,355	5,041
9	0,129	0,261	0,398	0,543	0,703	0,883	1,100	1,383	1,833	2,262	2,821	3,250	4,781
10	0,129	0,260	0,397	0,542	0,700	0,879	1,093	1,372	1,812	2,228	2,764	3,169	4,587
11	0,129	0,260	0,396	0,540	0,697	0,876	1,088	1,363	1,796	2,201	2,718	3,106	4,437
12	0,128	0,259	0,395	0,539	0,695	0,873	1,083	1,356	1,782	2,179	2,681	3,055	4,318
13	0,128	0,259	0,394	0,538	0,694	0,870	1,079	1,350	1,771	2,160	2,650	3,012	4,221
14	0,128	0,258	0,393	0,537	0,692	0,868	1,076	1,345	1,761	2,145	2,624	2,977	4,140
15	0,128	0,258	0,393	0,536	0,691	0,866	1,074	1,341	1,753	2,131	2,602	2,947	4,073
16	0,128	0,258	0,392	0,535	0,690	0,865	1,071	1,337	1,746	2,120	2,583	2,921	4,015
17	0,128	0,257	0,392	0,534	0,689	0,863	1,069	1,333	1,740	2,110	2,567	2,898	3,965
18	0,127	0,257	0,392	0,534	0,688	0,862	1,067	1,330	1,734	2,101	2,552	2,878	3,922
19	0,127	0,257	0,391	0,533	0,688	0,861	1,066	1,328	1,729	2,093	2,539	2,861	3,883
20	0,127	0,257	0,391	0,533	0,687	0,860	1,064	1,325	1,725	2,086	2,528	2,845	3,850
21	0,127	0,257	0,391	0,532	0,686	0,859	1,063	1,323	1,721	2,080	2,518	2,831	3,819
22	0,127	0,256	0,390	0,532	0,686	0,858	1,061	1,321	1,717	2,074	2,508	2,819	3,792
23	0,127	0,256	0,390	0,532	0,685	0,858	1,060	1,319	1,714	2,069	2,500	2,807	3,767
24	0,127	0,256	0,390	0,531	0,685	0,857	1,059	1,318	1,711	2,064	2,492	2,797	3,745
25	0,127	0,256	0,390	0,531	0,684	0,856	1,058	1,316	1,708	2,060	2,485	2,787	3,725
26	0,127	0,256	0,390	0,531	0,684	0,856	1,058	1,315	1,706	2,056	2,479	2,779	3,707
27	0,127	0,256	0,389	0,531	0,684	0,855	1,057	1,314	1,703	2,052	2,473	2,771	3,690
28	0,127	0,256	0,389	0,530	0,683	0,855	1,056	1,313	1,701	2,048	2,467	2,763	3,674
29	0,127	0,256	0,389	0,530	0,683	0,854	1,055	1,311	1,699	2,045	2,462	2,756	3,659
30	0,127	0,256	0,389	0,530	0,683	0,854	1,055	1,310	1,697	2,042	2,457	2,750	3,646
40	0,126	0,255	0,388	0,529	0,681	0,851	1,050	1,303	1,684	2,021	2,423	2,704	3,551
80	0,126	0,254	0,387	0,527	0,679	0,848	1,046	1,296	1,671	2,000	2,390	2,660	3,460
120	0,126	0,254	0,386	0,526	0,677	0,845	1,041	1,289	1,658	1,980	2,358	2,617	3,373
∞	0,126	0,253	0,385	0,524	0,674	0,842	1,036	1,282	1,645	1,960	2,326	2,576	3,291

Tableau 4 *:* Table de Student.

On remarque que la table de student s'applique également aux échantillons dont le cardinal est compris entre 30 et 120, les valeurs de la ligne ∞ correspondent à celles données dans le tableau 3. On admettra que pour un échantillon de taille supérieure à 30 on peut utiliser la distribution normale comme approximation. (En bleu ciel, la valeur de t pour α=0.05 donnée dans l'exemple sur la loi normale).

4.2. ANALYSE APPROPRIEE

On distingue deux types de simulation pour lesquels les opérations statistiques à mettre en œuvre divergent :

1. Les systèmes se terminant, aussi appelés à horizon fini :
   Dans ce cas, les données en sortie ne se doivent pas d'atteindre un état d'équilibre, on considère alors que toutes les données récoltées sont dans un état transitoire dans la mesure où leurs valeurs dépendent des conditions initiales.

2. Les systèmes ne se terminant pas ou dont la durée est très longue, communément appelés systèmes à état d'équilibre :
   En ce qui concerne ce type de système, dit *steady-state,* il s'agit d'étudier le comportement du système pendant une longue période, on dit alors que le système est en état d'équilibre si les mesures en sortie suivent une distribution équilibrée caractéristique d'un processus stochastique. Dans ce cas il faut distinguer les données récoltées en état transitoire de celles recueillies pendant l'état d'équilibre.

Pour les systèmes à horizon fini, les techniques à mettre en œuvre sont relativement simples alors que dans les systèmes à état d'équilibre la procédure à suivre est nettement plus complexe : d'une part à cause de la durée de la simulation qui rend les réplications coûteuses en terme de temps et d'autre part à cause des conditions initiales qui faussent les résultats.
Nous allons donc aborder dans un premier temps les méthodes statistiques pour le calcul d'intervalle de confiance s'appliquant aux systèmes se terminant puis celles concernant les systèmes ne se terminant pas, ensuite nous étudierons le volume de données à récolter nécessaire pour obtenir des résultats satisfaisants certaines exigences de qualité.

4.2.1. Systèmes à horizon fini

Supposons que l'on simule un système jusqu'à obtenir m données en sortie $X_1, X_2, \ldots, X_m$ avec l'objectif de déterminer la moyenne théorique μ des $\{X_i\}$.
Il est clair que l'estimateur classique $\overline{X(m)}$ (voir section 4.1.1) est non biaisé pour μ.
Malheureusement les X_i sont généralement des variables dépendantes faisant de $S^2(m)$ (voir section 4.1.1) un estimateur biaisé pour la variance.
Pour contourner ce problème on réalise k passes indépendantes de simulation sur m observations (d'où $k \times m$ observations au total) chacune avec des générateurs de nombres aléatoires différents (ou avec des graines différentes) pour assurer une réelle indépendance entre les mesures. A la $i^{ème}$ passe on collecte donc les observations $X_{i1}, X_{i2}, \ldots, X_{im}$ et on note $\overline{Y_i(m)}$ la moyenne interne d'une passe :

$$\overline{Y_i(m)} = \frac{1}{m}\sum_{j=1}^{m} X_{ij}$$

Les Y_i sont des variables IID, et leur moyenne,

$$\overline{Y(k \times m)} = \frac{1}{k}\sum_{i=1}^{k} \overline{Y_i(m)}$$

est un estimateur non biaisé pour la moyenne μ tout comme leur simple variance,

$$S_m^2(k \times m) = \frac{1}{k-1} \sum_{i=1}^{k} (\overline{Y_i(m)} - \overline{Y(m \times k)})^2$$

, est un estimateur non biaisé de la variance par rapport à μ.
On peut à présent, grâce au théorème centrale limite (voir section loi), utiliser la table de student pour déterminer l'intervalle I de confiance pour un niveau de confiance α donné.
On en déduit alors que l'intervalle :

$$I = \left[\overline{Y(m \times k)} - t_{\alpha/2,k-1} \times S_m^2(k \times m), \overline{Y(m \times k)} + t_{\alpha/2,k-1} \times S_m^2(k \times m) \right]$$

a (1- α)*100 % de chance de couvrir la moyenne théorique μ.

4.2.2. Systèmes à état d'équilibre

4.2.2.1. Période stationnaire et période « warm up »

Pour les systèmes ne se terminant pas il va falloir délimiter la zone à état d'équilibre de la période pendant laquelle le système s'initialise : la période « warm up » de taille l, dont les mesures faussent le calcul de la moyenne théorique.

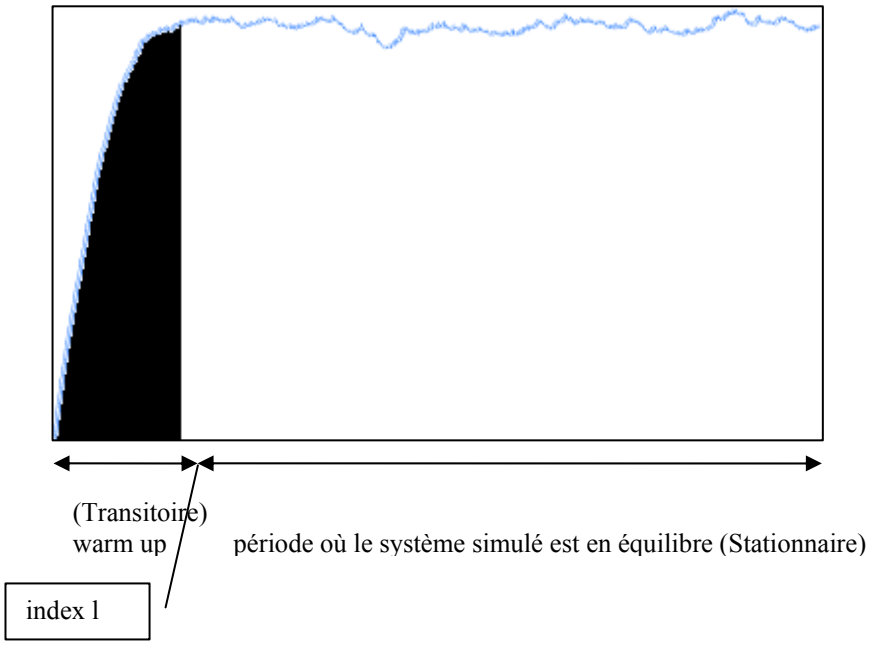


Figure 17*: Etat transitoire / état stationnaire.*

Il va donc s'agir d'éliminer dans le calcul de la moyenne les valeurs récoltées durant une certaine période l d'initialisation pour que l'estimation de la moyenne soit la moins biaisée possible. De nombreuses procédures ont été proposées pour détecter cet index l (voir

Fishman [36] ou Ockerman [37]). La plus populaire est, en partie grâce à sa simplicité et sa généralité, la procédure graphique de Welch [38].

Méthode de Welch[1]

Cette technique repose sur k réplications indépendantes de la simulation produisant à la $i^{ème}$ passe les observations $X_{i1}, X_{i2}, \ldots, X_{in}$.
On calcule alors les moyennes inter passes,

$$\overline{X_j} = \frac{\sum_{i=1}^{k} X_{ij}}{k}, \qquad j = 1,\ldots,n.$$

Ainsi pour une fenêtre de temps donnée w, la technique consiste à délimiter les moyennes mouvantes,

$$\overline{X_j(w)} = \left\{ \begin{array}{ll} \frac{1}{2w+1} \sum_{m=-w}^{w} \overline{X_{j+m}} & w+1 \le j \le n-v \\ \frac{1}{2j-1} \sum_{m=-j+1}^{j-1} \overline{X_{j+m}} & 1 \le j \le w \end{array} \right\}$$

par rapport à j.
L'index l est alors choisi comme étant la valeur de j pour laquelle la séquence des moyennes mouvantes convergent.
Cependant le choix du paramètre w peut être un problème difficile, particulièrement pour des systèmes encombrés ayant en sortie des distributions de temps hautement autos corrélées (voir [10], ce qui est le cas d'un trafic self similar !).

Méthodes séquentielles

Pour ce qui est des méthodes séquentielles il faut s'adapter aux conditions spécifiques de chaque simulation, le chapitre 5 développe différentes techniques pour effacer au mieux l'erreur due aux conditions initiales. On peut néanmoins définir les caractéristiques générales d'un période stationnaire. Il existe deux formes de stationnarité :

- De premier ordre si:

  La moyenne du processus pour un intervalle de temps est constante et ne dépend que de la largeur de la fenêtre choisie. Le processus n'a pas de tendance générale. Une série stationnaire est un mouvement qui fluctue autour d'une valeur moyenne constante μ.

- De second ordre si (et c'est celle qui nous intéresse) :

  La fonction d'auto covariance (défini dans 4.1.1, aussi appelée moment d'ordre 2, il s'agit de la covariance classique appliquée sur des intervalles de données de

[1] Ici on changera de notations, car les notations classiques seraient trop lourdes : on ne précise plus le nombre d'observations sur lequel les indicateurs sont établis.

taille m séparés par un espace s dont la formule est donnée par $R_{s,m}$ dans la section suivante), est indépendante de l'origine. La fonction d'auto covariance dans son ensemble exprimera la forme de la dépendance entre les observations séparées par des écarts temporels de plus en plus grands.

En utilisant les notations de 4-A-1 on dira que les variables aléatoires X_1, X_2, …, X_k, représentant chacun m observations sur un paramètre de simulation, forment un processus stationnaire de second ordre si :

1. $\mu_1=\mu_2=\ldots=\mu_k=\mu$, μ_i désignant la moyenne sur X_i.
2. La covariance entre X_i et X_j est égale à la variance de X_h pour $|i-j| = h$.

4.2.2.2. Intervalle de confiance sur période stationnaire

Une fois la période d'initialisation détectée, de même que pour les systèmes à horizon fini, les données recueillies ne sont en général pas indépendantes ce qui implique que les méthodes usuelles de la statistique doivent être appliquées selon des techniques spécifiques.

Technique réplication/délétion

Il s'agit ici de la même procédure que celle décrite pour les systèmes à horizon fini (voir plus haut) à cela prés que sur chaque réplication on supprime les informations recueillies pendant la période « warm up ».

Le grand inconvénient est que pour les systèmes à état d'équilibre la durée de la simulation est un paramètre qui, on le verra plus tard (section 4.3), tend à être conséquent pour obtenir un intervalle de confiance étroit. Si bien que cette technique se révèle très coûteuse si le nombre de réplications est élevé et surtout si chacune des passes de simulation est longue (d'autant plus que sur chaque réplication les données recueillies pendant le warm-up sont inutiles et peuvent ralentir l'expérience de simulation si cette période est longue). La figure ci-dessous illustre alors les deux voies d'analyse qui s'offre à nous, d'une part les techniques séquentielles et de l'autre l'approche parallèle.

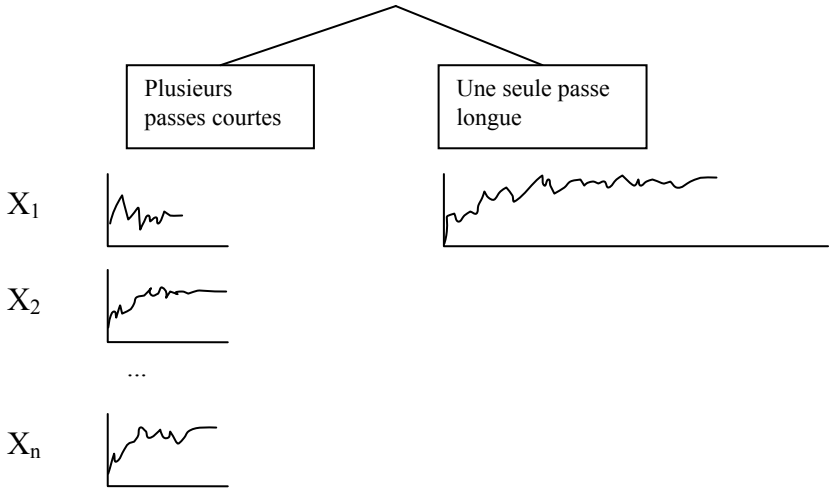


Figure 18 : ***Méthode des réplications face à l'analyse séquentielle.***

Etant donnée le coût de l'approche parallèle pour des résultats de qualité et surtout pour obtenir un intervalle de confiance étroit on préférera les méthodes séquentielles dont la plus connue et la plus simple est la technique de moyenne des lots (« *batch mean* »).

Technique de la moyenne des lots (ML)

Lorsque l'on recueille directement les mesures des variables d'intérêt en sortie, bien que les données introduites en entrée, sous forme de temps de transition par exemple, puissent être IID, la nature de leur dépendance s'en trouve liée. En effet, les mesures sur un paramètre en sortie à un instant donné sont le fruit des événements précédents et par la même des mesures antérieures du même paramètre. Pourtant, d'une période suffisamment longue à l'autre, on observe que la dépendance entre périodes (qu'on appellera lot ou intervalle dans la suite de ce document) « s'atténue » dans la mesure ou l'influence des observations antérieures s'estompent.

La méthode de la moyenne des lots se base sur ces observations et consiste à diviser une passe de simulation recueillant n mesures X_1, X_2, …, X_n sur un paramètre X en plusieurs intervalles de taille identique m. On obtient finalement k intervalles de taille m avec $n = k \times m$, et si l'on note le $i^{ème}$ lot Y_i on peut représenter cette méthode par la figure ci-dessous.

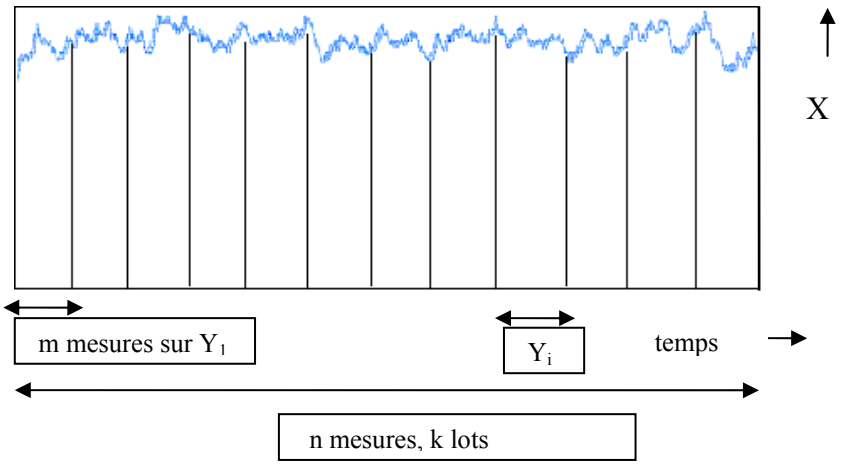


Figure 19 : ***Illustration de la technique de la moyenne des lots.***

Notons $\overline{Y_i(m)}$,

$$\overline{Y_i(m)} = \sum_{j=i}^{i+m} X_j$$

la moyenne de chaque intervalle Y_i, et $\overline{Y(n)}$ l'estimateur de la moyenne théorique μ,

$$\overline{Y(n)} = \sum_{i=1}^{k} \overline{Y_i(m)} = \sum_{i=1}^{n} X_j$$

alors l'estimateur de variance $S_m^2(n)$ pour des lots de taille m,

$$S_m{}^2(n) = \sum_{i=1}^{k} (\overline{Y_i(m)} - \overline{Y(n)})^2 \times \frac{1}{k-1}$$

est non biaisé par rapport à la moyenne $\overline{Y(n)}$ pour k suffisamment grand.

On remarque que l'estimateur de moyenne reste le même appliqué globalement ou par intervalle car il est linéaire alors que l'estimation de la variance, étant de nature quadratique, n'est plus la même que si on l'avait appliquer de manière globale sur l'ensemble des valeurs recueillies en sortie. $S_m^2(n)$, estimateur par lot m sur n observations, est un estimateur plus efficace que $S^2(n)$ (voir 4.1.1).

A présent il suffit d'appliquer les statistiques usuelles vues dans la section 4.1.3 , pour obtenir la demi largeur de l'intervalle I de confiance sur le paramètre X avec α comme niveau de confiance, et en supposant que le nombre d'intervalle est petit (inférieur à 30) et que la taille des intervalles est grande comme recommandé dans Schmeiser [44], on a alors :

$$\frac{I}{2} = t_{\alpha,k-1} \times S_m(n) \times \frac{1}{\sqrt{k}}.$$

Cette technique a donc l'avantage d'être relativement simple en apparence et associée à des procédures spécifiques (voir section durée), elle est sensiblement aussi efficace que les méthodes décrites dans (autres techniques séquentielles).

Néanmoins comment déterminer les paramètres m, k et donc n ?

Pour ce qui est de la taille m de chaque lot il existe un ratio de Von Neumann qui permet d'estimer la dépendance entre intervalles successifs pour que les conditions IID (et ici la dépendance) soient au mieux réunies.

Test de Von NeuMann [34]

Il s'agit de mesurer la corrélation entre les intervalles que l'on souhaite former pour la technique ML. En effet d'un intervalle à l'autre, la covariance entre les résultats obtenus à la fin d'un sous intervalle et les résultas du début de l'intervalle suivant n'est pas nulle. Cependant si les intervalles sont suffisamment longs on peut considérer que les intervalles formés sont indépendants.

Dans un premier temps il faut s'assurer avec un test d'hypothèse approprié, celui de Shapiro-Wilk [32] par exemple, de la normalité des variables sur lesquelles on mesure la corrélation, c'est-à-dire sur les intervalles successifs $\overline{Y_i(m)}$.

A présent, nous pouvons utiliser plusieurs indicateurs de dépendances linéaires nous permettant de juger de la dépendances entre les sous intervalles successifs de taille m que l'on souhaite construire.

Soit $\hat{R}_{s,m}$ un estimé de la covariance entre les variables $\overline{Y_i(m)}$ et $\overline{Y_{i+s}(m)}$ on peut alors écrire avec $0 \le s < k$,

$$\hat{R}_{s,m} = \mathrm{cov}(\overline{Y_i(m)}, \overline{Y_{i+s}(m)}) = \frac{\sum_{i=1}^{k-s} (\overline{Y_i(m)} - \overline{Y(n)})(\overline{Y_{i+s}(m)} - \overline{Y(n)})}{k-s}.$$ [1]

.

Notons $\hat{p}_{1,m}$ une estimation du coefficient de corrélation entre $\overline{Y_i(m)}$ et $\overline{Y_{i+1}(m)}$ alors on peut calculer,

[1] $\hat{R}_{0,m} \approx S_m^2(n)$

$$\hat{p}_{1,m} = \mathrm{corr}(\overline{Y_i(m)}, \overline{Y_{i+1}(m)}) = \frac{\hat{R}_{1,m}}{\hat{R}_{0,m}} = \frac{\sum_{i=1}^{k-1}(\overline{Y_i(m)} - \overline{Y(n)})(\overline{Y_{i+1}(m)} - \overline{Y(n)})}{\sum_{i=1}^{k}(\overline{Y_i(m)} - \overline{Y(n)})^2}.$$ [1]

Le test statistique de Von NeuMann C_k,

$$C_k = 1 - \frac{\sum_{i=1}^{k-1}(\overline{Y_i(m)} - \overline{Y_{i+1}(m)})^2}{2\sum_{i=1}^{k}(\overline{Y_i(m)} - \overline{Y(n)})^2},$$

peut être réécrit comme :

$$C_k = \hat{p}_{1,m} + \frac{\overline{(Y_1(m)} - \overline{Y(n)})^2 + \overline{(Y_k(m)} - \overline{Y(n)})^2}{2\sum_{i=1}^{k}(\overline{Y_i(m)} - \overline{Y(n)})^2}$$

Notons que si k est grand alors $C_k \approx \hat{p}_{1,m}$.

Pour mesurer la signifiance de la dépendance on approxime la distribution de $C_k \times \sqrt{\frac{k^2-1}{k-2}}$ par la distribution normale N (0, 1) pour k>7.[20]
Autrement dit, pour un niveau de confiance de 95% et k>7, on utilise un « β-one sided test » qui s'utilise de la manière suivante[2] :
Si

$C_k \times \sqrt{\frac{k^2-1}{k-2}} \leq (1-\beta) \times 1.96$, alors on peut considérer que les k intervalles de taille m sont indépendants. (En pratique on prendra β très petit, voire nul).
Grâce à ce test on peut donc estimer la corrélation entre intervalles, cependant son utilisation s'inscrit dans une procédure plus large car il faut à présent étudier les différentes possibilités qui s'offre à nous pour faire évoluer la taille des lots (le paramètre m). Nous allons donc étudier plusieurs règles dans la partie 4.3 qui joue sur l'évolution dynamique de la taille des lots en cours de simulation et en fonction du test décrit plus haut.

En ce qui concerne le paramètre n, nombre total de valeurs recueillies durant la simulation sur un paramètre X, il peut être déterminé par un seuil γ de précision, relatif à la moyenne, sur l'intervalle de confiance I, calculé comme décrit précédemment, tel que $\frac{|I|}{2} < \gamma \times \overline{Y(n)}$ ou directement par une valeur absolue v telle que |I|/2<v, ce seuil de

[1] Pour le rappel des définitions et des propriétés de la covariance et de la corrélation se reporter à la section 4.1.1.

[2] Les processus d'arrivée auto similaire et dépendants sur de grands intervalles (voir 3.2.1.3), vont poser problème sur ce test, dans la mesure où le nombre de données à récolter risque d'être exorbitant pour obtenir des lots suffisamment indépendants.

précision est laissé au choix de l'utilisateur en fonction de ses critères d'expérience de simulation.
Nous étudierons plus en détail, dans la section 4.3, comment décider à l'avance si possible, du nombre de données qu'il faut récolter en sortie pour obtenir des résultats correspondants aux besoins en précision des objectifs d'étude.

Autres méthodes séquentielles

Dans cette section nous allons brièvement aborder d'autres méthodes du même type pour construire des intervalles de confiance valide. (Voir [9] et [10])

Méthode régénérative

Cette méthode se base sur des *cycles de régénération*. Elle cherche à identifier les périodes de temps sur lesquelles la famille d'éléments $\{X_i\}$ étudiée semble se régénérer (en termes intuitifs, le paramètre X se comporte selon une caractéristique du système qui se charge puis se vide). On obtient alors un ensemble d'indices T_j marquants le début et la fin d'un cycle dont la distribution est la même quelque soit le cycle et indépendante du précédent (si le système simulé s'y prête). La Figure 20 illustre ce phénomène cyclique (avec en rouge les points de régénération).

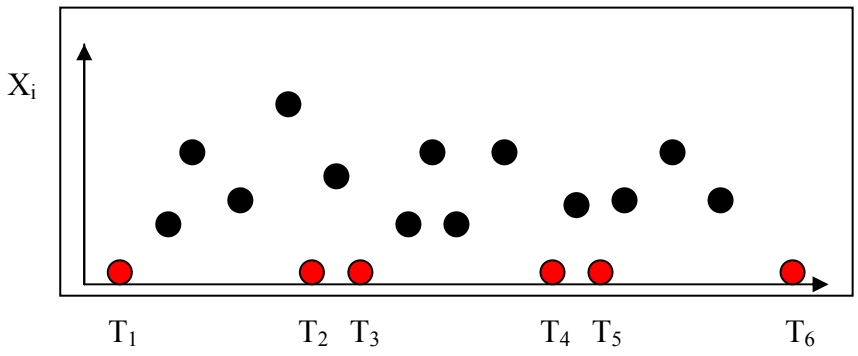


Figure 20 : ***Cycles et indices de régénération***

Pour construire une estimation solide de la moyenne on prendra alors l'estimateur E(X)=E(Y)/E (Z) sur n cycles, avec :

$$E(Y) = \frac{\sum_{i=1}^{n} \sum_{j=T_i}^{T_{i+1}-1} X_j}{n} \quad \text{et} \quad E(Z) = \frac{\sum_{i=1}^{n} T_{i+1} - T_i}{n}$$

Ainsi on peut considérer ces différents cycles comme IID et appliquer les statistiques usuelles sur la variable $Y-\mu Z$ en se basant sur le théorème centrale limite.
En pratique cette technique est peu voire pas utilisée, car les systèmes simulés ayant des points de régénération sont très rares, ou sinon leurs cycles sont extrêmement longs. Les systèmes d'inventaires de stock sont par exemple un des rares domaines sur lequel on peut appliquer ce type de méthode. (Voir [9]).

Séries temporelles standardisées

Cette méthode, proposée par Schruben [39], consiste à diviser les données analysées $\{X_i\}$ sur une passe de simulations produisant n données en k lots contiguës notés A_i i.e (n=k*b) pour i=1 à k:

$$A_i = \sum_{j=1}^{b} [(n+1)/2 - j] X_{(i-1)b+j}$$

.

On peut alors considérer ces lots comme IID si l'on a recueilli suffisamment d'informations (n grand) alors un bon estimateur de $E(A^2)$ (il s'agit d'une estimation de la variance du processus divisé par un facteur 12) est :

$$\hat{E}(A^2) = \frac{1}{(b^3 - b) \times k} \sum_{i=1}^{k} A_i{}^2 .$$

Et ainsi on peut calculer la demi largeur de l'intervalle de confiance pour la moyenne théorique :

$$t_{k,\alpha/2} \sqrt{12 \hat{E}(A^2)/n} .$$

Cette méthode semble avoir des avantages qualitatifs sur les procédures classiques ML (voir durée de simulation) mais en pratique le nombre d'informations à recueillir peut être exorbitant. Pour les fondements statistiques et mathématiques se référer à [10] et [39].

Analyse spectrale

Il s'agit ici d'estimer la corrélation structurelle entre les valeurs, prises par une variable d'intérêt, recueillies en sortie afin de réévaluer la variance pour une analyse statistique classique. En utilisant les notations de la ML ($n = k \times m$), la variance S^2_m (n) prend alors la forme suivante à condition que le processus étudié soit stationnaire d'ordre 2 (on parle de stationnarité d'auto covariance,voir section 4.2.2.1 et [9]) :

$$S_m{}^2(n) = \frac{1}{k}\left(\sigma^2 + 2\sum_{i=1}^{k-1} (1 - \frac{i}{k}) \hat{R}_{i,m} \right)$$

Avec $\hat{R}_{i,m}$ défini de la même manière que dans la section 4.2.2.2 *Ratio de Von Neumann*, il s'agit en fait de la covariance entre intervalles espacés de $i \times m$ données.

En utilisant la variance mesurée on obtient :

$$S_m{}^2(n) = \frac{1}{k}\left(\hat{R}_{0,m} + 2\sum_{i=1}^{k-1} (1 - \frac{i}{k}) \hat{R}_{i,m} \right)$$

On peut remarquer que la convergence de $k \times S^2{}_m(n)$ lorsque k→ ∞, vers

$$\sum_{-k}^{k} \hat{R}_{i,m}$$

dépend de la fonction de pondération

$$W_{n,i} = 1 - \frac{|i|}{n}$$.

A présent il suffit d'utiliser les statistiques usuelles en utilisant cette formule de variance pour obtenir l'intervalle I de confiance pour un niveau de confiance α, on obtient :

$$|I/2| = t_{\alpha/2,k-1} \times S^2{}_m(n)/\sqrt{k}\ .$$

(si le nombre de lots est k >30 on utilise $z_{\alpha/2}$).

L'avantage de cette technique est donc une estimation exacte de la variance (après suppression du warm up) et son analogie en terme de décomposition en intervalles avec la méthode ML.

On peut en outre utiliser des transformations en séries de Fourier pour accélérer le calcul des covariances qui peut se révéler coûteux si le nombre de lots est grand.

Overlapping Batch mean(OBM)

Cette technique développée par Meketon et Schmeiser [40] est une variante de la ML, qui consiste d'une part à diviser chacun des lots en sous lots, et d'autre part à analyser les lots formés en les superposant. Le document [10] nous en donne les principaux fondements :
pour une taille de lots m donné, cette méthode utilise n-m+1 lots pour estimer la moyenne et la variance théorique d'un paramètre X du système étudié.
Le premier lot formé rassemble les informations $X_1, X_2, \ldots, X_m$, le second les observations $X_2, \ldots, X_b, X_{m+1}$, et ainsi de suite jusqu'au n-m+1 ième lot.
L'estimateur de moyenne OBM de la moyenne théorique est alors :

$$\overline{Y_{OBM}} = \frac{1}{m-b+1} \sum_{i=1}^{m-b+1} \overline{Y_i(m)}$$,

où $\overline{Y_i(m)} = \frac{1}{m} \sum_{j=i}^{i+m-1} X_j$ pour i=1, …, n-m+1 sont les lots décrits précédemment.

Pour le calcul de l'intervalle de confiance sur la moyenne mesurée, on utilisera la simple variance sur les intervalles $Y_i(m)$, notons que le comportement de cette variance semble moins sensible aux choix des lots que celui de la variance issue de la ML classique [10].
Welch a remarqué que la ML ou OBM sont des cas spéciaux de l'estimation spectrale à la fréquence 0, et il suggère pour obtenir une réduction optimale de la variance, de diviser les lots formés en sous lots et appliquer OBM au lots principaux espacés. Par exemple pour des lots rassemblant 128 observations, on les divisera en 4 sous lots distincts et le premier lot OBM sera de la forme $X_1, X_2, \ldots, X_{128}$, le second $X_{33}, X_{34}, \ldots, X_{160}$,etc.

Autres Méthodes séquentielles

Il existe encore de nombreuses méthodes basées sur la même idée générale que la ML comme les méthodes auto régressives qui analysent la présence de cycles dans la fonction d'auto corrélation ou encore des variantes de la méthode des séries temporelles selon le modèle de loi choisi (par exemple, ARMA ou *loi des moyennes mouvantes*, voir [9]).
Une autre méthode est à noter par son originalité, il s'agit de la méthode antithétique qui se base sur deux passes (ou expérience selon le nombre de réplications dans les deux cas) de simulations distinctes, l'une avec la variable aléatoire X tirée dans U (0, 1) et l'autre avec la variable 1-X qui sera elle aussi comprise dans [0, 1]. La moyenne et la variance sont alors calculés sur l'ensemble des deux passes de simulation entraînant ainsi une réduction de l'erreur échantillonné grâce à la corrélation négative présente entre les deux passes.

De manière générale, la ML dans sa version de base semble produire des résultats aussi satisfaisants [10] que d'autres méthodes plus complexes (dont certaines ont été évoquées ici), à condition qu'elle s'inscrive dans une procédure de simulation plus large.(Voir 4.3.2)

4.3. DUREE DE SIMULATION

Cette section traite de la quantité d'informations à recueillir pour satisfaire certains critères qualitatifs en termes de mesures de performance, par exemple, pour évaluer et comparer la qualité de service que peuvent offrir deux protocoles différents dans le cadre d'un simulateur de réseau IP. En effet si l'on veut pouvoir tirer des conclusions sur les moyennes mesurées sur un paramètre quantitatif comme le délai moyen d'acheminement des paquets il faut que les intervalles de confiance calculés pour chacun des protocoles ne se chevauchent pas. (Voir Figure 21).

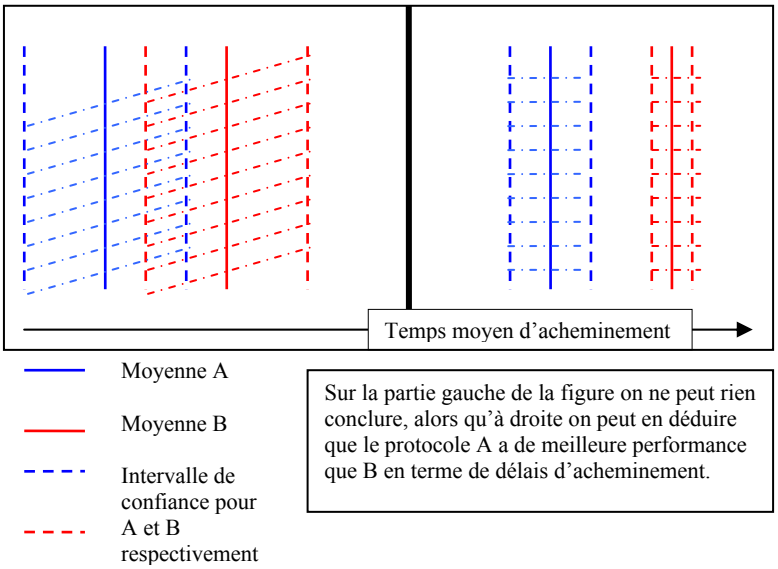


Figure 21 : ***La validité des résultats dépend de la largeur de l'intervalle de confiance.***

De même que pour les calculs d'intervalle de confiance, selon que le système se termine ou non les approches divergent :

Pour un système se terminant il existe une méthode qui permet, en étudiant le système sur une expérience de simulation relativement courte (le nombre de réplication k est faible), d'anticiper le temps nécessaire pour obtenir un intervalle de confiance suffisamment étroit.
Pour un système ne se terminant pas, la loi forte des nombres ($\lim n \rightarrow \infty X(n)=\mu$) nous indique la voie à suivre : plus on récolte de données, plus les calculs de moyennes sont précis mais en pratique on ne peut pas récolter des données à l'infini, il va donc falloir définir des algorithmes nous permettant d'obtenir, en procédant par itération, un intervalle de confiance assez étroit relativement aux besoins de l'utilisateur de la simulation, selon ses objectifs d'études.

4.3.1. Pour les systèmes à horizon fini

Les 4 étapes qui suivent constituent la démarche à suivre pour obtenir le nombre de réplications à effectuer (donc la durée de l'expérience de simulation) pour l'intervalle de confiance désiré I avec un niveau de confiance α. (Notons qu'on utilise une approximation de la variance, il faut donc répéter ces 4 étapes jusqu'à que I soit atteint).

- Prendre k un entier positif > 1. Et notons $w = t_{\alpha/2,\, k-1} / I$.

- Observer la famille d'éléments étudiés sur X_1, X_2, …, X_k. (chacun des X étant déjà une moyenne sur une passe de simulation sur m mesures).
- Calculer la moyenne et la variance sur ces k observations, et si on note $S^2_m(k \times m)$ la variance, alors on note K= max (k+1, $\lfloor (w \times S_m(k \times m))^2 \rfloor$).
- Alors il faut encore répliquer, si K ≠ k+1, K-k simulations, 0 sinon car l'intervalle de confiance est suffisamment étroit avec k simulation.

4.3.2. Pour les systèmes à état d'équilibre

Comme décrit dans le chapitre précédent la méthode de la moyenne des lots nécessite l'estimation de la corrélation inter lots et un test de normalité sur intervalles, d'autre part le test « final » doit calculer selon les méthodes usuelles (section 4.1.3) la largeur de l'intervalle de confiance, ainsi au cours de ces différentes étapes il faut recalculer le nombre d'intervalle k et le nombre total, n, d'informations à recueillir pour que tout ces test soient satisfaits. Cette section va donc traiter des différentes procédures possibles à mettre en place entre les tests pour modifier la taille des paramètres k et n.

Préambule

Law et Kelton [43] font figure de pionniers dans les nombreuses procédures développées sur la base de la ML, ils ont définis les premières techniques (des algorithmes « statiques » avec des suites arithmétiques) pour faire évoluer la taille des lots selon la réussite ou l'échec sur le test de Von Neumann. Aujourd'hui il existe plusieurs règles (*rules*) pour modifier les paramètres n, k et m, ainsi que des procédures construites sur ces règles pour obtenir l'intervalle de confiance désiré.

FNB rule

La règle du nombre fixe de lots (*fix number of batch*) se définit comme suit :
On fixe définitivement le nombre de lots pour toute la simulation à k=K, ainsi lorsque $n \to \infty$ et $m \to \infty$ l'erreur systématique due à la dépendance des lots va s'estomper et sous des conditions assez généralement satisfaites on peut construire un intervalle de confiance valide.

The Square Root (SQRT) rule

Cette règle essaye de sélectionner k et m tel que :
$k \approx \sqrt{n}$ et $m \approx \sqrt{n}$ lorsque $n \to \infty$.
Cette technique garantit, lorsque n suffisamment grand, que l'erreur due au fait de l'ignorance des résidus de corrélation inter lots diminue ($m \approx \sqrt{n}$) et que l'erreur due à la pseudo génération s'estompe ($k \approx \sqrt{n}$).

Pourtant l'utilisation de l'une de ces deux règles, telles quelles, tend à sérieusement négliger un certain nombre de facteurs mis en avant par Fischmann (voir [10]).
D'où Fischmann et Yarberry [45] ont eu l'idée d'alterner entre ces deux règles, proposant ainsi deux procédures de calcul, basées sur la notion de passage en revue, comparables à des points de vérification, espacés dans le temps, sur l'ensemble des mesures antérieures :

4.3.2.1. LBATCH procedure[1]

- On commence par utiliser la FNB rule sur le premier passage en revue j=1des intervalles formés avec les paramètres initiaux.
- On effectue un test de dépendance :
  - s'il est rejeté, on utilise la FNB rule sur le passage en revue suivant j+1.
  - s'il est accepté, on utilise la SQRT rule sur tout les passages en revue suivant j+1, j+2,... en ignorant pour toutes ces futures revues le test d'indépendance.

4.3.2.2. ABATCH procedure[2]

- On commence par utiliser la FNB rule sur le premier passage en revue des intervalles formés avec les paramètres initiaux.
- On effectue un test de dépendance :
  - s'il est rejeté, on utilise la FNB rule sur le passage en revue suivant j+1.
  - s'il est accepté, on utilise la SQRT rule sur le passage en revue suivant j+1.

Ces deux procédures dynamiques proposent des avantages que les méthodes statiques (bien que pour ce type d'algorithme la complexité soit du même ordre $\leq O(n)$) ne présentent pas, elles offrent en effet la possibilité de contrôler le moment où la variance, en cours de calcul, commence à se stabiliser, et ainsi d'avoir une notion de la validité de l'intervalle de confiance formé.

Il existe également des alternatives à ces procédures, qui sont elles aussi des méthodes séquentielles (voir [18]et [19]) dont l'avantage réside principalement dans le re-calcul de la taille et du nombre de lots et aussi sur un test de normalité multi variables durant la passe de simulation. Nous allons brièvement les détailler.

4.3.2.3. ASAP procedure

Cette procédure proposée par Steiger et Wilson existe en plusieurs versions (ASAP, ASAP2, ASAP^{+}…) dont la première est décrite dans [18]. Le schéma ci-dessous décrit la procédure complète ASAP. Les parties en bleu ciel sur le schéma vont être détaillées dans

[1] et [2] Sous ces deux procédures les paramètres m et k évoluent selon des séquences aléatoires, dont le comportement est décrit sous forme de pseudo code dans [45], de plus, les passages en revue (ou « Checkpoint ») ont lieu au temps $n_l \approx 2^{l-1} n_1$ pour l=1,2… .

cette section, le reste étant similaire aux méthodes vues précédemment. ASAP propose également des paramètres initiaux génériques i.e k=96, m=16 avec les notations précédentes.

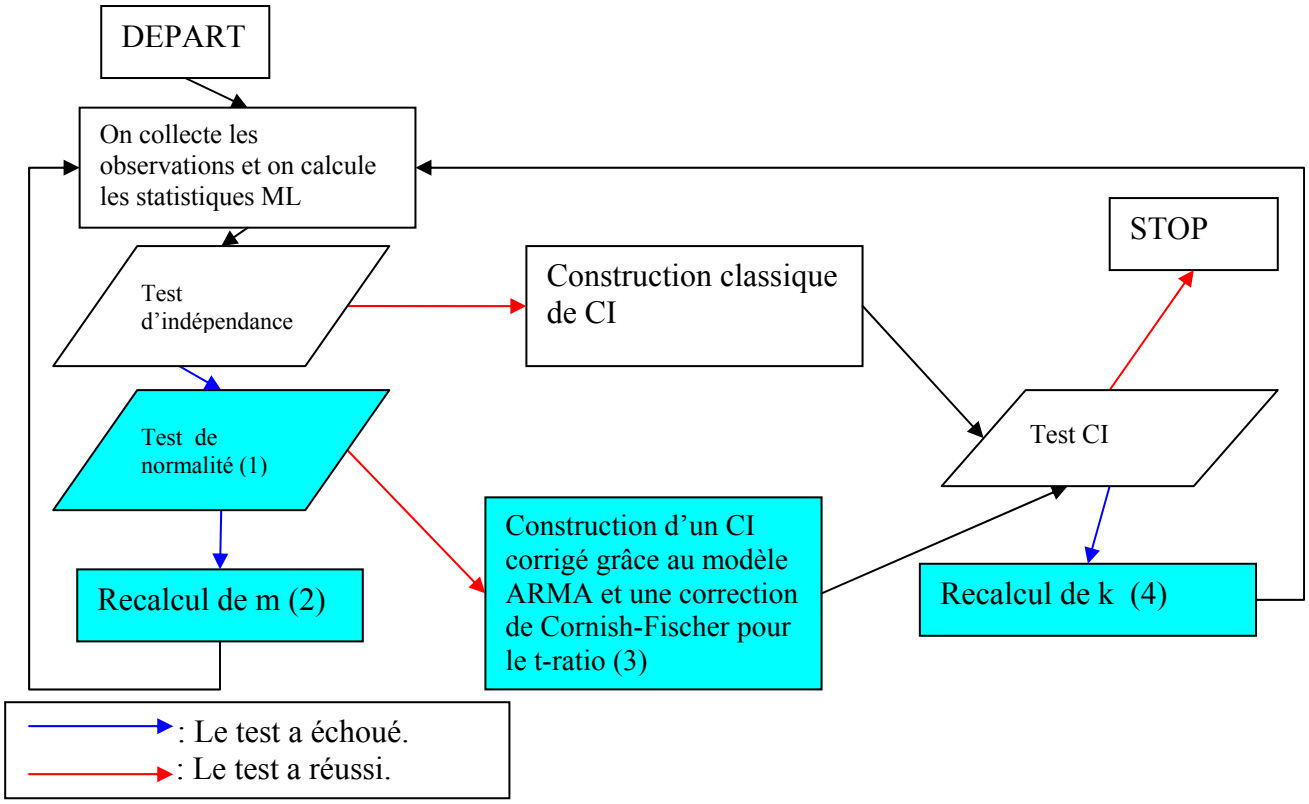


Figure 22 : ***Vu d'ensemble de la procédure ASAP.***

1. *Test de normalité multi variables*

Si le test d'indépendance échoue, ASAP va construire 16 vecteurs constitués chacun de 4 lots consécutifs. D'autre part les vecteurs sont espacés par 2 lots qui sont ignorés. On obtient ainsi : $Y_3(m)$, $Y_4(m)$, $Y_5(m)$, $Y_7(m)$, $Y_8(m)$, $Y_9(m)$, $Y_{10}(m)$, $Y_{11}(m)$, $Y_{12}(m)$, $Y_{13}(m)$,…, $Y_{93}(m)$, $Y_{94}(m)$, $Y_{95}(m)$, $Y_{96}(m)$.(avec en rouge les vecteurs)
On applique à présent un test de Shapiro-Wilk sur ces 16 vecteurs en procédant comme décrit dans [18] avec certains facteurs du test obtenu par expérimentation.

2. *Recalcul de m*

Si le test décrit au dessus échoue à son tour, de la même manière dont on avait noté les passages en revue (j) pour les procédures ABATCH et LBATCH, on modifie les paramètres m_j et k_j de la façon suivante :

i. $m_{j+1} = \lfloor v2 \times m_j \rfloor$ et $k_{j+1} = k_j$.

3. *Construction d'un modèle de séries temporelles pour lots dépendants et ajustement de l'intervalle de confiance*

En revanche si le test de Shapiro-Wilk nous donne un résultat positif, on va utiliser un modèle de séries temporelles autorégressif sur moyennes mouvantes (ARMA) pour

estimer avec le plus de précision possible la variance du processus étudié. Une fois la variance ré estimé selon la méthode définie dans [18] on utilise l'expansion inversée de Cornish-Fischer pour évaluer la largeur de l'intervalle de confiance en fonction de la nouvelle variance. (Voir [18] pour plus de détail).

4. Recalcul de k

Soit H^* la demi longueur de l'intervalle de confiance calculé selon la réussite à l'un des deux tests précédents et H l'intervalle de confiance que l'on désire. (On est donc dans le cas $H^* < H$ car le Test CI a échoué).

Admettons que nous sommes à l'itération j dans ASAP ($j^{ème}$ passage en revue) alors :

$$k_{j+1} = \left\lceil (H / H^*)^2 \right\rceil \times k_j \text{ et } m_{j+1} = m_j \text{ d'où } n_{j+1} = m_{j+1}\, k_{j+1}.$$

L'avantage de ASAP[1] par rapport à LBATCH ou ABATCH réside surtout dans sa qualité d'assurance (ou coverage, voir section 4.4.2) comme semble l'indiquer le Tableau 5 sur l'exemple d'une queue M/M/1.

Niveau de précision sur le CI	Procédures		
	LBATCH	ABATCH	ASAP
Aucune précision			
Coverage	44%	60%	83%
Moyenne de CI/2	1.7	2.67	11.8
Variance de CI/2	0.683	3.92	523.0
15 % de précision relative			
Coverage	79%	80%	88%
Moyenne de CI/2	0.543	0.613	0.783
Variance de CI/2	0.027	0.039	0.082
7,5 % de précision relative			
Coverage	88%	90%	94%
Moyenne de CI/2	0.353	0.382	0.413
Variance de CI/2	0.012	0.039	0.018

Tableau 5 : Performance de différentes procédures pour une queue M/M/1 avec un taux de 0.9 basé sur 100 réplications indépendantes avec un niveau de confiance de 90%.(tiré de Steiger).

CI : intervalle de confiance, précision relative = erreur relative (section 4-3).

Par contre sur des petits échantillons le test de normalité a tendance à être réussi avant le test d'indépendance créant à priori une meilleure assurance mais des intervalles de confiance plus larges. (Voir Tableau 5 et [10])

Par conséquent ASAP nécessite des échantillons de taille beaucoup plus grande (et donc des simulations de durée plus élevée) que ABATCH et LBATCH pour obtenir des intervalles de confiance respectant la même erreur relative (voir Tableau 5).

[1] Il est à noter que ASAP pose en plus des paramètres initiaux une limite supérieure sur nombre de lots ($k \leq 1502$) ce qui nous conduit, lorsque ces limites sont atteintes, à redéfinir la taille m de chaque lot (voir [18]).

4.3.2.4. WASSAP procedure

Cette procédure a également été développée par M.Steiger, R.Wilson ainsi que K.Lada [19].
WASSAP est basé sur la méthode ML et détermine une période warm-up à supprimer ainsi que la taille des lots nécessaires pour que les lots étudiés forment un processus stationnaire gaussien. Comme ASAP elle propose un mécanisme pour tester la normalité des variables mais il faut au préalable que les lots soient considérés comme indépendants en fonction du ratio de Von Neumann, à noter que ces tests sont évalués sur des lots espacés par une variable S dont le comportement est décrit dans [19].
De plus, si les deux test sont réussis (en effet il est impératif que le test d'indépendance et le test de normalité soient satisfaits à la différence de la procédure ASAP où le succès à l'un des deux tests est suffisant), on n'aura plus à justifier de leurs succès jusqu'à la fin de la procédure.
D'autre part, et c'est là que WASSAP apporte une réelle contribution en terme de qualité, le calcul de l'intervalle de confiance se fait selon une approximation de la variance du processus original via une analyse spectral logarithmique des lots basé sur des séries de vaguelettes (voir [19]) (sur les lots définis avec les deux tests qui ont précédés et cette fois ci non espacés et privés des S premiers lots considérés comme la période d'initialisation du processus, le warm-up).
Ce calcul est répété jusqu'à obtenir l'intervalle de confiance souhaité par l'utilisateur en modifiant les paramètres m et k de la même manière que dans ASAP.
Au vu des résultats publiés dans [19] sur une queue M/M/1, WASSAP¨est une méthode qui semble plus robuste que ASAP en particulier pour produire des intervalles de confiance plus étroit avec un coverage à peine inférieur. Comparé à l'analyse spectrale classique (dévellopé par Heidelberg et Welch [14]) donné en 4.2.2.2, les résultats obtenus sur ces deux critères sont plus performants mais nécessitent des simulations plus longues en général.

Dans le cadre des réseaux à commutations de paquets, le chapitre 5 développe une procédure spécifique appliquée sur plusieurs paramètres représentatifs des délais moyens d'acheminement des paquets.

4.4. RECHERCHE MULTI VARIABLES ET ASSURANCE

4.4.1. Multi target tracking (ou recherche multi variables)

Les procédures que nous avons étudiées auparavant ne s'appliquent qu'à une famille d'individus Xi représentant une variable d'intérêt pour nos objectifs d'études. Or en réalité, lors d'une simulation, on s'intéresse à de nombreuses variables, il existe alors plusieurs techniques nous permettant de construire des intervalles de confiance sur n paramètres d'études. En général on peut utiliser l'inégalité de Bonferroni pour calculer un coefficient de confiance sur un ensemble de variables (voir [10]). Notons D_i un intervalle de confiance pour un niveau de confiance $1-\alpha_i$ pour la moyenne μ_i d'un paramètre d'intérêt. Alors

$P[\bigcap_{i=1}^{n}\{\mu_i \in D_i\}] \geq 1 - \sum_{i=1}^{n}\alpha_i$ (P exprime la probabilité que l'ensemble des moyennes théoriques recherchées appartiennent effectivement à leurs intervalles de confiance respectif).
Ce résultat peut entraîner des complications si par exemple n=20 et α=0.05 car la partie droite de l'égalité est alors égale à 0.
Pour n=2, en 2 dimensions, on peut parler de boite de Bonferroni et d'ellipse de confiance pour contrôler le produit des $\alpha_{i's}$.
Prenons QoSim comme exemple, et admettons que nous étudions d'une part les délais d'acheminement des paquets d'un service privilégié et d'autre part la même mesure sur un service normal. Soit $\{a_1, a_2\}$ l'intervalle de confiance formé sur le premier paramètre et $\{b_1, b_2\}$ sur le second, alors la figure 42 ci-dessous illustre la construction d'une boite et d'une ellipse de confiance.

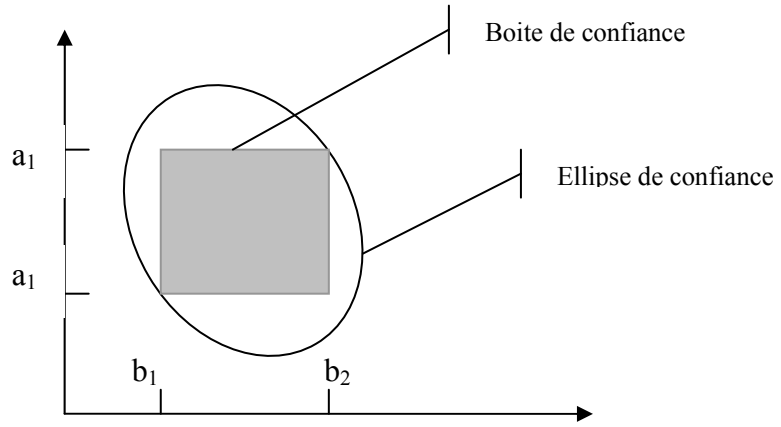


Figure 23 : ***Intervalle de confiance en deux dimensions.***

En n dimensions, on peut continuer à travailler avec ces deux notions, l'utilisateur définit alors une aire, un volume, un hyper volume … (selon sa dimension) de confiance à atteindre par l'ensemble des paramètres d'intérêt. (C'est-à-dire que l'unité de confiance calculée doit être assez étroite aux yeux de l'utilisateur pour que la simulation puisse s'arrêtée). [12] compare plusieurs algorithmes séquentiels usants d'ellipses de confiance pour la recherche multi cibles.
La section 5.3.2 évoque une méthode spécifique pour obtenir des résultats fiables sur l'ensemble des variables d'intérêt.

4.4.2. Coverage(ou assurance)

Il s'agit en fait d'une mesure de l'assurance qu'on peut porter à l'intervalle de confiance formé par des méthodes et des procédures qui, on l'a vu plus tôt, sont approximatives. Le coverage se définit comme la fréquence relative à laquelle l'intervalle de confiance formé couvre effectivement la vraie moyenne théorique (cette valeur que l'on notera c est comprise entre 0 et 1). Cette mesure ne peut donc se faire que sur des systèmes analysables de manière analytique, comme une queue M/M1 par exemple, dans la mesure où la moyenne théorique doit être connue.

On peut également, de la même manière que pour une moyenne, définir un intervalle de confiance I pour le coverage, $I = [c - z_{\alpha/2} \times \sqrt{(c(c-1)/n_c)}, c + z_{\alpha/2} \times \sqrt{(c(c-1)/n_c)}]$.

Pawlikowski [13] rassemble un certain nombre de résultats intéressants, il semble d'une part que le nombre de réplications nécessaire pour obtenir un coverage c de bonne qualité ($c \geq 0.95$ par exemple) soit très important avec la technique ML aussi bien qu'avec les méthodes d'analyse spectrale, et d'autre part il propose une analyse « séquentielle » (on utilise tout de même des réplications) pour mesurer le coverage de l'intervalle de confiance.

Cette technique est basée sur les règles suivantes :

- R1.Le coverage peut être analysé séquentiellement i.e son analyse s'arrête lorsque l'erreur relative associée à son intervalle de confiance satisfasse un certain niveau de précision.
- R2.Une estimation du coverage doit être calculée à partir d'un échantillon représentatif de données, et son analyse doit donc commencer seulement après « qu'un minimum de mauvais intervalles de confiance »sur c soit calculé.
- R3.Les résultats d'une simulation qui est clairement trop courte ne doivent pas être pris en compte.

En pratique on peut déterminer le nombre n minimum de mauvais intervalles de confiance à enregistrer (R2) selon la méthode définie dans [13] (n=200), et dès lors décider du seuil s minimum auquel on considère qu'une simulation est trop courte (R3), afin d'éliminer toutes les réplications dont la durée est inférieure à s et ainsi commencer l'analyse « séquentielle » sur les réplications dont la durée est suffisamment longue au vu de ces deux critères(ils sont en effet liés selon la méthode décrite dans [13])jusqu'à ce que l'erreur relative sur c satisfasse le niveau de précision convenu (R1).

Cette technique est en fait plus dynamique que séquentielle, mais elle présente l'avantage de sélectionner les réplications dont la durée est longue et au vu des résultats présentés dans [13], c'est un signe de qualité en terme de mesure de coverage et surtout plus représentatif des réelles performances des procédures définies dans 4.3.

5. APPLICATION AUX RESEAUX, L'EXEMPLE QoSim

5.1. PRESENTATION GENERALE

Dans le but de mieux comprendre la méthodologie de modélisation ainsi que les méthodes de traitement statistique décrites dans cette état de l'art, nous allons étudier dans ce chapitre un exemple de simulateur à événements discrets : un simulateur de réseau IP pour mesurer les différentes politiques de qualité de service (QoS) (voir [35]).
Cette simulation se déroule au niveau datagramme IP, on modélise une certaine topologie réseau, constitué d'un ensemble de N nœuds « de routage » indépendant, sur laquelle on injecte un trafic sous la forme d'un ensemble de flux de paquets.
Chacun des nœuds gère des files d'attente de type FIFO où sont stockés les paquets avant d'être propagés vers le nœud suivant.
Une fois la topologie et le trafic définis on veut pouvoir tester différentes politiques de routage avec des mécanismes comme TCP et son évitement de congestion où d'autres techniques de QoS.
La Figure 24 décrit la procédure de simulation et ses différentes phases.

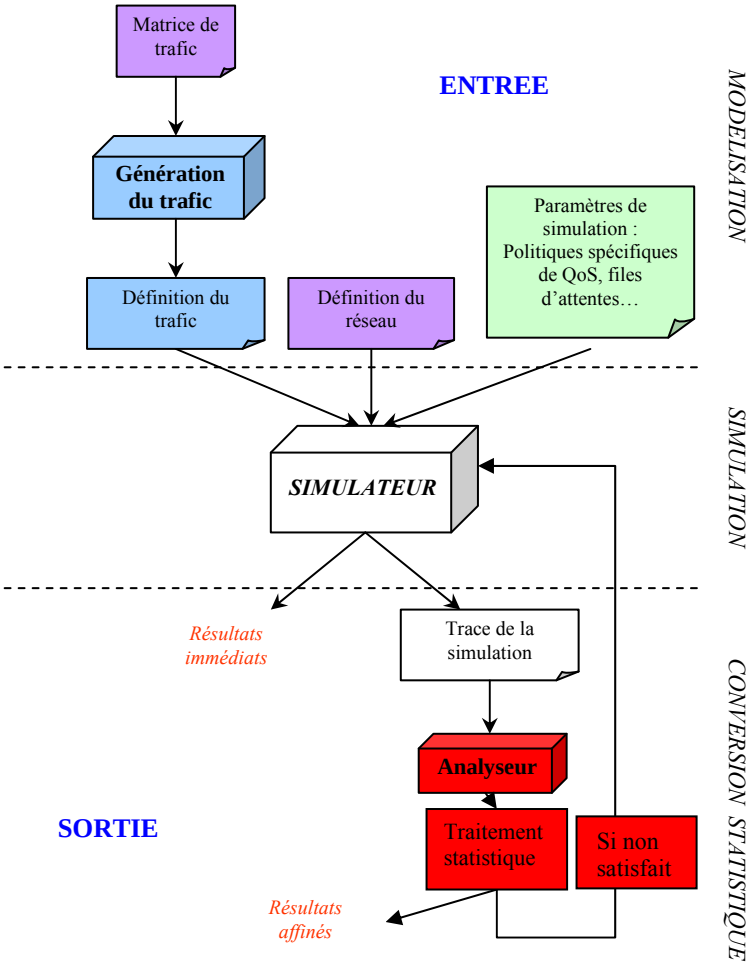


Figure 24 : ***Une procédure de simulation de réseau IP.***

5.2. DONNEES EN ENTREE ET MOTEUR DE SIMULATION

Dans cette sous partie nous allons mettre en pratique les concepts mis en avant dans la méthodologie de modélisation décrite dans le chapitre 3.
Comme le Figure 24 le souligne, on peut décomposer la modélisation du réseau simulé en trois axes principaux : sa topologie, la nature du trafic qui transit et les différents paramètres de simulation (politique de qualité de service (QoS), routage, files d'attentes…).

5.2.1. Topologie

Afin de modéliser les caractéristiques « statiques » d'un réseau IP il va falloir définir un certain nombre d'objets et leurs attributs comme :

- les nœuds de routage (avec leurs files d'attentes, leurs services de routage…).
- les liens (avec leurs couples de nœud (source, destination), leur débit…).
- les flux (avec leurs tailles, leurs privilèges, leur décomposition en paquets…).

Et sans oublier de typer correctement chacun des attributs selon les critères définis dans 3.

Si l'on souhaite modéliser une congestion on peut par exemple définir un réseau basé sur 7 nœuds en « étoile » générant un certain trafic (voir 3-B-1) dont chacun est constitué d'une queue classique de type M/M/1 dont la file d'attente est de capacité finie. En effet si la file est infinie les débordements sont impossibles. La topologie choisie est représentée sur la Figure 25 ci dessous.

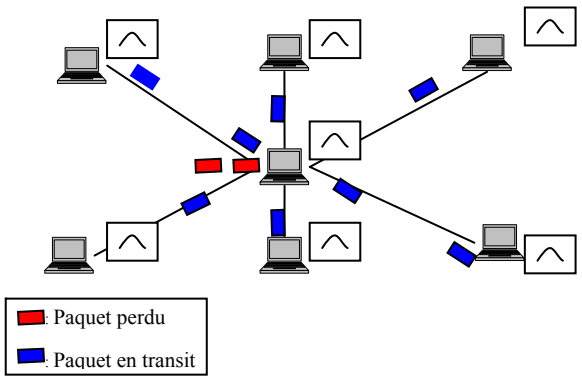


Figure 25 : ***Exemple de modèle topologique pour une simulation de congestion.***

Une mesure de congestion peut être le nombre de paquets perdus sur le nœud central. L'objet nœud aura par conséquent un attribut nommé Paquet_perdu de type entier positif dont le nombre croît de 1 à chaque arrivée de paquet sur lui lorsque sa file d'attente est déjà pleine. Afin de définir pleinement un tel modèle on pourra s'inspirer de la méthodologie conique (CM) décrite dans 3c et on peut obtenir pour un nœud :

Objet	Attributs	Nature	Type
Nœud M/M/1	Moyenne d'arrivée	Indicatif temporel permanent	Réel positif
	Moyenne de service	Indicatif temporel permanent	Réel positif
	Prochaine arrivée	Indicatif temporel transitionnel	Réel positif
Serveur	Etat	Indicatif transitionnel d'état	(Libre, occupé)
	Taille_file	Indicatif permanent d'état	Entier positif
	Nb_file	Indicatif transitionnel d'état	Entier positif
	Nb_servi	Indicatif transitionnel d'état	Entier positif
	Nb_max_servi	Indicatif permanent d'état	Entier positif
	Paquet_perdu	Indicatif transitionnel d'état	Entier positif

Tableau 6 : Décomposition en objet (inspiré de la CM) d'un nœud M/M/1.

Les attributs de haut niveau de ce modèle sont d'une part des moyennes, une sur les inter arrivées de clients dans la queue (Moyenne d'arrivée) et une sur les temps de service (Moyenne de service), et d'autre part une variable de temps représentant la prochaine arrivée de clients. En ce qui concerne l'objet Serveur, sous objet de nœud M/M/1, on défini son état (Etat), la taille de sa file d'attente (Taille_file), le nombre de clients actuellement en attente (Nb_file), le nombre de clients déjà servis (Nb_servi), le nombre maximum de clients à servir (Nb_max_servi), et pour finir le nombre de clients rejetés en raison d'un débordement dû à la capacité finie de la queue (Paquet_perdu, à noter que cette variable n'évolue que si Nb_file=Taille_file).

L'objet « nœud M/M/1 » s'inscrit lui aussi dans une structure plus large contenant 7 nœuds et dont un des attributs serait le temps du système, de nature indicative transitionnelle temporelle (*time-based-signal ou system time),* l'estampille de temps global du réseau modélisé.

Dans le cadre du simulateur QoSim, le réseau, (il est possible d'en définir d'autres) sur lequel les simulations sont effectuées, est formé de 26 nœuds, représentants de manière simplifiée un réseau national de communication. Afin de modéliser les demandes (et leurs charges) qui unissent les différents nœuds on utilise *une matrice de trafic* associée, $M_{i,j}$ i.e la valeur présente à l'indice (i, j) représente la demande en octet du chemin reliant le nœud i au nœud j.

	N1	N2	N3	…
N1	0	10Mo	100Mo	…
N2	10Mo	0	0	…
N3	50Mo	10Mo	0	…
…	…	…	…	0

Figure 26: ***Exemple de matrice de demande (ou de trafic).***

5.2.2. Trafic sur QoSim

Cette section va définir les processus d'arrivée de flux en fonction de la matrice (de dimension $N \times N$ si N nœuds) de trafic et les différents événements régissant la vie du réseau (en d'autres termes la CS vu dans 3.1.4.2).

Pour commencer nous allons définir une matrice de probabilités $P_{i,j}$ telle que [1]:

$$P_{i,j} = \frac{M_{i,j}}{D}$$

, avec $M_{i,j}$,[2] matrice de trafic (voir 5.2.1) et D, débit total en entrée par unité de temps, vaut :

$$D = \sum_{i=1}^{n} \sum_{j=1}^{n} M_{i,j} .$$

D'une part, le temps qui sépare les flots qui vont transiter sur le réseau suit une loi exponentielle (voir 3.2.1.1) de paramètre $\lambda = D / V_{moy}$.
Avec $V_{moy}=(V_{min} + V_{max})/2$, car le volume d'un flot est tiré dans $U(V_{min}, V_{max})$.
Un couple (source, destination) est alors tiré en respect avec la matrice $P_{i,j}$.
D'autre part, la fragmentation en paquet d'un flot est elle aussi tirée selon une loi uniforme U (1600, 8000) c'est-à-dire que la taille d'un paquet oscille entre 200 et 1000 octets, et pour finir, le flot est estampillé à un service (de plus ou moins haute priorité) en fonction d'une proportion déterminée par l'utilisateur.
L'utilisateur peut également influer sur la quantité de flots circulant dans le réseau pendant la simulation grâce à un paramètre α représentant le nombre moyen de flot point à point (autrement dit, par couple). On a alors l'égalité suivante :

$$\alpha n^2 = \lambda \frac{V_{moy}}{d}$$

(avec d débit moyen d'un flot) car les deux « cotés » de cette égalité représente le nombre total de flots vivants moyen théorique dans la simulation. On peut également déduire de cette égalité le nombre de paquet émis par seconde pour un flot, où la taille des paquets est t, on obtient :

$\lambda \frac{V_{moy}}{\alpha n^2 t}$ (L'unité est en paquets par seconde).

Il semble raisonnable d'estimer dans la modélisation que les inters arrivées de paquets soient constantes, on peut néanmoins les espacer eux aussi (comme pour les flots) selon une loi exponentielle (distribution de poisson avec paramètre λ_2).

En outre l'utilisateur peut également définir un certain pourcentage de flots supplémentaire (perturbateurs) ayant les mêmes propriétés que les flots classiques définis plus haut.

---

[1] On a bien $P_{i,j} \leq 1$, il s'agit donc bien d'une proportion de probabilité et la valeur présente à l'indice (i,j) représente la probabilité que le couple (i, j) soit « tiré » pour être la source i et la destination j d'un flot.

[2] Cette matrice ne représente pas les liens directs et les débits qui unissent les nœuds du réseau mais seulement les demandes entre nœuds, déterminées selon des connaissances empiriques ou aléatoirement.

5.2.3. Paramètres de simulation et QoS

Dans cette section nous allons étudier les principaux aspects à modéliser pour pouvoir analyser et comparer différentes politiques de routage, d'évitement de congestion et autres facteurs de qualité de service.
Dans un premier temps un aspect incontournable d'une certaine QoS est la notion de file d'attente. Prenons l'exemple classique d'une queue M/M/s où s désigne le nombre de services, il est alors possible d'utiliser une chaîne de Markov pour sa modélisation [4], cependant le schéma Figure 27 met en avant la difficulté d'appliquer de telles méthodes sur des files dont les propriétés sont complexes.

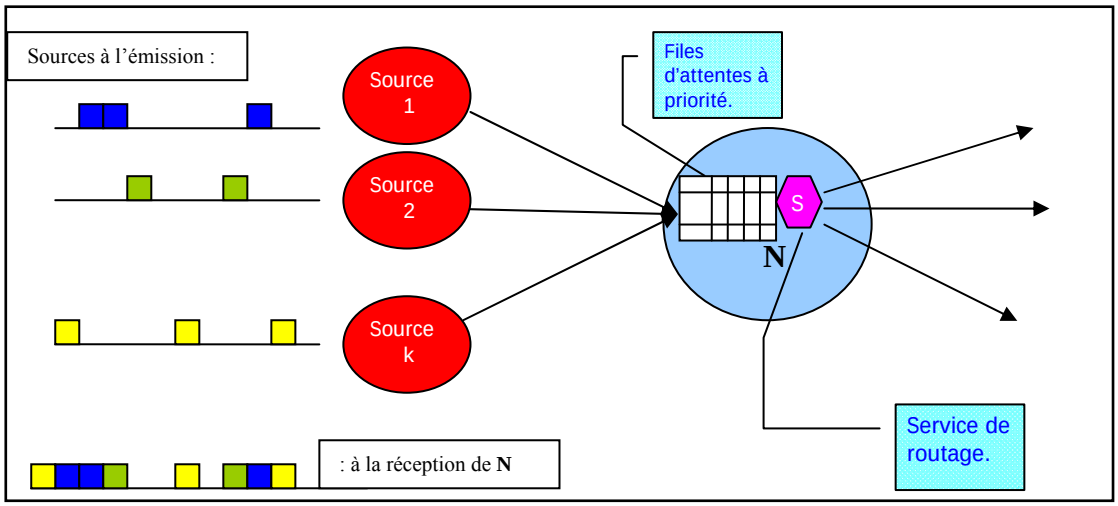


Figure 27 : ***Schéma d'un noeud de routage supportant une politique de qualité de service.***

Dans QoSim, on considère que le temps de service de routage S est négligeable en comparaison avec les temps d'attentes dans les files, on modélise donc des files FIFO « en sortie » comme on pourra le voir la section moteur de simulation et on ignore la durée service de routage (car ce n'est pas le but de ce simulateur, cependant si un objectif d'étude est l'analyse de performances de protocoles de routage, par exemple entre un routage hiérarchique et un routage RIP, la simulation du service de routage est capitale).

Le routage utilisé dans QoSim peut être construit sur :

- Les plus courts chemins (Dijkstra) (plus chemins sous optimaux) :

  - Par saut (RIP…).
  - Par une métrique spécifique (OSPF…).

- Des tables de routage généralisées (voir Figure 28) :

  - Routage proportionnel. (La charge en émission est répartie.)

- Bi routage par bifurcation. (Les paquets sont routés sur des chemins différents)

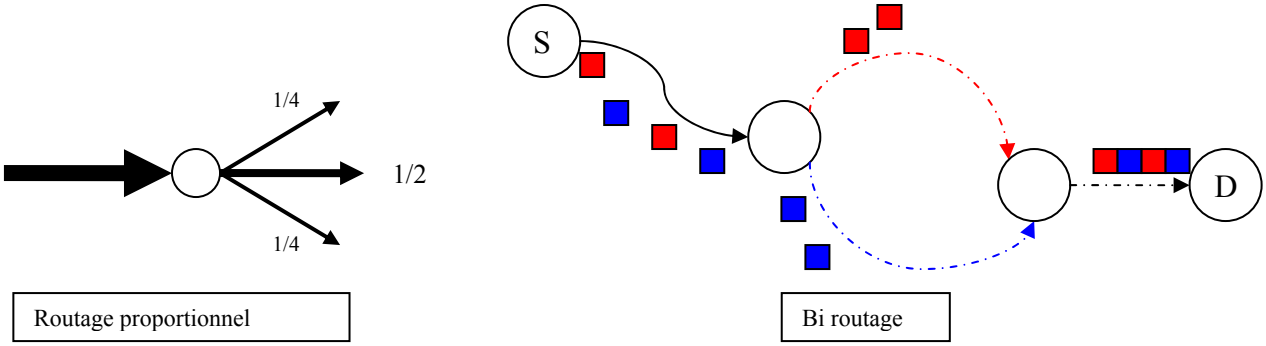


Figure 28 : ***Routage généralisé.***

En revanche, on peut définir des files d'attentes à priorité et à taille variable, de sorte que tous les paquets placés dans une file d'attente prioritaire soient transmis vers le nœud suivant avant les paquets placés dans les files à priorité moindre. La Figure 29, ci-dessous, exprime cette notion de priorité avec dans l'ordre de priorité croissante FIFO1, FIFO2 et FIFO3. (En pratique on définira trois types de services : contrôle, prioritaire, normal.)

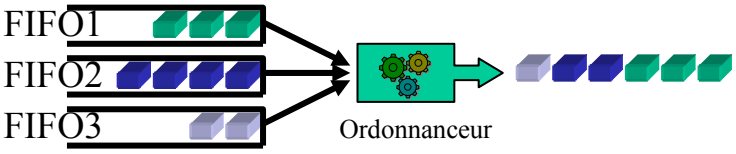


Figure 29 : ***Files d'attente à priorité variable.***

En outre, en cas de débordement de file (en d'autres termes de congestion), le paquet en transit est soit détruit soit dévié vers un meilleur voisin.
Trois autres facteurs de qualité de service sont simulés :

- L'évitement de congestion, avec TCP simulé de manière centralisé, de telle sorte qu'une perte de paquet entraîne une réduction du débit, pour finalement revenir linéairement au débit initial.
- Priorités variables pour les flots best-effort (flots prioritaires).
- Pseudo réservation, c'est-à-dire la réservation de ressources sans états.

On peut donc constater vu la complexité de tels mécanismes que la théorie des files d'attentes avec des relations comme celle de Little [46]:
$L=\lambda W$ avec L : nombre moyen de paquet dans la file d'attente, W : temps moyen d'attente dans le système (service inclus) et λ paramètre constant d'une loi exponentielle (voir 3.2) ou encore l'utilisation de diagrammes de transition fondés sur des processus de naissance et de mort (base des chaînes de Markov) n'est pas applicable directement sur des ensembles de files d'attentes gérants des priorités.

5.2.4. Moteur de simulation

Nous allons ici définir les états fondamentaux construits autour de l'unité de base, le paquet IP (datagramme IP). Ils sont au nombre de quatre :

- état 0 : départ d'un paquet de la source-il s'agit en fait de la « source départ » du flot auquel le paquet appartient (tout les paquets appartenant au même flot ont la même source).
- état 1: arrivée d'un paquet dans un nœud intermédiaire.
- état 2: placement d'un paquet dans une file d'attente.
- état 3 : arrivée au nœud destination.

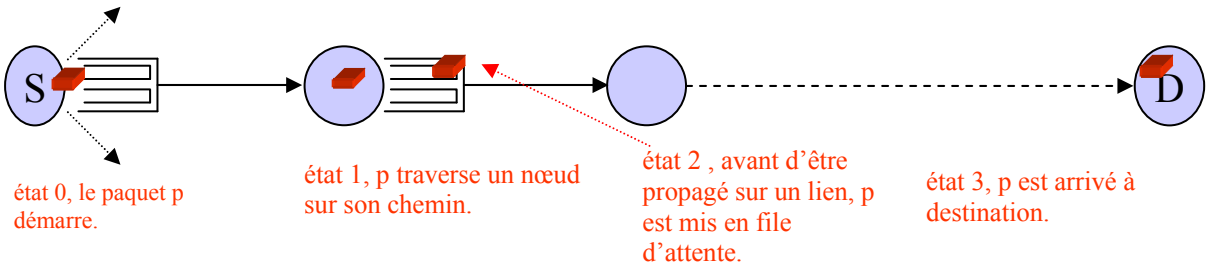


Figure 30 : ***les quatre états fondamentaux.***

Le parcours classique d'un paquet ou datagramme prend donc la forme suivante :
$0 \rightarrow 2 \rightarrow 1 \rightarrow 2 \rightarrow 1 \ldots\ldots\ldots\ldots\ldots \rightarrow 1 \rightarrow 2 \rightarrow 3$, bien qu'il puisse prendre cette forme :
$0 \rightarrow 2 \rightarrow 1 \rightarrow 3$ si la source et la destination sont reliées par un lien direct.

D'un état à l'autre il faut déterminer les événements générés par les changements d'état et les calculs qui en découlent, on peut résumer ces mises à jour de la manière suivante :

- Etat 0 → Etat 2 :
  - Initiation de la pseudo réservation si elle a lieu d'être, génération du datagramme suivant dans le flot courant.
  - Analyse de la table de routage, gestion des débordements, mise à jour de l'ordre de sortie des files multiples selon les priorités définies.

- Etat 2 → Etat 1 :
  - calcul du temps de propagation (selon le débit du lien emprunté et la taille du paquet).

- Etat 1 → Etat 2 :
  - Analyse de la table de routage, gestion des débordements, mise à jour de l'ordre de sortie des files multiples selon les priorités définies.

- Etat 1 → Etat 3 :
  - Calcul du temps de propagation.
  - Libération des ressources pseudo réservées s'il y a eu réservation au préalable.
  - *Mise à jour des statistiques datagramme.*

Par exemple, si on utilise la représentation décrite dans 3.1.2.2, les TGOE, le passage d'un paquet de taille p de l'état 2 à l'état 1 (avec un lien de débit d, entre le nœud d'émission et le nœud de réception du paquet) est formalisé de la manière suivante (Figure 31) :

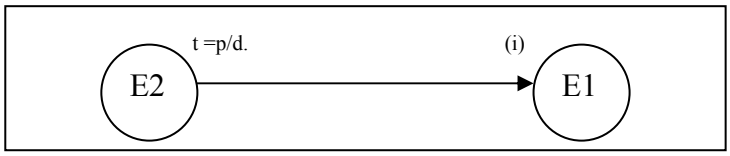


Figure 31 : ***Exemple d'application des TGOE.***

Avec (i) : le paquet est sorti de la file d'attente.

D'un point de vue événementiel, on parlera d'échéancier d'événements (en pratique il s'agit d'une liste chaînée) pour définir toute la suite d'actions dans le temps, la Figure 32 définit la dynamique qui anime l'échéancier. Le premier événement de la liste est traité, créant un nouvel événement à classer dans l'échéancier selon l'estampille de temps qui lui à été attribué dans les calculs correspondant au traitement spécifique vu dans les transitions d'état.

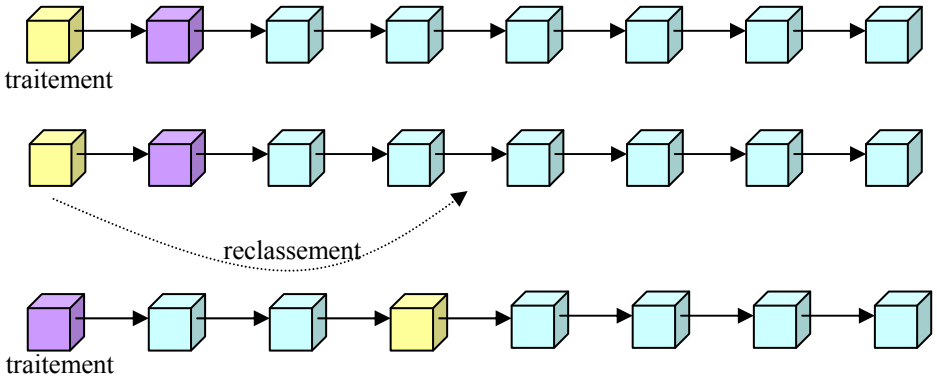


Figure 32 : ***L'échéancier est une liste chaînée d'événements.***

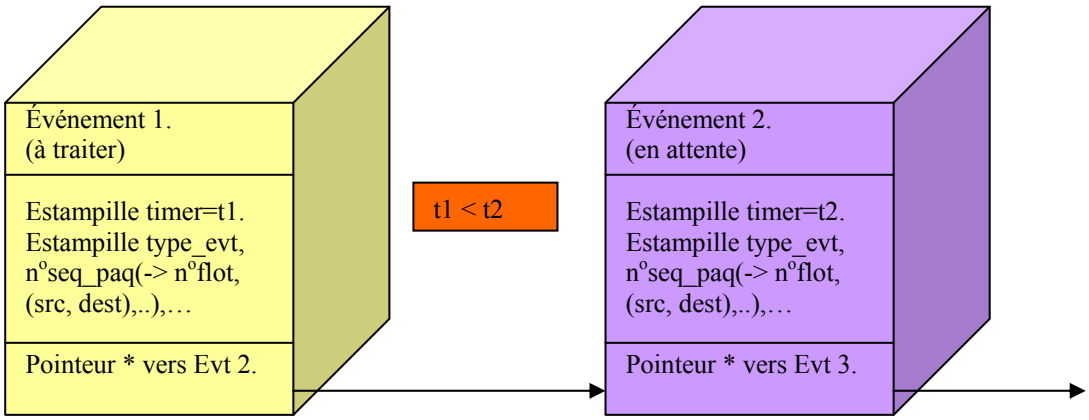


Figure 33 : ***Détail d'un événement.***

La Figure 32 illustre l'ordonnancement des événements au cours du temps, c'est-à-dire le chaînage événementiel constituant l'échéancier, alors que la Figure 33 détaille

grossièrement les principaux attributs d'un événement : son type (0, 1, 2 ou 3), le numéro de séquence du paquet, ainsi que le flot auquel il appartient et toutes les informations qui lui correspondent (sa priorité, sa source, sa destination…). Les événements consécutifs dans l'échéancier sont ordonnancés en fonction de leur estampille de temps.

D'un point de vue implémentation l'algorithme de simulation prend souvent la forme suivante qui se résume à une boucle:

Initialisations des variables d'état
calcul des premiers événements
tant que ((N < Nmax) ou (T < Tmax)) et (il existe un événement dans la liste)
faire dépiler le premier événement et le traiter
mettre à jour les variables d'état
calculer les prochains événements, conséquences de l'événement courant,
et les ranger dans l'échéancier
collecter les statistiques
(si nécessaire) tracer variables et événements
fait
calculer et éditer les résultats finaux
si les résultats ne sont pas satisfaisants au regard des critères utilisateur, on reprend la simulation

Avec Nmax : nombre maximum de données à recueillir (pour des raisons de quantité d'informations à traiter par exemple).
et Tmax : temps maximum de simulation(pour des raisons de coût par exemple).

Les étapes en italique constituent la base de donnée utilisée dans la partie suivante, c'est-à-dire la collecte des mesures de moyenne, de variance et autres indicateurs statistiques en sortie précieux pour déterminer la durée d'une simulation. Cette collecte d'observations se fait dans l'état 3, état dans lequel le paquet arrive à son noeud destination, et où l'on peut mesurer sa durée de vie dans le réseau.

QoSim propose à l'utilisateur, dans sa version de base, un temps T à définir pour déterminer la durée de simulation.(T étant un temps de simulation et non pas un temps « réel »). Dans la partie qui suit nous développerons une procédure qui détermine la durée de simulation par rapport à la quantité d'informations à recueillir au vu d'un certain nombre de critères de précision à fixer par l'utilisateur. En outre, il faut utiliser des techniques spécifiques, comme celles présentées en 4.2.2.2, car les mesures recueillies ne sont pas IID. En effet, si l'on prend l'exemple des temps d'acheminement des paquets, on remarque que le remplissage des files d'attentes contribue à lier les variables comme l'illustre le schéma ci-dessous.

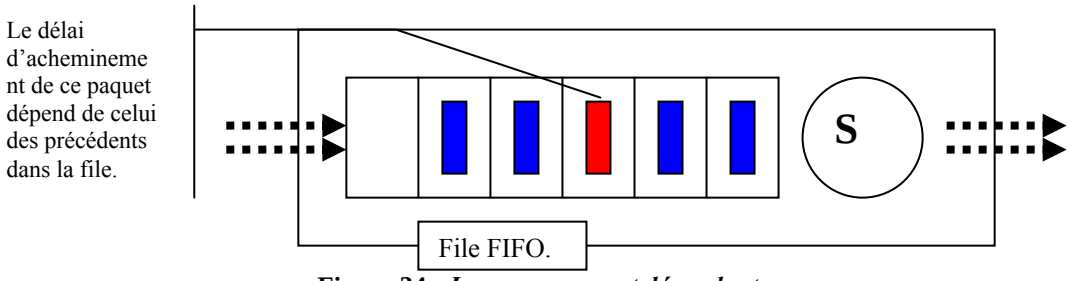


Figure 34 : ***Les mesures sont dépendantes.***

5.3. TRAITEMENT DES DONNEES EN SORTIE

Dans ce sous chapitre, nous allons nous intéresser à la manière dont on peut interpréter les mesures en sortie effectuées sur des objectifs d'étude. Pour cela, il faut bien entendu déterminer les variables et les mesures d'intérêt pour le système étudié, en l'occurrence un réseau à commutation de paquets.

QoSim est un simulateur dont le but fondamental est la mesure de qualité de service apportée par différentes politiques de routage [35] (pseudo réservation, évitement de congestion, files d'attente à priorité variable,…), c'est pourquoi les délais moyens d'acheminement des paquets constituent une réelle mesure en terme de qualité de routage, dans la mesure où ils permettent de juger de la vitesse à laquelle leur routage est traité. Dans la suite de ce chapitre nous allons nous consacrer à cette variable d'intérêt subdivisible en multiples sous mesures, en fonction de la longueur du chemin (nombre de routeurs à file d'attente rencontrées selon le couple source, destination), et du niveau de priorité du flot auquel le paquet appartient.

Cependant, de nombreuses autres mesures peuvent se révéler intéressantes comme le nombre de paquets perdus sur un nœud lors d'une congestion (il ne s'agit à priori pas d'une moyenne, sauf si on établi cette mesure sur l'ensemble des noeuds) ou le nombre moyen de clients par nœud.

Dans un premier temps, nous allons également nous intéresser au volume total de flots vivants dans le réseau à un instant donné pour le calcul de la période de chauffe (warm up) du système, et dans une seconde partie nous allons définir une procédure de calcul déterminant la durée de simulation nécessaire pour obtenir des résultats conformes aux critères utilisateur.

5.3.1. Suppression du warm-up

Le but de cette sous-section est de déterminer un index l, moment auquel le processus étudié entre en période stationnaire. La variable représentant le nombre de flots vivants au cours de la simulation doit être distribué selon une répartition de poisson de paramètre λ au vu de la génération de trafic de QoSim. (Voir 5.2) si bien que cette variable (dont la mesure ne nous intéresse pas en tant que telle, car sa valeur théorique est fixée par l'utilisateur par le biais de α, voir 5.2) doit vérifier des priorités de stationnarité propres à la distribution de poisson.

Nous allons donc nous baser sur cette mesure pour déterminer le nombre d'observations l à supprimer sur tous les paramètres d'intérêt (les délais moyens d'acheminement en l'occurrence voir Figure 35).

La période de chauffe du système correspond à un fort taux de naissance de flots (voir Figure 34-d) et donc à une grande variance en terme de volume de flot présent sur une période donnée (de même pour la « décharge » du réseau), alors que la période stationnaire correspond à des oscillations relativement faibles autour d'une moyenne théorique αN^2 si N représente le nombre de sources/destinations.

Par conséquent, on peut estimer que sur des intervalles suffisamments grands (par exemple, on prendra 30 observations) :

- Si la variance divisée par la moyenne relative à l'intervalle est faible (→ utilisation d'un seuil), alors on est entré en période stationnaire.
- Sinon on est toujours en phase initiale.

Les figures 34(a et d) ci-dessous illustrent le comportement de la variable « nombre de flots vivants » au cours du temps selon le trafic utilisé [(a, b, c) ou (d, e, f)]. De plus les figures 34(b, c, e et f), à la même échelle en abscisse, c'est-à-dire le numéro de séquence absolu (tous flots confondus) du paquet traité, illustrent « la relation logique » entre flots et paquets.

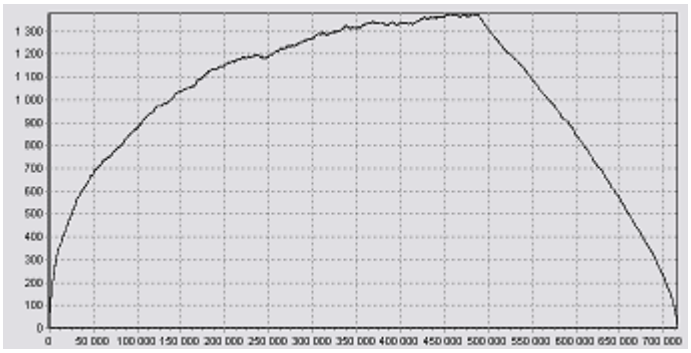


Figure a : nombre de flots vivants selon le numéro de paquet.

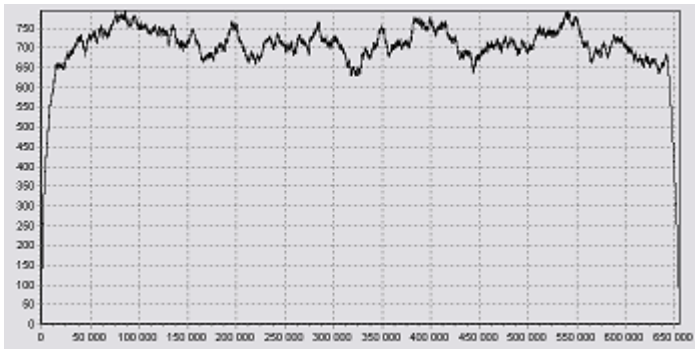


Figure d : nombre de flots vivants selon le numéro de paquet.

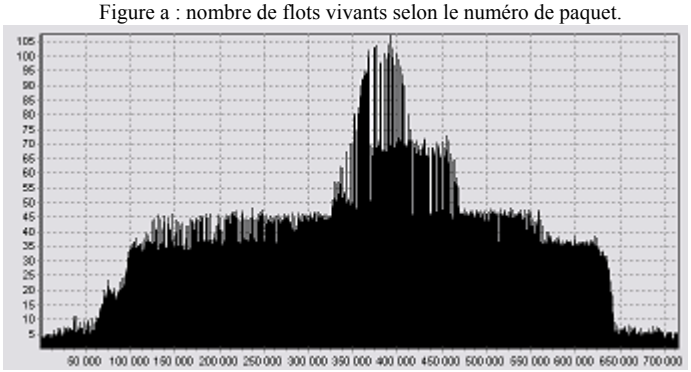


Figure b: délai d'acheminement ponctuel.

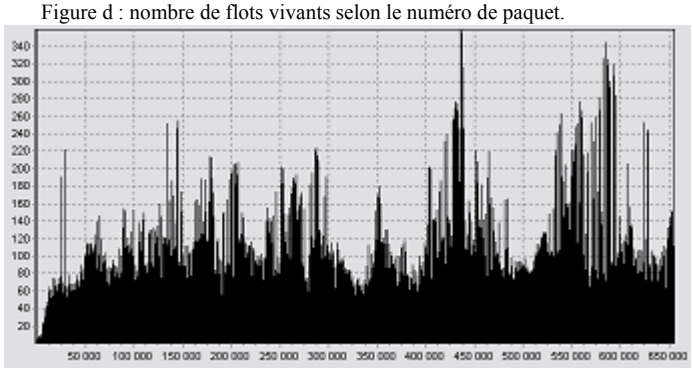


Figure e : délai d'acheminement ponctuel.

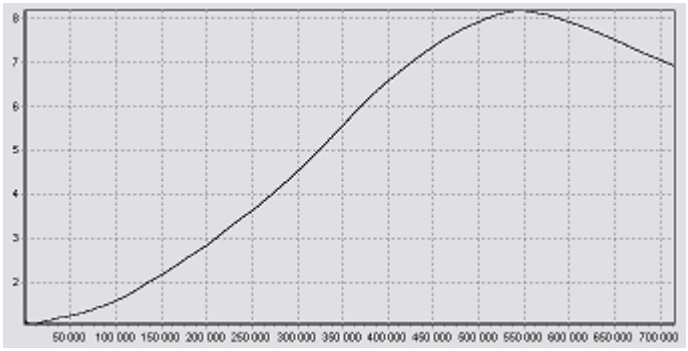


Figure c : moyenne du délai d'acheminement.

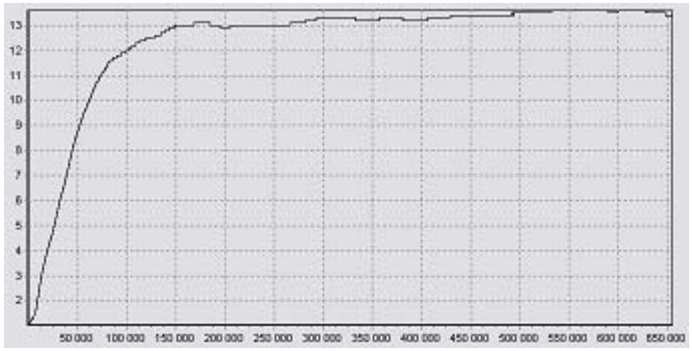


Figure f : moyenne du délai d'acheminement

Figure 35 ***: Nombre de flots et délais d'acheminement (en ms) en fonction de l'estampille de séquence du paquet.***

Les figures a, b et c correspondent à une simulation sur QoSim avec 26 nœuds et $\alpha=2$($N=26$, $\alpha=2$) et avec des flux de gros volumes alors que les figures d, e et f sont issues d'une simulation avec $\alpha=1$ sur approximativement la même « durée » en terme de paquets traités avec des volumes de flots nettement plus petits. (Voir 5.1)

Au vu des figures a et c, on peut déduire que la passe de simulation produisant les résultats a, b et c est trop courte pour ce volume de flot élevé car le nombre de flots vivants n'a pas le temps de se stabiliser autour de sa moyenne (a) et de plus la moyenne du délai d'acheminement n'a pas le temps de se stabiliser(c) et ne semble pas représentative de la

moyenne réelle (b). On peut considérer que presque tout les résultats de cette simulation sont obtenus dans un état transitoire, donc inexploitables.
En revanche, les figures d et f semblent présenter un état stationnaire, on peut donc considérer que les calculs de moyenne de délais d'acheminement peuvent commencer dès que le nombre de flots vivants entre en état stationnaire. (Approximativement après 50000 paquets traités).
De manière générale, la figure 45 suggère que le calcul de l'index l, fin de la période de chauffe, peut se faire sur la variable du volume de flots présent dans le réseau car c'est elle qui détermine le comportement des autres variables.

La Figure 36 correspond à une analyse sous forme de spectres de variances sur des intervalles de taille 30, (α=5, N=26), on constate que la variance est beaucoup plus importante sur les zones d'initialisation et de terminaison que durant la période stationnaire.

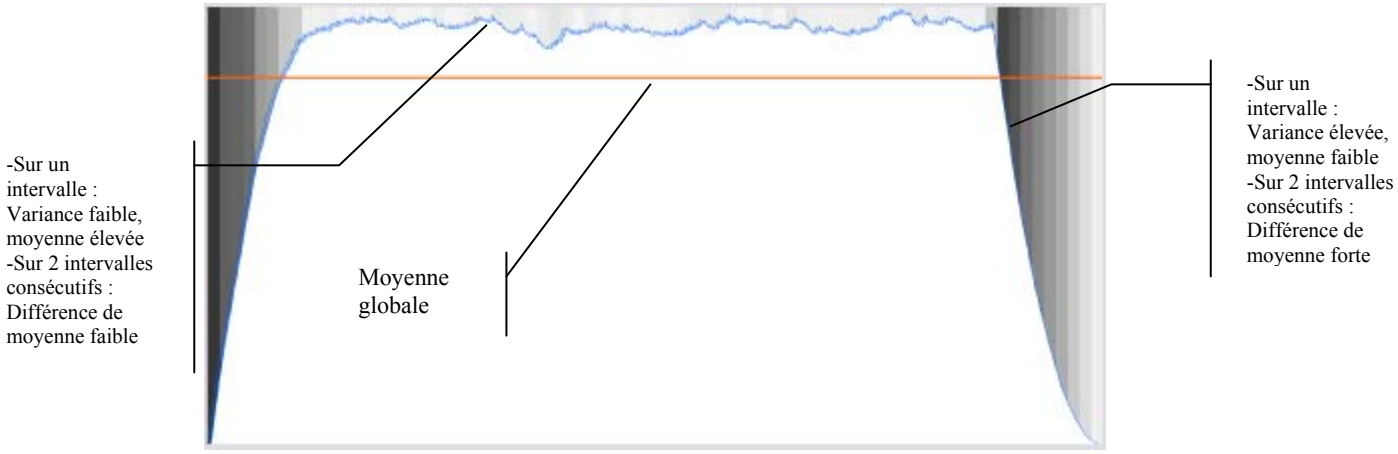


Figure 36 : ***Analyse du comportement d'une période stationnaire sous forme de spectres de variance.***

(Les nuances de gris désignent les différences de variance : noir pour une variance élevée et inversement…)

On peut à présent définir le ratio r i.e. $r = \frac{S(n)}{X(n)}$, avec n représentant la taille de l'intervalle choisi. Ce ratio va nous permettre d'évaluer l'index auquel on considère que la simulation entre en phase stationnaire.[1]

En cours de simulation dès que $r < s$, avec s seuil de précision à déterminer par l'utilisateur, on peut alors collecter les statistiques obtenues sur les variables en sortie comme le délai d'acheminement des paquets. (On prendra en pratique n=30 et s=0.05).
Les figures 46,47 et 48 montrent le résultat d'une telle technique sur le calcul de la moyenne du nombre de flots vivants, on peut constater que la nouvelle moyenne calculée (privée donc des observations appartenants aux intervalles ne respectant pas le seuil s) est visiblement plus caractéristique du phénomène stationnaire (et très proche de αN^2).

[1] On peut effectivement, en utilisant un seuil bien choisi, éliminer les zones de prises de mesures, apparaissant en foncées sur la Figure 36, caractéristiques de la période transitoire, où les flots inonde le réseau.

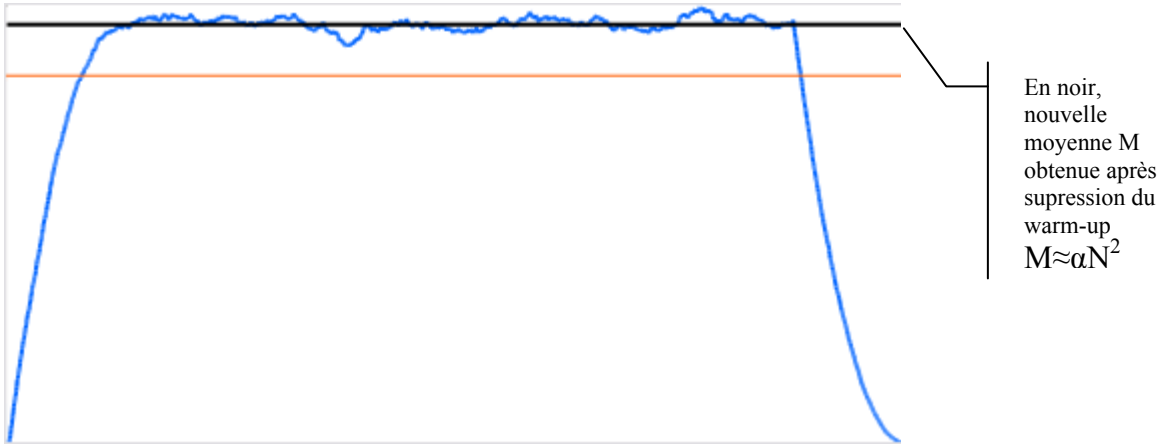


Figure 37 : ***Moyenne global et moyenne recalculée pour N=26, α=5, n=30, s=0.05.***

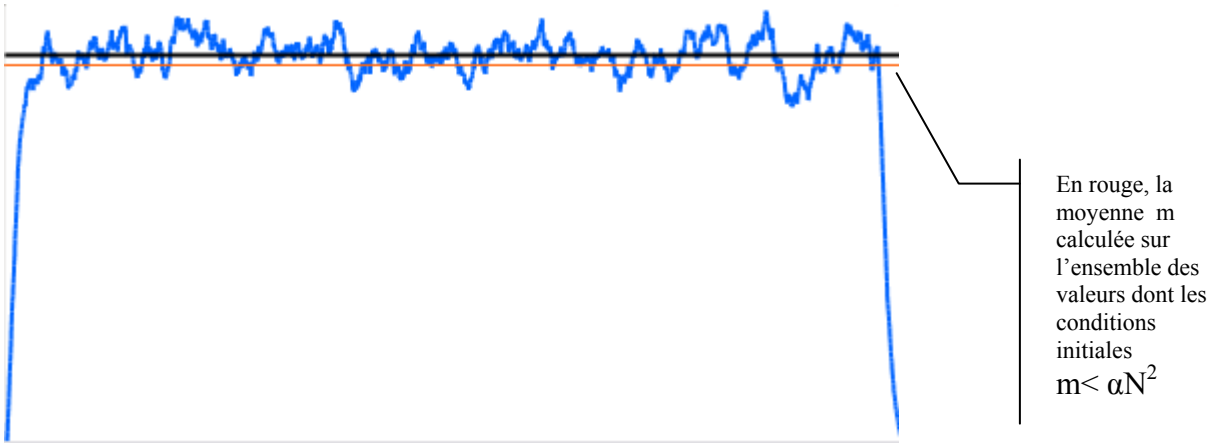


Figure 38 : ***Moyenne global et moyenne recalculée pour N=26, α=1, n=30, s=0.05.***

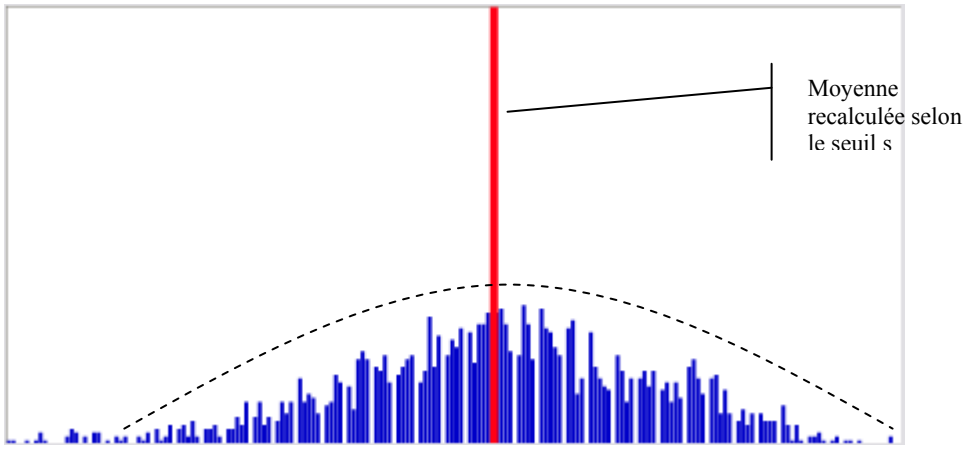


Figure 39 : ***Distribution du nombre de flots vivants autour de la moyenne recalculée pour N=26, α=1, n=30, s=0.05.***

La Figure 39 indique que la distribution des mesures après suppression du warm up semble plus conforme à la génération de trafic voulue en entrée (poissonienne) que celle obtenue avec les conditions initiales qui contient nettement plus de résidus sur les petites

valeurs du nombre de flots vivants entraînant ainsi une détérioration du calcul de la moyenne théorique (cf. Figure 37 et Figure 38).

Cette méthode a l'avantage, dans la mesure où elle permet de déclencher automatiquement le départ de la procédure décrite dans 5.2.2, de s'inscrire dans une procédure séquentielle. Cependant, il faut déterminer le seuil s caractéristique d'une distribution de poisson avec $\lambda = \alpha N^2 \times \frac{d}{V_{moy}}$ (comme décrit en 5.1.2) pour une taille m d'intervalle suffisamment grande, d'autant que la variance est approximée selon l'estimateur classique S(m) alors que les conditions IID ne sont pas réunies. En effet, bien que les intervalles de temps générés par un processus de poisson (3.1.1) soient indépendants (et de moyenne 1/λ), le nombre de flots vivants évolue en fonction de ses mesures antérieures (+1 ou -1 d'une mesure à l'autre, donc dépendant sur de petits intervalles).

Pourtant en pratique, les paramètres génériques m=30 et s=0.05 (ou s=0.01 si α grand) produisent des résultats satisfaisants si les mesures sur le nombre de flots vivants sont espacés dans le temps (soit par une congruence sur le nombre d'éléments recueillis, soit par un laps de temps fixe) comme sur les résultats présentés dans les figures 36, 37 et 38.

Il est à noter que le recalcule de la moyenne n'est pas intéressant en soi (on connaît déjà la moyenne théorique αN^2), mais permet de définir l'indice l auquel la variable du nombre de flots vivants semble entrer en phase stationnaire (Et on considère que toutes les autres variables y entrent également). Une autre méthode basée sur le test de student,

$$\frac{\left|\overline{Y_i(m)} - \overline{Y_{i+1}(m)}\right|}{\sqrt{(S_{i+1}(m) + S_{i'}(m))/m}} \leq z_{\alpha/2}$$

[6], entre intervalles consécutifs pourrait, au vu des remarques sur la Figure 36, déterminer cet index l sans utiliser la moyenne théorique, cependant la différence de moyenne entre les lots formés par le processus semble être significative sur la quasi totalité de la phase stationnaire (et heureusement, toujours sur le warm up) d'après les résultats obtenus.

5.3.2. Calcul de l'intervalle de confiance

Cette section va décrire la procédure séquentielle mise en place dans QoSim sur les observations sélectionnées dès lors que le test (r<s) sur le nombre de flots vivants est réussi selon les paramètres décrits dans la section précédente.

Il s'agit d'une procédure à υ variables ($\upsilon \leq N \times N \times 2$ avec N : nombre de sources/destinations) car nous allons étudier le comportement de chacun des flux point à point (26^2 sur le réseau étudié bien qu'en réalité il suffit d'utiliser le nombre de valeurs non nulles dans la matrice de trafic) en fonction de leurs services (2 au maximum, car les paquets de contrôle sont ignorés dans le calcul d'intervalles de confiance I).

Plutôt que les méthodes décrites dans 4.4.1, et étant donné le nombre de variables d'intérêt, on préférera utiliser deux seuils s1 et s2 à fixer par l'utilisateur selon ses besoins en précision relative sur les intervalles de confiance :

- Le premier sur le service prioritaire : tel que si $\frac{|I_{\alpha 1}|/2}{\overline{Y(n)}} \leq s1$ alors la précision souhaitée est atteinte sur un paramètre de simulation du service prioritaire.
- Le second sur le service normal : tel que si $\frac{|I_{\alpha 2}|/2}{\overline{Y(n)}} \leq s2$ alors la demi largeur, relative à la moyenne, de l'intervalle de confiance est suffisamment étroite au vu des critères utilisateur sur un paramètre du service normal.

Le calcul de l'intervalle de confiance I_α se fait selon la technique ML (d'où $\overline{Y(n)}$ comme notation pour la moyenne) sur des lots considérés comme indépendant au vu du ratio de Von Neumann (voir 4.2.2.2) avec α comme niveau de confiance.

Ce double seuil est utilisé dans un souci de satisfaire au mieux les besoins en précision de l'utilisateur de la simulation. On peut effectivement constater que la variance des délais d'acheminement du service normal est bien supérieure à celle du service prioritaire, ainsi les intervalles de confiance calculés sur le service prioritaire sont plus rapidement étroits que pour le service normal (Voir Figure 44 et Figure 45).

En pratique, on prendra comme niveau de confiance 95 % pour les deux services, $\alpha 1 = \alpha 2 = 0.05$, et pour les seuils relatifs on prendra s1=0.01 et s2=0.05.
Pour déterminer la durée de la simulation, on estime que tous les intervalles de confiance des délais d'acheminement de chaque service sur l'ensemble des couples (source, destination) doivent vérifier leur test respectif (s1 ou s2 selon le service). En d'autres termes, si on note $I_{1,j}$ et $I_{2,j}$ pour $1 \leq j \leq N^2$ les intervalles de confiance du service 1 et 2 sur l'ensemble des couples(sources, destination) :
Si $\max(I_{1,j}) \leq s1$ et $\max(I_{2,j}) \leq s2$ alors la simulation est finie, sinon on applique la procédure de la Figure 40 sur chacun des $\upsilon \leq N \times N \times 2$ paramètres, jusqu'à ce que soit le cas.

Pour éviter le problème posé par la recherche multi variables, il faudrait utiliser une métrique (nombre de bond, débit des liens,…) commune pour l'ensemble des délais d'acheminement, ce qui parait extrêmement difficile dans la mesure où les chemins empruntés sont plus ou moins long (en terme bond ou en sur une autre métrique, type OSPF…), les files d'attente plus ou moins chargées …cependant cela permettrait d'analyser seulement deux moyennes (selon le service) sur l'ensemble du réseau.
Ici, pour contourner le problème d'une métrique commune, on étudie uniquement les délais sur chacun des flux point à point, et d'autre part on normalise les paquets en fonction de leurs volumes respectifs en octet afin d'obtenir des moyennes les plus représentatives possible.

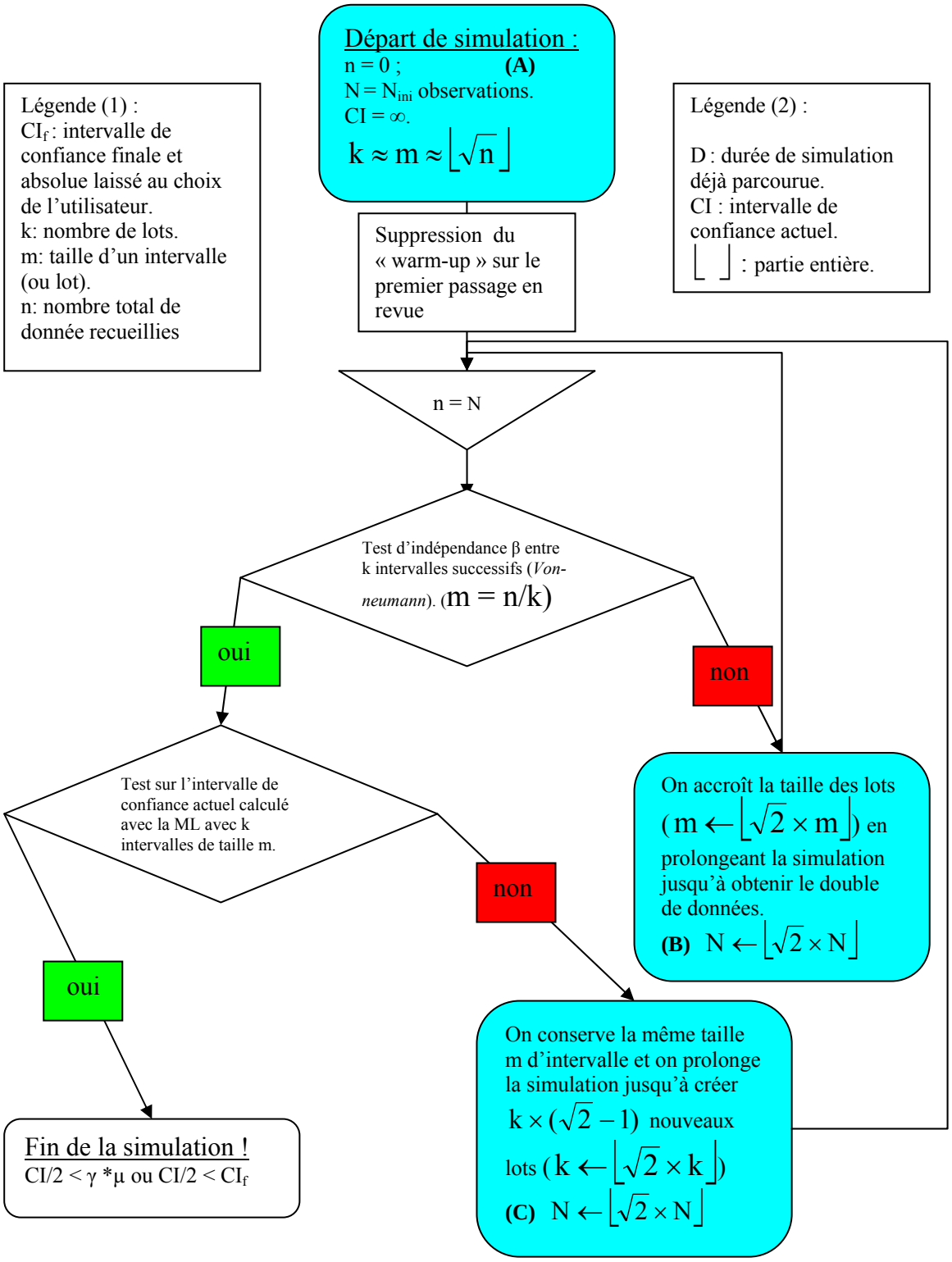


Figure 40 : ***Schéma descriptif de la technique mise en place pour déterminer la durée d'une simulation.***

Données à fournir par l'utilisateur :

- γ ou CI_f pour les critères précision.
- β pour la robustesse du test d'indépendance.

Ce schéma décrit de manière simplifiée la marche à suivre pour obtenir l'intervalle de confiance désirée (sa demi largeur doit être inférieure à $\gamma * \mu$ avec $0 < \gamma < 1$) sur un paramètre de simulation. L'intervalle de confiance est calculé sur les mesures recueillies selon la méthode de la moyenne des lots et en respectant l'indépendance des lots successifs.

Il faut bien noter qu'on ne recommence pas à chaque test échoué la simulation à « vide » mais qu'on reprend là où le test a échoué, c'est-à-dire que l'on conserve les informations collectées antérieurement.

On peut remarquer que la durée de simulation pourrait être prolongée en terme de temps de simulation, voire même pour des raisons de coût en « temps réel ».
Cependant on préférera faire évoluer le paramètre n par l'intermédiaire de k et m (en fonction respectivement du test d'indépendance et du test sur l'intervalle de confiance, car ils permettent une meilleure appréciation pour les réajustements que la durée de simulation en terme de temps).
Ces deux test successifs représentent en quelque sorte un « Check point » sur la largeur de l'intervalle de confiance formé sur les données recueillies précédemment et la simulation s'arrête seulement si celui-ci est suffisamment étroit au vu des critères définis par l'utilisateur. Ces « Check point » sont espacés de manière strictement croissante au fur et à mesure de la simulation (les tests sont, par bloc de deux, séparés par la récolte d'environ n nouvelles observations).

De plus, le nombre de lots utilisés pour le calcul de la largeur de l'intervalle de confiance est multiplié par deux à chaque échec sur le test de l'intervalle de confiance, afin de réduire la variance inter lots.

La suppression du « warm-up » se fait bien entendu qu'une seule fois, en fonction du nombre de flots vivants selon les méthodes décrites dans 5.3.1, et va permettre d'accélérer l'efficacité de la méthode ML, qui, dans les procédures décrites dans cet état de l'art partie 4, considère que les effets du régime transitoire vont disparaître avec le temps et donc conservent les données recueillies dans les conditions initiales pour le calcul de l'intervalle de confiance ou ne précise pas la technique utilisée pour supprimer le biais due aux conditions initiales aussi bien en terme de moyenne qu'en terme d'intervalle de confiance (hormis la procédure WASSAP, 4.3.2.4).

Description :

(A) Initialisation :

On considère que n doit atteindre N_f observations au minimum pour chaque paramètre X_i étudié, $1 \leq i \leq n$, avant de commencer les calculs d'intervalles de confiance et les tests d'indépendance. (Sans prendre en compte les données supprimées par 5.3.1)
Dès lors, on utilise la SQRT rule sur le premier passage en revue, $k = m = \lfloor \sqrt{N_{ini}} \rfloor$
et on préfèrera utiliser la loi normale plutôt que celle de student. En pratique, on prendra donc N_{ini}= 900 observations (nombre minimum d'observations à recueillir pour démarrer l'analyse).

(B) Recalcul de m :

« La dépendance entre intervalles, calculée selon le ratio de Von Neumann, décroît au fur et à mesure que la taille des lots augmente»* (Voir Figure 42), c'est pourquoi on double la taille des lots tout les 2 tests d'indépendance échoués.

(C) Recalcul de k :

« La largeur de l'intervalle de confiance I décroît au fur et à mesure que le nombre de lots k augmente, ceci dans la mesure où $|I_\alpha / 2| = z_{\alpha/2} \times S_m(n) / \sqrt{k}$ est fonction du facteur $\frac{1}{\sqrt{k}}$ »*. La Figure 43 illustre le comportement de la variable $\rho = S_m(n) / \sqrt{k}$ selon la taille du lot m et donc du paramètre k (car n est fixé sur cette exemple) sur un couple (source, destination) et un service donné.

D'autre part on ne modifie pas la taille des lots m, car on peut estimer que les nouveaux lots formés ($k - \lfloor \sqrt{2} \times k \rfloor$ lots) sont également indépendants entre eux et avec les précédents, au vu de la taille m sélectionnée pour les intervalles, par la succession des tests d'indépendance échoués (B), mais on teste tout de même leurs indépendances à nouveau. (Voir Figure 43). *sous des conditions assez généralement satisfaites au vu des résultats sur l'ensemble des couples (*src, dest*) (Voir 4.)

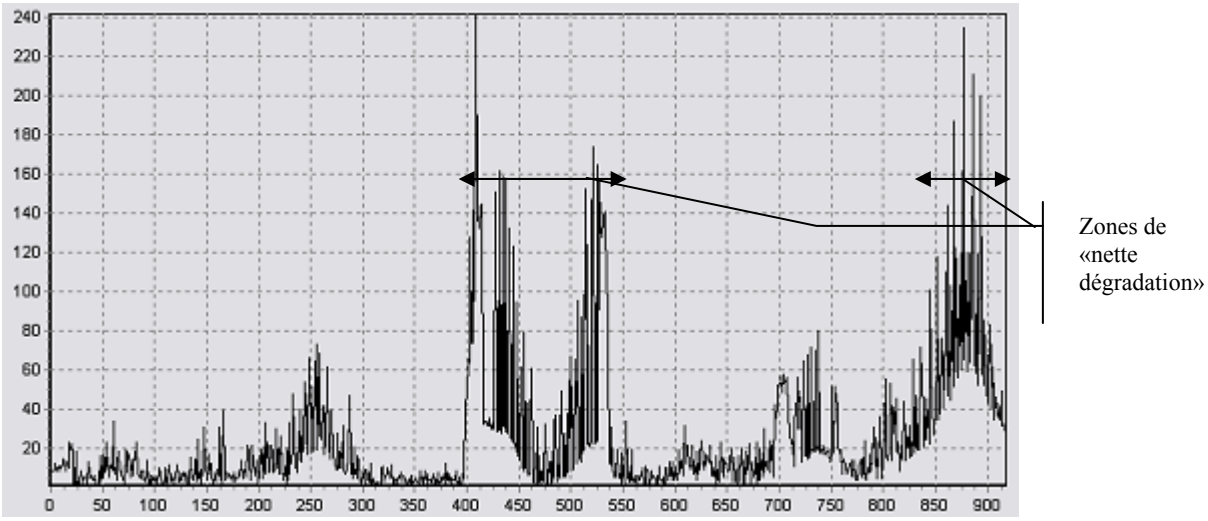


Figure 41 : ***Délais d'acheminement (ms) en fonction du paquet traité (n°) sur l'ensemble des flots du couple (N4, N21).***

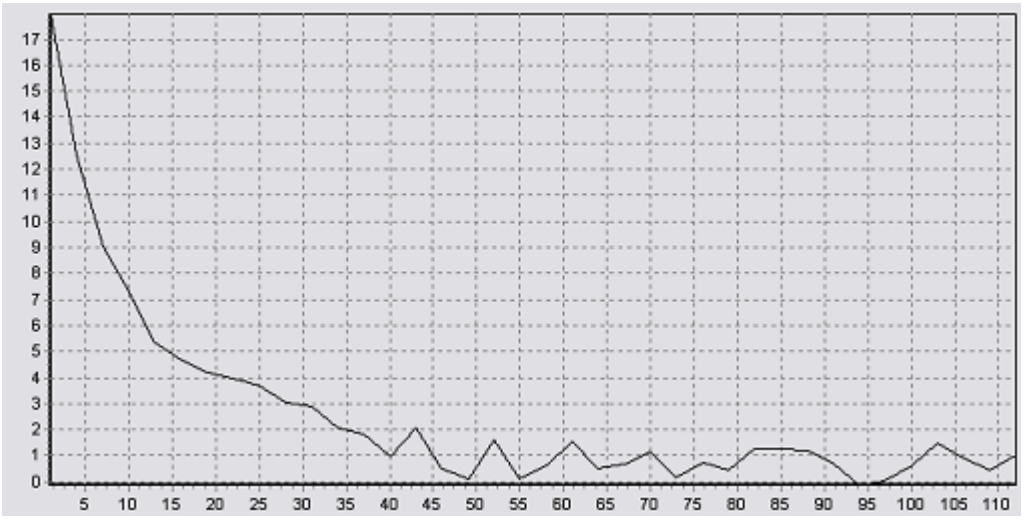


Figure 42 : ***Ratio de Von Neumann en fonction de la taille des lots. (Sur les observations recueillies pour le couple (N4, N21), voir Figure 41).***

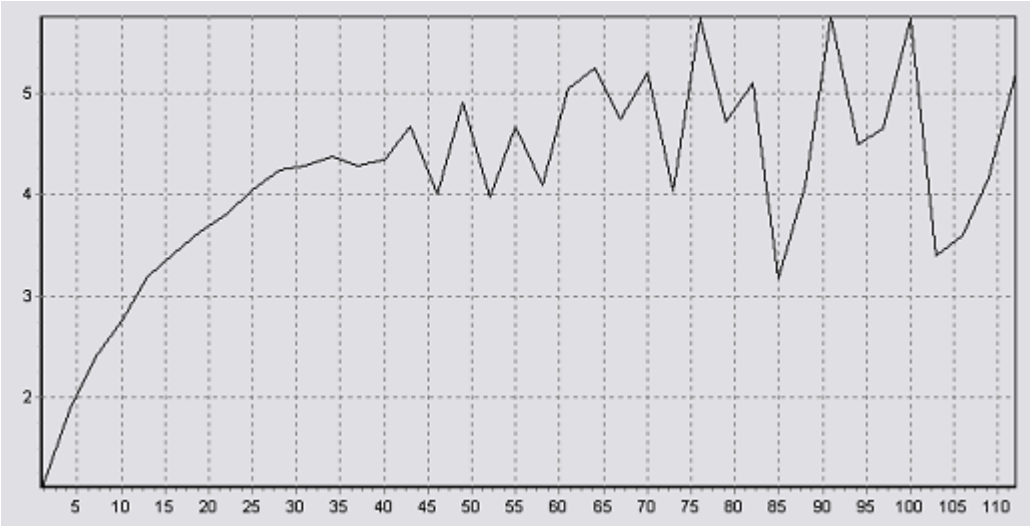


Figure 43 : ***Evolution du rapport p en fonction de la taille des lots. (Sur les observations recueillies pour le couple (N4, N21), voir Figure 41).***

Rappel : $P=\frac{S_m(n)}{\sqrt{k}}$.

Les Figures 41, 42 et 43 sont issues d'une simulation sur QoSim (N=26, α=1). Les mesures sont récoltés sur l'ensemble des paquets appartenants au flux transitant sur le couple (source, destination) = (N4, N21) estampillés au service normal.
On peut constater sur la Figure 41 que les paquets semblent, durant certaines périodes, être nettement ralentis, vraisemblablement en fonction du remplissage des files d'attentes rencontrées[1], étant donné que le chemin N1→ N4 est identique pour tous les délais récoltés.
La Figure 42 confirme que la dépendance inter lot décroît avec l'augmentation de la taille des lots, on constate qu'avec des lots dont le cardinal est supérieure à 35, l'indépendance des lots semble justifiée.
La Figure 43 met en avant le comportement de la variable p qui diminue avec l'augmentation du paramètre k (ou augmente avec l'accroissement du paramètre m, car n est fixé sur cet exemple, n ≈ 900 observations).
De manière générale, ces deux figures ont de réelles tendances générales jusqu'à m= 40, puis leur comportement devient chaotique et donc moins analysable.

Pour déterminer k, on peut aussi utiliser une approximation analogue à 4.3.1 et calculer en anticipant sur la variance le nombre de lot k nécessaires,

$$k \leftarrow \left\lceil \frac{(z_{\alpha/2} \times S_m(n))^2}{\gamma \times \overline{Y(n)}} \right\rceil$$

en utilisant les notations de la ML et γ seuil utilisé dans la procédure décrite sur la Figure 40.

On peut également comparer la différence entre les mesures de délais sur le service *prioritaire* (bénéficiant de privilèges en terme de vitesse de routage) et sur les paquets transitants avec une estampille de service *normal*.
Les Figures 43 et 44 illustrent la différence entre les moyennes des deux services étudiés, les mesures étant prises sur l'ensemble du trafic en transit sur le réseau simulé.

[1] Cela correspond aux zones où le délai d'acheminement forme des pics (voir Figure 41) sur certaines séquences de mesures.

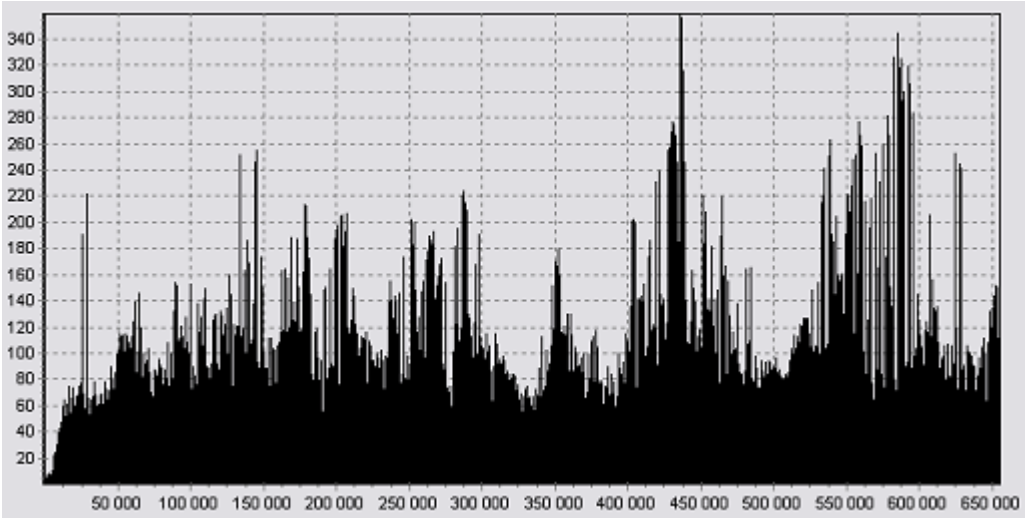


Figure 44 : ***Délais d'acheminent (ms) sur l'ensemble des paquets traités dans le réseau pour le service normal.***

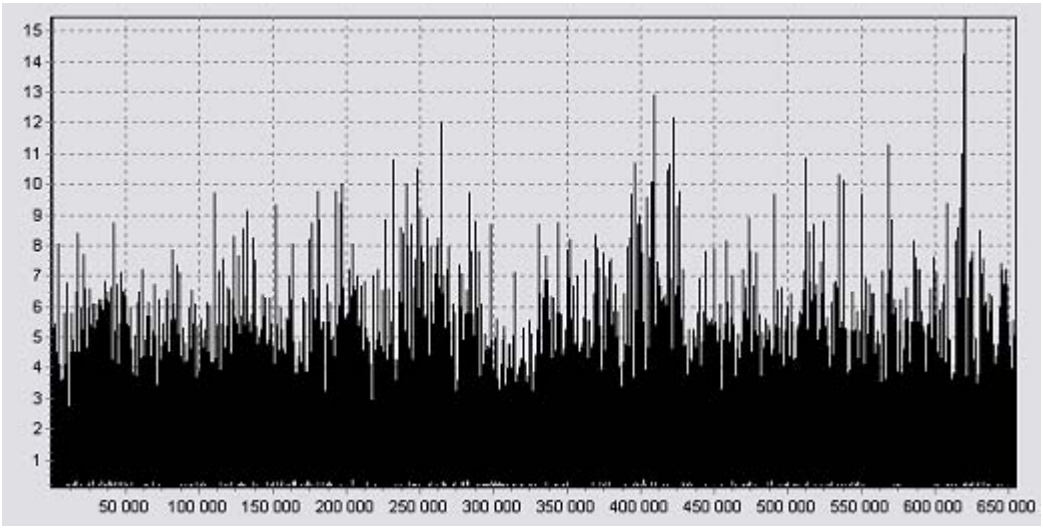


Figure 45 : ***Délais d'acheminent (ms) sur l'ensemble des paquets traités dans le réseau pour le service prioritaire.***

Les flots prioritaires représentent 20 % du volume du trafic généré sur cette simulation QoSim ($N=26$, $\alpha=1$), on peut constater avec satisfaction que le service prioritaire est de manière générale bien plus rapidement traité (≈ 4ms) que le service normal (≈ 100ms).
(Ce qui explique le fait d'utiliser une analyse statistique appropriée à leurs comportements respectifs).

Le schéma Figure 40 s'inscrit donc dans une procédure multi variables, qui s'exprime ainsi : tant que l'ensemble des intervalles de confiance de chacun des services ne respecte pas les critères utilisateur [s1, s2] (pour l'intervalle de confiance relatif) et [β] (pour le calcul de l'indépendance), on rallonge la durée de simulation afin de produire autant d'observations que nécessaire sur chacun des paramètres de simulation où l'intervalle de confiance calculé avec la ML n'est pas suffisamment étroit (ou si le test d'indépendance entre lots a échoué). Par conséquent le nombre de mesures sur les paramètres ayant déjà satisfait les deux tests va lui aussi augmenter (si on arrêtait la charge émise par le couple (*src*, *dest*) concerné le reste du trafic ne serait plus représentatif), et on en profite alors pour recalculer le nouveau CI produit avec la même taille de lot m. (Qui sera plus étroit encore que le besoin en précision de l'utilisateur).

Conclusion et perspectives

Cet état de l'art définit les principales méthodes et techniques décrivant le développement et le déploiement d'une simulation tant au niveau méthodologie de modélisation que pour l'interprétation des résultats en sortie. Plus particulièrement, nous nous sommes intéressé aux modèles de réseaux de communication grande échelle qui constituent un exemple idéal pour représenter la complexité de telles techniques.
En effet, la modélisation du trafic de paquets sur des réseaux de type IP est un domaine de recherche encore très ouvert notamment grâce aux récentes avancées (le modèle ON/OFF en est un exemple) en la matière.

De plus, l'analyse statistique des mesures en sortie, et ici des délais moyens d'acheminement des paquets selon leur estampille de service, met en avant d'une part la difficulté à interpréter les résultats correctement, d'autre part on peut s'apercevoir que le processus représentant les délais d'acheminement des paquets ne semble pas complètement stationnaire (particulièrement pour le service normal).

Il faudrait pour une meilleure analyse, étudier le comportement de la durée de vie des paquets en subdivisant le processus étudié en lots représentatifs de congestion ou de « vie normale » et ainsi calculer deux moyennes : l'une pour représenter la durée de vie d'un paquet pris dans une congestion et l'autre pour étudier le délai moyen d'acheminement classique hors congestion, ceci démultipliant encore le problème de la recherche multi variables (à moins d'aboutir à une métrique commune pour l'ensemble des paquets ou d' analyser les couples (src, dest) sous formes de lots indépendants…).

La procédure décrite dans ce document présente néanmoins l'avantage de produire les intervalles de confiance valables et étroits (en fonction des besoins de l'utilisateur) en différenciant au mieux les mesures de délais selon la nature des différents paquets.

Par ailleurs, la nature stochastique des données en entrée peut également être sujette à des efforts de recherche, on peut imaginer étudier de tel processus d'arrivée par expérimentation, c'est-à-dire en analysant la demande des utilisateurs d'Internet sur de gros serveurs WEB, par exemple, pour produire de nouveaux modèles de trafic.

Pour conclure à titre personnel, ce travail m'a permis d'une part, à travers mon effort de documentation, de découvrir le monde de la recherche sous un angle enrichissant et captivant, et d'autre part d'appliquer les connaissances ainsi acquises, sur un domaine que j'apprécie particulièrement : les télécommunications, avec une méthodologie complètement nouvelle à mes yeux, la simulation à événements discrets.

REFERENCES

[1] Krzysztof, Hae-Duck Joshua Jeong, and Jong-Suk Ruth Lee, *ON Credibility of Simulation Studies of Telecommunication Networks*, IEEE Communications Magazine, Janvier 2002.

[2] E.H.Page, *Simulation modeling Methodology*, *Principles and Etiology of Decision Support,* , PhD Thesis, Virginia Polytechnic Institute and State University, 1997.

[3] S. Fortin, B. Sericola, *A Markovian Model for the Stationary behavior of TCP*, IRISA publication interne 1410, 2001.

[4] P.Laguesdron, J.Pellaumail, Gerardo Rubino, B.Sericola, Transient nalysis of the M/M/1 queue, IRISA publication interne 720, 1993.

[5] G.Siegel, *PROSIT*, *Un environnement pour la programmation de simulations à événements discrets*, Thèse de Doctorat, Université de Nice-Sophia Antipolis, 1997.

[6] R.F. Woolson, *Statistical Methods for the Analysis of Biomedical Data*, John Wiley and sons, 1987.

[7] Paxson and Floyd, *Wide aera traffic: the failure of Poisson modeling*,IEEE/ACM Transactions on Networking 3,1994.

[8] M.Crovella, *Self similarity in Worl wide web traffic: evidence and possible causes*, IEEE/ACM Transactions on Networking 5,1996.

[9] W.David Kelton, *Statistical analysis of simulation output*, Proceedings of the 1997 Winter Simulation Conference, 1997.

[10] C.Alexopoulos,S-H Kim, *Output data analysis for simulations*, Proceedings of the 2002 Winter Simulation Conference, 2002.

[11] C.W.Zobel ,K.Preston, *Determining a warm up period for a telephone network routing Simulation*, Proceedings of 1999 Winter Simulation Conference, 1999.

[12] C.Hue, J.P. LeCadre, P.Perez,*Performance analysis of two sequential monte carlo methods and posterior crameo-rao bounds for multi-target tracking*, IRISA publications interne 1457, 2002.

[13] K.Pawlikowski, D.C.McNickle, G.Ewing, *Coverage of confidence intervals in sequential steady-state simulation*,EUROSIM Congress Vienne, 1995.

[14] P.Heidelberg, P.D. Welch, *A spectral Method for Confidence Interval Generation and Run Lenght Control in Simulation*,Communications of ACM, 1981.

[15] D.Goldsmann, L.Schruben*, New Confidence Interavl Estimator Using Standardised Time Series*,Management Sciences, 1990.

[16] P Zheng and L. M.Ni, *EMPOWER, A Network Emulator for Wireline and Wireless Networks,* IEEE Infocom, 2003.

[17] K.Fall, *Network emulation in the VINT/ns simulator*, Proceedings of the 4th IEEE Symposium of Computers and Communications, 1999.

[18] N.M.Steiger, J. R.Wilson, *Improved Batching for confidence interval construction in steady-state simulation*, Proceedings of the 1999 Winter Simulation Conference, 1999.

[19] N.M.Steiger, J. R.Wilson, E.K Lada, *A wavelet-based spectral method for steady-state simulation analysis*, Proceedings of the 2003 Winter Simulation Conference, 2003;

[20] P.Mullarkey, S.Gavirneni, *Dynamic output data analysis for simulations of manufacturing environments*, Proceedings of the 2000 Winter Simulation Conference, 2000.

[21] A.M.Law, W.D.Kelton, *Simulations modeling and analysis 2nd edition*, McGrawHill, 1991.

[22] P.L'ecuyer,Good parameters and Implementations for Combined Multiple Recursive Random Number Generators, Ops.Res., 1999.

[23] M.Matsumoto, T. Nishimira,*Mersenne Twister: A 623-Dimensionally Equidistributed Uniform Pseudo-Random Number Generator*, ACM Transactions Modeling and Comparaison Sim.,1998.

[24] B.Gaither, *Empty Empiriscim*, ACM Performance Evaluation Review, 1990.

[25] M.R.Lackner, *Toward a general Simulation Capability*, Proceedings of the AFIPS Sprint Joint Computer Conferance, 1962.

[26] B.P Zeigler, *Theory of Modeling and Simulations*, John Wiley and Sons, 1976.

[27] H.Praehofer, D.Pree,*Visual Modeling of DEVS-based Multiformalism System Based on Higraphs*, Proceedings of the 1993 Winter Simulation Conference, 1993.

[28] L.Schruben, *Simulations Modeling with event Graph*, Communications of the ACM, 1993.

[29] E.Yucesan, L. Schruben, *Simulation Graph Duality : A world view Transforamtions for Simple Queuing problem*, Proceedings of the 1993 Winter Simulation Conference, 1993.

[30] R.E.Nance, *The Conical Methodology and the evolution of Simulation Model devellopment*, Annals of Operations research, 1994.

[31] C.M. Overstreet, R.E.Nance, *Model Specification and Analysis for DES*, PhD Dissertation, Virginia Polytechnic Institute and State University, 1982.

[32] S.S Shapiro, M.B Wilk, *An analysis of variance test for normality*, Biometrika, 1965.

[33] G.Kramer, G.Pesanvento, *On generating Pseudo Pareto Distribution*, Technical Brief, 2001.

[34] V. Neumann, *The mean square difference* , Ann. of Math. Statistics,1941.

[35] S. Cateloin, *Routage et qualité de service dans le réseau Internet*, Thèse de Doctorat de l'université de Technologie de Compiègne, 2001.

[36] G.S.Fischmann, *Discrete event simulation*, Springer Verlag, 2001.

[37] D.H.Ockerman, *Initialization bias tests for stationnary stochastic processes based upon standardized time series techniques*, Ph.D. Thesis, Georgia Institute of Technology, 1995.

[38] P.D. Welch, *The statistical analysis of simulation results*, The Computer Performance Modeling Handbokk, Academic Press, 1983.

[39] L.W.Schruben, *Confidence interval estimation using standardized time series*, Operations Research, 1983.

[40] M.S.Meketon, B.W. Schmeiser, *Overlapping batch means: Something for nothing?*, Proceedings of the 1984 Winter Simulation Conference, 1984.

[41] R.G. Sargent, *Requiments of a Modeling Paradigm*, Proceedings of the 1992 Winter Simulation Conference, 1992.

[42] R.E. Nance, *Model Representation in DES : Prospects for Developing Documentation standards*, Current Issues in Computer Simulation Academic Press, 1979.

[43] A.M. Law, W.D Kelton, *Confidence Intervals for steady-state simulations, I:A survey of fixed sample size procedures*, Operations Research, 1984.

[44] B.W. Schmeiser, *Batch size effects in the analysis of simulation output*, Operations Research, 1982.

[45] G.S. Fishman, L.S. Yarberry, An *implementation of the batch means method*, INFORMS journal on Computing, 1997.

[46] R. Mazumdar, V. Badrinath, F. Guillemin, C. Rosenberg., *A note on the pathwise version of Little's formula*, Operations Research Letters, 1993.