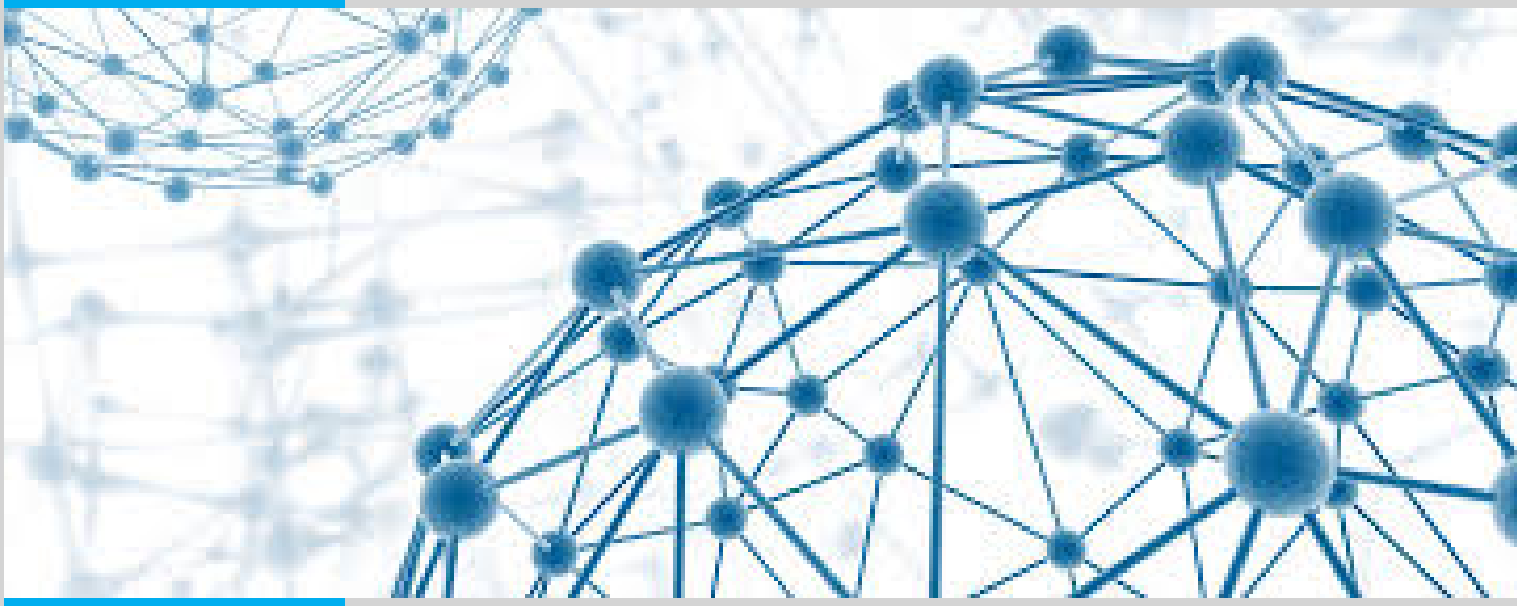
Support de Cours

UNIVERSITE
DE BATNA 2

DEPARTMENT
D'INFORMATIQUE

Performance & Simulation des réseaux.



Dr. Chafik ARAR
chafik.arar@univ-batna2.dz

Table des matières

1 Simulation et simulation des réseaux

1.1 Définitions

Définition 1 : La simulation [1] est l'imitation d'un système du monde réel par une reconstitution computationnelle de son comportement selon les règles décrites dans un modèle mathématique.

Définition 2 : La simulation [2] sert à imiter un système ou un processus réel. L'acte de simuler un système implique généralement d'envisager un nombre limité de caractéristiques et de comportements clés dans le système d'intérêt physique ou abstrait, qui est autrement infiniment complexe et détaillé.

Définition 3 : Une simulation nous permet d'examiner le comportement du système sous différents scénarios, qui sont évalués par ré-promulgation dans un monde virtuel de calcul. La simulation peut être utilisée, entre autres, pour identifier les goulots d'étranglement dans un processus, fournir un banc d'essai sûr et relativement moins coûteux (en termes de coût et de temps) pour évaluer les effets secondaires et optimiser les performances du système, avant de réaliser ces systèmes dans le monde physique.

1.2 Pourquoi la simulation est-elle importante ?

Au début du vingtième siècle, la modélisation et la simulation n'ont joué qu'un rôle mineur dans le processus de conception du système. Ayant peu d'alternatives, les ingénieurs sont passés directement de la conception papier à la production afin qu'ils puissent tester leurs conceptions.

Aujourd'hui, la modélisation et la simulation sont largement utilisées. Ils sont utilisés non seulement pour trouver si une conception d'un système donné fonctionne, mais pour découvrir une conception de système qui fonctionne le mieux. Plus important encore, la modélisation et la simulation sont souvent utilisées comme une technique peu coûteuse pour effectuer des analyses d'exception et de «quoi faire si !», surtout quand le coût serait prohibitif lors de l'utilisation du système réel.

Actuellement , la simulation est utilisée dans de nombreux contextes, y compris la modélisation des systèmes naturels afin de mieux comprendre leur fonctionnement.

Les questions clés de la simulation comprennent :

1. L'acquisition d'informations à partir de sources valides,
2. La sélection des caractéristiques et des comportements clés,
3. L'utilisation d'approximations et d'hypothèses simplificatrices au sein de la simulation.

La simulation permet une expérimentation orientée vers un objectif avec des systèmes dynamiques, c'est-à-dire des systèmes avec un comportement dépendant du temps. elle est devenue une technologie importante, et elle est largement utilisée dans de nombreux domaines de recherche scientifique.

De plus, la modélisation et la simulation sont des étapes essentielles de la conception technique et de la résolution des problèmes et sont entreprises avant la construction d'un prototype physique. Les ingénieurs utilisent des ordinateurs pour dessiner des structures physiques et pour faire des modèles mathématiques qui simulent le fonctionnement d'un dispositif ou d'une technique. Les phases de modélisation et de simulation sont souvent la plus longue partie du processus de conception. Au début de cette phase, les ingénieurs ont plusieurs objectifs à l'esprit :

1. Le produit/problème répond-il à ses spécifications ?
2. Quelles sont les limites du produit/problème ?
3. D'autres conceptions/solutions fonctionnent-elles mieux ?

Les phases de modélisation et de simulation passent généralement par plusieurs itérations, les ingénieurs testant diverses conceptions pour créer le meilleur produit ou la meilleure solution à un problème. La simulation fournit une méthode pour vérifier sa compréhension du monde et aide à produire de meilleurs résultats plus rapidement.

On peut utiliser la simulation lorsque la solution analytique n'existe pas, où elle est trop compliquée ou nécessite plus de temps de calcul que la simulation. La simulation ne doit pas être utilisée dans les cas suivants :

- La simulation nécessite des mois ou des années de temps CPU. Dans ce scénario, il n'est probablement pas possible d'exécuter des simulations.
- La solution analytique existe et elle est simple ! Dans ce scénario, il est plus facile d'utiliser la solution analytique pour résoudre le problème plutôt que d'utiliser la simulation (sauf si l'on veut relâcher certaines hypothèses ou comparer la solution analytique à la simulation).

1.3 Qu'est-ce qu'un modèle ?

Un modèle informatique est un programme informatique qui tente de simuler un modèle abstrait d'un système particulier. Les modèles informatiques peuvent être classés selon plusieurs critères binaires orthogonaux, y compris :

1. État continu ou état discret : Si les variables d'état du système peuvent prendre n'importe quelle valeur, le système est modélisé à l'aide d'un modèle à état continu. D'autre part, un modèle dans lequel les variables d'état ne peuvent prendre que des valeurs discrètes est appelé un modèle à états discrets. Les modèles à états discrets peuvent être des modèles à temps continu ou discret.
2. Modèles à temps continu ou à temps discret : si les variables d'état du système peuvent changer leurs valeurs à tout instant, le système peut être modélisé par un modèle en temps continu. D'autre part, un modèle dans lequel les variables d'état peuvent changer leurs valeurs uniquement à des instants discrets de temps est appelé un modèle à temps discret.
3. Une simulation évènementielle (DES) à temps discret gère les évènements dans le temps. Dans ce type de simulation, le simulateur maintient une file d'attente d'évènements triés par le temps simulé qu'ils devraient se produire. Le simulateur lit la file d'attente et déclenche de nouveaux évènements lorsque chaque évènement est traité. Il n'est pas important d'exécuter la simulation en temps réel. Il est souvent plus important de pouvoir accéder aux données produites par la simulation, de découvrir des défauts logiques dans la conception ou la séquence des évènements. La plupart des simulations d'ordinateur, de test logique et d'arbre de défaillance sont de ce type.
4. Un type particulier de simulation discrète qui ne s'appuie pas sur un modèle avec une équation sous-jacente, mais peut néanmoins être représenté formellement, est la simulation basée sur l'agent. Dans la simulation basée sur l'agent, les entités individuelles (telles que les molécules, les cellules, les arbres ou les consommateurs) du modèle sont représentées directement (plutôt que par leur densité ou concentration) et possèdent un état interne et un ensemble de comportements ou de règles qui déterminent comment L'état de l'agent est mis à jour d'une étape à une autre.
5. Déterministe ou probabiliste : Dans un modèle déterministe, la répétition d'une même entrée donnera toujours la même sortie. D'autre part, dans un modèle probabiliste, la répétition d'une même entrée peut conduire à des sorties différentes. Les modèles probabilistes utilisent des générateurs de nombres aléatoires pour modéliser les évènements fortuits ou aléatoires ; Elles sont aussi appelées simulations de Monte Carlo (MC).
6. Les modèles chaotiques constituent un cas particulier de modèles déterministes à états continus, dans lesquels la sortie est extrêmement sensible aux valeurs d'entrée. Les modèles linéaires ou

non linéaires : Si les sorties du modèle sont linéairement liées aux entrées, le modèle est appelé modèle linéaire. Cependant, si les sorties ne sont pas des fonctions linéaires des entrées, le modèle est appelé un modèle non linéaire.

7. Modèles ouverts ou fermés : Si le modèle possède une ou plusieurs entrées externes, le modèle est appelé modèle ouvert. D'autre part, le modèle est appelé un modèle fermé s'il n'a pas d'entrées externes du tout.
8. Modèles stables ou instables : Si le comportement dynamique du modèle arrive à un état stationnaire avec le temps, alors le modèle est appelé stable. Les modèles qui n'atteignent pas l'état stable avec le temps sont appelés modèles instables. Notez que la stabilité se réfère au temps, tandis que la notion de chaos est liée aux sensibilités comportementales aux paramètres d'entrée.
9. Local ou distribué : si le modèle s'exécute uniquement sur un seul ordinateur, il est appelé un modèle local. Les modèles distribués, d'autre part, fonctionnent sur un réseau d'ordinateurs interconnectés, éventuellement dans une vaste zone, sur Internet. Les simulations dispersées sur plusieurs ordinateurs hôtes sont souvent appelées simulations distribuées. Il existe plusieurs normes militaires pour la simulation distribuée, y compris le protocole de simulation de niveau d'agrégation (ALSP), la simulation interactive distribuée (DIS) et l'architecture de haut niveau (HLA).

1.4 Mesures et évaluation des performances (PET)

La modélisation traite la représentation des sous-systèmes et des sous-processus interconnectés avec un objectif final d'obtenir une estimation d'une propriété systémique agrégée ou d'une mesure de la performance.

Le mécanisme par lequel on peut obtenir cette estimation est le sujet de l'évaluation de la performance. Une estimation qui doit être :

- Obtenue efficacement.
- Obtenue avec confiance.

Une technique d'évaluation des performances (PET) est défini comme un ensemble d'hypothèses et de processus analytiques (appliqués dans le contexte d'une simulation) et dont l'objectif est l'estimation efficace de certaines mesures de performance. Les PET se répartissent en deux grandes catégories :

- Mesure directe.
- Modélisation.

1.4.1 Mesure directe

C'est la technique la plus évidente et la plus efficace pour l'évaluation de la performance d'un système. Cependant, certaines limites sont présentes :

- La mesure directe d'un système n'est disponible qu'avec les systèmes opérationnels. La mesure directe n'est pas possible avec les systèmes en cours de conception ou de développement.(on ne peut pas faire des mesures directes sur des systèmes inexistants)
- La mesure directe du système peut affecter le système mesuré tout en obtenant les données requises. Ceci peut conduire à des mesures erronées du système mesuré.
- Il peut ne pas être pratique de mesurer directement le niveau de performance de bout en bout soutenu sur tous les chemins du système. Par conséquent, pour obtenir les objectifs de performance, des modèles analytiques doivent être mis en place pour convertir les mesures brutes en mesures significatives de performance.

1.4.2 Modélisation

Lors des phases de conception et de développement d'un système, la modélisation peut être utilisée pour estimer les mesures de performance à obtenir lorsque le système est mis en œuvre. La modélisation peut être utilisée pour évaluer les performances d'un système opérationnel, en particulier après que le système subit quelques modifications.

La modélisation n'affecte pas le système mesuré comme c'est le cas de la technique de mesure directe, comme elle peut être utilisée pendant les phases de conception. La modélisation est extrêmement importante dans la conception et le développement d'un système, car elle (si elle bien faite) permet d'avoir une idée sur la façon dont ce système se comporterait. (Modélisation un système = le représenter d'une manière plus simple)

Avec la modélisation, les paramètres du système peuvent être modifiés, testés et analysés. Plus important encore, la modélisation, si elle est bien gérée, elle permet de réduire les couts de développement du système. Pour modéliser un système, certaines hypothèses simplifiées sont souvent nécessaires. Il est important de noter que : beaucoup d'hypothèses simplifièrent la modélisation, mais peuvent conduire à une représentation inexacte du système.

Cependant, la modélisation souffre des problèmes suivants :
- Le problème de l'abstraction du système. Cela peut conduire à analyser un modèle qui ne représente pas le système réel en cours d'évaluation.
- Problèmes de représentation de la charge de travail du système.
- Problèmes dans l'obtention de mesures de performance pour le modèle et la cartographie des résultats au système réel.

Il existe deux approches largement définies pour déterminer les mesures de performance réelles :

1. La modélisation analytique.
2. La modélisation par simulation.

Modélisation analytique. Le concept général de la modélisation analytique est d'abord de trouver une façon de décrire le système mathématiquement à l'aide des mathématiques appliquées et des outils tels que les fils d'attente et les théorèmes probabilistes, puis appliquer les méthodes numériques pour avoir un aperçu du modèle développé mathématiquement.

Quand le système est simple et relativement petit, la modélisation analytique est préférable (par rapport a la simulation). Dans ce cas, le modèle tend à être mathématiquement traitable et les solutions numériques ne sont pas exigeantes en puissance de calcul.

Si elle est correctement utilisée, la modélisation analytique peut être rentable et peut fournir une vue abstraite des composants interagissent les uns avec les autres dans le système. Cependant, si de nombreuses hypothèses simplificatrices sur le système sont faites au cours du processus de modélisation, les modèles analytiques ne peuvent pas donner une représentation précise du système réel.

Modélisation par simulation En revanche, la modélisation par simulation détermine les mesures de performance en effectuant des mesures directes d'un système virtuel simplifié qui a été créé par des moyens informatiques.

La simulation est largement utilisée dans la modélisation des systèmes (pour des applications allant de la recherche en ingénierie, analyse d'affaires, planification de la production, l'expérimentation et la science biologique, ...) Par rapport à la modélisation analytique, la simulation

nécessite généralement moins d'abstraction dans le modèle (par exemple, moins d'hypothèses) et presque tous les détails possibles sur les spécifications du système peut être mis dans le modèle de simulation pour mieux décrire le système actuel. Lorsque le système est assez vaste et complexe, une formulation mathématique simple peut ne pas être possible ; dans ce cas, l'approche de simulation est généralement préféré à l'approche analytique.

Tout comme la modélisation analytique, la modélisation par simulation peut laisser de côté certains détails, car trop de détails peut entrainer une simulation ingérable et un effort de calcul considérable. Donc il est important d'examiner attentivement les mesures à étudier et à ne pas inclure les détails sans importance dans la simulation.

Dans la modélisation par simulation, il existe peu de PET qui peuvent être appliqués en général, bien que de nombreux PET soient basés sur des modifications de la méthode dite de Monte Carlo. Plus souvent, chaque PET doit normalement être conçu sur **Au cas par cas** compte tenu du système en place. Le processus de définition d'un PET implique :

1. Les hypothèses ou les simplifications des propriétés des éléments atomiques dans un système et la logique sous-jacente de leur évolution par les interactions.
2. Les Hypothèses statistiques sur les propriétés des aspects non déterministes du système.
3. Les techniques statistiques pour agréger les estimations de la mesure de la performance obtenues à partir de disparates simulations de différentes configurations de systèmes.

1.5 Techniques de simulation :

Les Techniques de simulations les plus utilisées sont :

1.5.1 Emulation

Le processus de conception et de construction de matériel (c'est-à-dire, prototype) qui imite la fonctionnalité du système réel. L'émulation réseaux signifie que le réseau est simulé dans le but d'évaluer ces performances ou pour prédire l'impact des changements ou des optimisations réalisées. (Attacher un équipement terminal à un émulateur et ce dernier agira exactement comme s'il attouchés à un réseau réel). Le point important à noter ici est que la mission d'un émulateur réseaux est d'émuler le réseau qui permet de connecter les terminaux et pas les terminaux eux-mêmes.
Exemple : NS2 est un simulateur réseaux qui peut être utilisés comme un émulateur mais avec des fonctions limites. au contraire de WANSIM qu'est un émulateur typique qui permet l'émulation de bridged wan et qui utilise quelque fonctionnalités de linux.

1.5.2 Simulation Monte Carlo

C'est une simulation qui n'a pas d'axe temporel. La simulation de Monte Carlo est utilisée pour modéliser des phénomènes probabilistes qui ne changent pas avec le temps, ou pour évaluer des expressions non probabilistes utilisant des techniques probabilistes.

C'est une technique de simulation classique. En fait, Monte Carlo n'est qu'une application particulière d'une méthode autrement générale, applicable à la fois dans des contextes déterministe et probabiliste. Au cœur de Monte Carlo on trouve une procédure de calcul dans laquelle une mesure de performance est estimée à l'aide d'échantillons tirés au hasard d'une population ayant des propriétés statistiques appropriées.

La sélection des échantillons, à son tour, nécessite un générateur de nombres aléatoires (RNG) approprié. Idéalement, les séquences "aléatoires" générées sont une contrepartie logicielle complètement fidèle du non-déterminisme sous-jacent au processus réel. Le terme «méthode de Monte Carlo» est généralement utilisé pour désigner toute technique de simulation liée à l'utilisation de

nombres aléatoires. Cela inclut des méthodes telles que la simulation de Monte Carlo, l'intégration de Monte Carlo, l'échantillonnage d'importance, les algorithmes génétiques, le recuit simulé, l'algorithme de Hasting-Metropolis, la percolation, la marche aléatoire et le dépôt balistique, pour n'en nommer que quelques-uns. Parmi celles-ci, la simulation de Monte Carlo est peut-être la classe de méthodes numériques la plus courante et la mieux comprise pour les simulations et les expériences informatiques. La simulation Monte Carlo génère des entrées aléatoires (en référence à une distribution connue) et utilise les entrées pour déterminer une valeur numérique de la mesure de la performance.

La partie essentielle de la simulation de Monte Carlo est une procédure pour générer un nombre pseudo-aléatoire R qui est réparti uniformément sur l'intervalle 0 <R <1. Une telle procédure est appliquée à plusieurs reprises au sein de la simulation Monte Carlo pour produire une séquence de nombres aléatoires générés indépendamment.

1.5.3 Trace-driven simulation

Toute simulation qui utilise une liste ordonnée d'événements du monde réel comme entrée.

1.5.4 Simulation à évènements continus

Dans certains systèmes, les changements d'état se produisent tout le temps, pas simplement à des instants discrets. Par exemple, le niveau d'eau dans un réservoir avec des entrées et des sorties données peut changer tout le temps. Dans de tels cas, la «simulation continue» est plus appropriée, même si la simulation par évènements discrets peut servir d'approximation.

1.5.5 Simulation à évènements discret

Une simulation à évènements discrets se caractérise par deux caractéristiques :

1. dans n'importe quel intervalle de temps, on peut trouver un sous-intervalle dans lequel aucun évènement ne se produit et aucune variable d'état ne change ;
2. le nombre d'évènements est fini.

Toutes les simulations à évènements discrets ont les composantes suivantes :

- File d'attente d'évènements : liste contenant tous les évènements en attente (à l'avenir). La mise en œuvre de la liste des évènements et des fonctions à exécuter sur celle-ci peut affecter de manière significative l'efficacité du programme de simulation.
- Horloge de simulation : Une variable globale qui représente le temps simulé. Le temps de simulation peut être avancé par des méthodes pilotées par le temps ou par des évènements. Dans l'approche temporelle, le temps est divisé en incréments constants et petits, puis les évènements se produisant dans chaque incrément sont vérifiés. Dans l'approche évènementielle, d'autre part, le temps est incrémenté au moment du prochain évènement imminent. Cet évènement est traité et l'horloge de simulation est de nouveau incrémentée à l'heure de l'évènement imminent suivant, et ainsi de suite. Cette dernière approche est celle qui est généralement utilisée dans les simulations informatiques.
- Variables d'état : Variables qui ensemble décrivent complètement l'état du système.
- Routines d'évènements : Routines qui gèrent l'occurrence des évènements. Si un évènement se produit, sa routine d'évènement correspondante est exécutée pour mettre à jour les variables d'état et la file d'attente d'évènements de manière appropriée.
- Saisie d'entrée : routine qui obtient les paramètres d'entrée de l'utilisateur et les fournit au modèle.
- Génération de rapports : routine chargée de calculer les résultats et de les imprimer à l'utilisateur final.

- Routine d'initialisation : routine chargée d'initialiser les valeurs des différentes variables d'état, des variables globales et des variables statistiques au début du programme de simulation.
- Programme principal : Le programme où les autres routines sont appelées. Le programme principal appelle la routine d'initialisation ; Le sous-programme d'entrée exécute différentes itérations, appelle enfin la routine de génération de rapport et termine la simulation.

1.6 Modélisation des entrées et analyse des résultats

1.6.1 Modélisation des entrées

Les modèles d'entrée influent grandement sur les résultats d'un modèle de simulation. Dans les applications de simulation du monde réel, il est crucial que l'on choisisse les distributions appropriées pour représenter les données d'entrée. Si la distribution des intrants est mal interprétée, cela conduira à des conclusions inexactes sur le système. Les modèles d'entrée sont utilisés pour représenter les caractéristiques de l'aléatoire ou de l'incertitude de la source d'entrée. La modélisation d'entrée est l'acte de choisir la représentation qui exprime le mieux les données source. Cette tâche implique souvent de choisir une famille appropriée de distributions de probabilité. La tâche est simplifiée si l'on peut faire l'une des hypothèses suivantes :

- Les données d'entrée peuvent être décrites comme une séquence de variables aléatoires indépendantes, et la distribution pour chacune d'elles est identique.
- La distribution commune partagée par toutes les variables aléatoires fait partie d'une famille générale de distributions qui sont largement implémentées dans la plupart des systèmes de simulation ou des langages, par exemple le binôme, Poisson, normal, log normal, exponentiel, gamma, bêta, Erlang, Weibull, discrète Ou des distributions continues uniformes, triangulaires ou empiriques.
- Les données d'entrée peuvent être représentées comme générées par un processus stochastique stationnaire, pour lequel la distribution de probabilité reste la même pour tous les temps. Ses paramètres, tels que la moyenne et la variance (s'ils existent), ne changent pas non plus avec le temps ou la position.

Il existe quatre grandes étapes pour développer un modèle utile de données d'entrée :

1. Collecter des données d'un scénario réel.
2. Choisissez une distribution de probabilité pour représenter le processus d'entrée.
3. Identifier les paramètres associés à la famille de distribution.
4. Évaluer la qualité d'ajustement de la distribution choisie et les paramètres associés.

Ces quatre étapes sont la procédure standard à suivre quand on veut une correspondance étroite entre le modèle d'entrée et la vraie distribution de probabilité sous-jacente associée au système. En général, il y a aussi cinq questions fondamentales qui doivent être répondu quand il y a un ensemble de données recueillies sur l'élément d'intérêt :

1. Les données ont-elles été recueillies de manière appropriée ?
2. Un modèle non paramétrique ou un modèle de distribution de probabilité paramétrique devrait-il être choisi comme mécanisme probabiliste d'entrée ?
3. Si un modèle paramétrique est choisi, quel type de distribution décrira le mieux les données ? En d'autres termes, quel type de générateur de variables aléatoires semble générer la même distribution de données ?
4. Si l'on choisit un modèle paramétrique, quelles sont les valeurs du ou des paramètres qui correspondent le mieux à la distribution de probabilité ? Si la distribution est Gamma (a, b), par exemple, quelle est la valeur de a et b ?
5. Après que la distribution de probabilité et ses paramètres associés ont été choisis, dans quelle mesure avons-nous confiance que les valeurs choisies sont "meilleures" pour décrire le processus d'entrée ?

1.6.2 Analyse des résultats

De nombreuses études de simulation incluent le caractère aléatoire afin d'avoir une idée des intrants typiques auxquels le système pourrait faire face. Par exemple, dans la simulation d'un système de processeur, les temps de traitement requis peuvent suivre une distribution de probabilité donnée ou les temps d'arrivée de nouveaux travaux peuvent être définis de façon stochastique. De même, dans une simulation bancaire, les clients peuvent arriver à des moments aléatoires et le temps passé à un guichet peut être stochastiquement déterminé.

En raison du caractère aléatoire des composants conduisant une simulation, sa production est également aléatoire, de sorte que les techniques statistiques doivent être utilisées pour analyser les résultats et les agréger en mesures significatives à partir desquelles tirer des conclusions.

Les méthodes d'analyse de données dans les cours de statistique d'introduction supposent généralement que les données sont i.i.d. Avec une distribution normale. Malheureusement, les données de sortie des simulations n'est souvent pas i.i.d. Et pas normal. Par exemple, considérez les temps d'attente des clients avant de voir un guichetier dans une banque. Si un client a un temps d'attente exceptionnellement long, alors le prochain client probablement aussi, donc les temps d'attente des deux clients sont dépendants. En outre, les clients qui arrivent à l'heure du déjeuner auront généralement des délais d'attente plus longs que les clients qui arrivent à d'autres moments, donc les délais d'attente sont non identiquement réparties tout au long de la journée.

Enfin, les temps d'attente sont toujours positifs et souvent biaisés vers la droite, avec un mode possible à zéro, donc les temps d'attente ne sont pas normalement distribués. Pour ces raisons on ne peut souvent pas analyser la sortie de simulation en utilisant les techniques statistiques classiques développées pour i.i.d. des données normales. Pour ces raisons, des méthodes statistiques plus sophistiquées sont nécessaires pour examiner et analyser les expériences de simulation pour mesurer la performance.

L'une des premières étapes de toute étude de simulation est le choix de la mesure de la performance pour le calcul. En d'autres termes, quelles mesures seront utilisées pour évaluer la «bonne» qualité du système. Par exemple, la performance d'un système de file d'attente peut être mesurée par le nombre attendu de clients desservis en une journée, ou bien nous pouvons utiliser le coût quotidien moyen à long terme ou l'inverse de la taille maximale de la file d'attente ou l'inverse de la moyenne de la taille de la file d'attente comme mesure du rendement d'une chaîne d'approvisionnement. Il existe principalement deux types de mesures de performance pour les systèmes stochastiques :

1. Mesures des performances transitoires, également appelées mesures de terminaison ou d'horizon fini, qui évaluent l'évolution du système sur un horizon de temps fini.
2. Mesures de performance en régime permanent, qui décrivent comment le système évolue sur un horizon temporel infini. Ces mesures sont également appelées mesures à long terme ou à horizon indéfini.

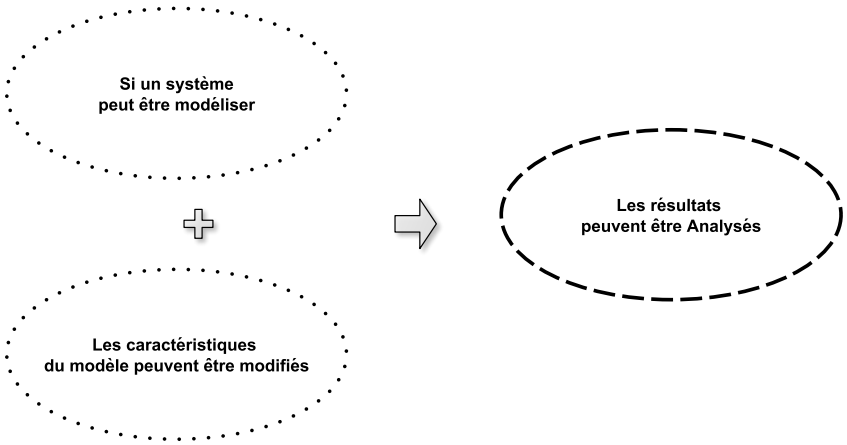
1.7 Simulation réseau.

Dans le domaine de recherche en réseaux, il est très couteux de faire un test complet contenant tout les composantes réseaux (hosts, routeurs et liaisons de données . . .) pour valides et vérifier un protocole ou un algorithme spécifique ; les simulateurs réseaux permet d'économiser beaucoup de temps et de moyennes.

La simulation est «le processus de conception d'un modèle d'un système réel et de tenir des expériences avec ce modèle dans le but de comprendre le comportement de ce système et / ou l'évaluation des différentes stratégies pour son fonctionnement." avec la nature dynamique des ré-

seaux informatiques, nous avons donc fait face à un modèle dynamique d'un système dynamique réel.

De manière générale, les simulateurs réseaux essayent de modéliser les réseaux du monde réel, l'idée est que :



Un simulateur réseaux est composé d'un large éventail de technologie réseaux et de protocoles et permet aux utilisateurs de construire des réseaux complexes à partir de bloc de composantes de base (nœuds, liens, ...) avec ces simulations, on peut concevoir différents topologie, utiliser différents type de nœuds. (nœuds terminaux, concentrateurs, pont réseaux, routeurs, unités mobiles, . . .). Comme le processus de modification du modèle est relativement facile (bon marché) que la mise en œuvre complète et réelle, une grande variété de scénarios peuvent être analysés à faible coût (par rapport à des changement sur des réseaux réels).

Toute fois, les simulateurs réseaux ne sont pas parfais, ils ne peuvent pas donner parfaitement tous les détails du modèle des réseaux, Cependant, si ils sont bien modélisés, ils seront suffisamment proches de façon a donner un aperçu significatif sur les réseaux à testés et monter comment tout changement d'un paramètre affecte le fonctionnement du réseaux.

Les simulateurs réseaux sont particulièrement utilises par les concepteurs réseaux pour le test de nouveaux protocoles ou le chargement de protocoles existants d'une manière contrôlée et reproductible.

1.7.1 Les Bases de simulation réseau.

La plupart des simulations de réseaux [3] [4] [5] [6] [7] sont basées sur des techniques de simulation d'évènements discrets. Dans ce cas, les changements d'état se produisent à des moments discrets du temps de simulation en raison d'évènements tels que la génération des messages, l'arrivée des paquets, le départ des paquets, etc. Lors de chaque évènement, certains traitements doivent être effectués et pris dans l'ordre de l'horodatage Arrivée de l'évènement.

Les éléments de base de toute simulation réseaux sont :

1. **Entités :** sont des objets qui interagissent les uns avec les autres dans un programme de simulation pour provoquer des modifications de l'état du système. Dans le cadre d'un réseau informatique, les entités peuvent inclure des nœuds informatiques, des paquets, des flux de données, ou des objets soft tels que des horloges de simulation. Pour distinguer les différentes entités, des attributs uniques sont attribués à chacun d'eux. Par exemple, une entité paquet peut avoir des attributs tels que la longueur du paquet, le numéro de séquence, la priorité et l'en-tête.
2. **Ressources :** Les ressources font partie de systèmes complexes. En général, une quantité limitée de ressources doit être partagée entre un certain ensemble d'entités. C'est généralement

le cas pour les réseaux informatiques, où la bande passante, le temps d'émission, le nombre de serveurs, par exemple, représentent des ressources de réseau qui doivent être partagées.

3. **Activités et évènements :** De temps en temps, les entités se livrer à certaines activités. Cet engagement crée des évènements et déclenche des changements dans l'état du système. Un exemple courant d'activités qui comprend retard et attente : Lorsqu'un ordinateur a besoin d'envoyer un paquet, mais trouver le support occupé, il attend que le support se libère et dans ce temps le paquet s'engage dans une activité d'attente.
4. **Ordonnanceur :** Un Ordonnanceur gère la liste des événements et leur temps d'exécution. Lors d'une simulation, il dirige une horloge de simulation, crée des événements et les exécute.
5. **Variables globales :** Une variable globale est accessible par n'importe quelle fonction ou entité dans le système, et conserve essentiellement la trace de certaines valeurs communes de la simulation. Dans le cadre des réseaux informatiques, ces variables peuvent représenter, par exemple, la longueur de la file d'attente de paquets pour un serveur, le temps d'occupation d'antenne dans un réseau sans fil ou le nombre total de paquets transmis.
6. **Générateur de nombres aléatoires :** Un générateur de nombres aléatoires (RNG) est nécessaire pour introduire l'aléatoire dans le modèle de simulation. Dans de nombreuses situations, plusieurs résultats statistiques sont nécessaires. Un RNG doit commencer à récolter des valeurs de autre emplacement mais dans le même prédéfini psudo-aléatoire. Dans le cas contraire, les résultats de chaque exécution serait le même. Dans une mise en œuvre effective, un RNG est initialisé avec une graine. Une graine identifie l'emplacement de départ dans un ordre aléatoire où un RNG commence le prélèvement des nombres. Différentes Simulations initialisé avec différentes graines génère donc des résultats différents (mais statistiquement identiques). Dans une simulation réseau, le processus d'arrivée des paquets, processus en attente ou les processus de service sont généralement modélisés comme des processus aléatoires.
7. **Collecteur de statistiques :** Il permet de collecter les données générées par la simulation afin que des inférences significatives peuvent être tirées de ces données.

Actuellement, plusieurs modules sont disponibles pour la réalisation de simulations de réseau. Quelques-uns des plus populaires sont NS2, Opnet et GloMoSim.
La première étape dans la réalisation d'une expérience de simulation de réseau est de créer un modèle de la topologie de réseau. Ceci peut être réalisé en définissant un ensemble de nœuds et leurs interconnexions. Les différents aspects de la topologie de réseau normalement spécifiés sont les suivants :

Routeurs réseau : Les différents aspects spécifiés pour les routeurs réseau incluent le nombre d'interfaces réseau, les caractéristiques des supports physiques utilisés par les différentes interfaces, le type de file d'attente déployé à chacune des interfaces de sortie, le type de protocole de routage utilisé par chacune Interface du routeur, et les disciplines de mise en file d'attente d'entrée et de sortie déployées.

Le dernier paramètre est généralement spécifié pour les routeurs haute performance, pour lesquels la file d'attente d'entrée (file d'attente d'entrée) et la file d'attente de sortie (file d'attente de sortie) doivent être spécifiées.

Liaisons de communication : Les différents aspects spécifiés pour une liaison de communication comprennent le taux de transmission de la liaison (par exemple, 100 Mbps), le délai de propagation du support et si la liaison est full duplex ou half duplex.

Systèmes d'extrémité : Les différents aspects qui doivent être spécifiés pour un système d'extrémité comprennent habituellement les protocoles (par exemple, IPv4, TCP, UDP, etc.) pris en charge, le retard éprouvé pendant que les paquets progressent et descendent la pile de protocoles et l'action qui Le système d'extrémité effectue en réponse à des paquets illégaux ou inattendus.

Les autres aspects généralement spécifiés pour une expérience de simulation comprennent les caractéristiques de trafic. Par exemple, pour simuler une application Web, il faut spécifier la distri-

bution de probabilité de la requête au serveur Web, la taille de la réponse du serveur et le temps de traitement requis par le serveur.

1.7.2 Type du simulateur réseaux

On peut classes le simulateur réseaux par rapport a plusieurs critères :

1. • Simulateurs commerciaux.
   • Simulateurs open source.
2. • Simple.
   • Complexe.

Simulateurs commerciaux et open source Un simulateur réseau commercial ne fournit ni le code source du logiciel ni le code des paquets aux utilisateurs finaux, l'utilisateur paye pour obtenir une licence d'utilisation uniquement. L'avantage principal d'un simulateur commercial est qu'il est bien documenté et qu'il à périodiquement des mise à jour.

Un simulateur open source à l'avantage d'être très ouvert, le code du programme est donné, de cette manière tous le monde peut contribuer à trouver les bogues et améliorer les fonctionnalités. Pas seulement le code, l'interface est aussi ouverte, les simulateurs open source présente l'avantage qu'ils sont très flexibles et peuvent suivre les nouvelle technologies de manière rapide par rapport aux simulateurs commerciaux. Le point faible des simulateur open source est la documentation surtout quand les nouvelles versions sortant avec beaucoup de nouvelles améliorations.

Simple et complexe Actuellement, il existe une grande variété de simulateurs réseaux, allons du plus simple au plus complexes. Au minimum un simulateur réseaux doit permettre aux utilisateurs de représentes une topologie réseaux, la définition de certaines nombres de scénarios, la spécification des nœuds du réseaux et les liens entre ces nœuds et le trafic Un simulateur réseaux plus complexe peut permette à l'utilisateur de tout spécifier tout ce qui concerne les protocoles utilisés pour traiter le trafic réseaux. Les applications graphiques permettront également aux utilisateurs de visualises le fonctionnement de l'environnement simulé.

1.8 CAC et contrôle de congestion

1.8.1 Le contrôle d'établissement des connexions

Le contrôle d'établissement des connexions (dit aussi contrôle d'admission d'appel (CAC : Call Admission Control)) permet de limiter le nombre de connexions dans le réseau de manière à permettre la garentie de la qualité des connexions établies en maintenant le partage des ressources entre un nombre raisonnable de connexions.

Le CAC est un ensemble d'actions et de permissions dans la communication réseau qui identifie où la connexion est permise (cette decession est basée sur la capacité du réseau. Cet ensemble d'actions de réseau est initié pendant la configuration de la connexion ou lorsque les connexions sont reconnectés. Il est basé sur un algorithme simple utilisé pour isloer un trafic réseau entrant ou sortant. CAC est également utilisé pour décider quel trafic doit être autorisé ou rejeté par un réseau spécifique, il est souvent utilisé dans les réseaux ATM.

Le rôle de base du contrôle d'admission de connexion dans les réseaux orientés connexion est de décider s'il existe suffisamment de ressources réseaux libres disponibles avant d'établir une nouvelle connexion. Une connexion de bout en bout n'est établie que lorsque la disponibilité des ressources gratuites est assurée.

CAC effectue les deux opérations suivantes lors de l'établissement d'une connexion :

• Il établit une connexion lorsque les ressources sont gratuites et disponibles.

- Si la connexion est rejetée en l'absence de ressources libres, une notification est renvoyée à l'expéditeur ou au demandeur de l'appel ou de la connexion.

Les facteurs suivants doivent être pris en compte lors de l'établissement ou de la demande de connexion :

1. Le type de service requis.
2. Paramètres de trafic. (les paramètres de trafic source sont analysés)
3. Les deux directions demandent leur QoS requise, qui est également prise en compte lors de l'établissement d'une connexion.

CAC réactif ou CAC préventif Il existe deux types de CAC : un CAC réactif et un CAC préventif.

- Un CAC réactif, réagit après coup : il pourra mettre fin à certaines connexions qui sont en cours pour en privilégier d'autres et cela en cas d'une réduction importante de la capacité. Il favorise alors le taux d'acceptation des connexions.
- Un CAC préventif, quant à lui, peut refuser d'admettre certaines connexions lorsqu'il estime que des affaiblissements peuvent survenir sur le lien de communication. Ainsi, un CAC préventif privilégie un faible taux d'abandon des connexions.

1.8.2 Contrôle de congestion

La congestion du réseau dans le réseau de données et la théorie des files d'attente est la qualité de service réduite qui se produit lorsqu'un nœud de réseau transporte plus de données qu'il ne peut gérer.

Les effets typiques incluent le retard de file d'attente, la perte de paquets ou le blocage de nouvelles connexions. Une conséquence de la congestion est qu'une augmentation progressive de la charge offerte conduit soit à une faible augmentation, soit même à une diminution du débit du réseau.

Les différentes phases basiques des algorithmes d'évitement de congestion : Le principe de base d'évitement de congestion est de réagir en réduisant le débit de la connexion. Les protocoles mesurent l'importance du problème en observant divers phénomènes comme l'augmentation du temps de réponse ou la duplication de messages de type ACK signifiant une perte de paquet par exemple. Si le protocole ne réagit pas aux congestions, le nombre de retransmission peut continuer à augmenter et aggraver ainsi la congestion. C'est la raison pour laquelle un algorithme de contrôle réduit le flux en cas de congestion.

Ces phases sont [8] :

1. **Slow Start :** Lors du démarrage d'une connexion, nous ne connaissons pas le lien qu'il y a entre nous et le destinataire, donc on va progressivement augmenter le nombre de paquets qu'on envoie. Le but est donc de trouver la bande passante disponible, et utiliser toutes les ressources à disposition. On va alors utiliser une fenêtre d'émission (cwnd), qui représente un nombre de paquets à envoyer sans attendre d'acquittement. Cette fenêtre grandira au fur et à mesure que nous recevons des acquittements pour les paquets envoyés (chaque ACK fera augmenter cette fenêtre de 1, autrement dit, chaque RTT doublera la taille de la fenêtre).
2. **Congestion Avoidance :** Au-delà d'une certaine limite de valeur de cwnd (slow start threshold, ssthresh), TCP passe en mode d'évitement de congestion. À partir de là, la valeur de cwnd augmente de façon linéaire et donc bien plus lentement qu'en slow start : cwnd s'incrémente de un MSS (= un paquet) à chaque RTT. Dans ce mode de fonctionnement, l'algorithme détecte aussi rapidement que possible la perte d'un paquet : si nous recevons trois fois le ACK même paquet, on n'attend pas la fin d'un timeout pour réagir. En réaction à cette perte, on fait descendre la valeur de ssthresh ainsi que cwnd (on repasse éventuellement en mode de Slow Start). On utilise la technique de Fast Retransmit pour renvoyer rapidement les paquets perdus.

3. **Fast Retransmit :** Comme expliqué plus tôt, on passe par cet algorithme en cas de détection de perte de segment(s) lorsque nous sommes en mode « Congestion Avoidance ». Un ACK dupliqué s'explique par la manière dont les segments sont traités à la réception. En effet, si nous recevons un segment TCP qui n'est pas dans l'ordre attendu, on doit envoyer un ACK avec une valeur égale au numéro de segment qui était attendu. S'il y a une perte de segment, le destinataire n'enverra plus que des ACK avec le numéro du segment perdu. On peut donc pallier cette perte rapidement (après 3 ACK dupliqués généralement), sans attendre le timeout.
4. **Fast Recovery :** Plutôt que repasser en mode Slow Start lors d'une duplication de ACK (et après un passage par le mode Fast Retransmit), nous renvoyons le segment perdu et on attend un ACK pour toute la fenêtre transmise précédemment avant de retourner en mode Congestion Avoidance. Si on atteint le timeout, on repart en mode Slow Start. Grâce à cette technique nous évitons de baisser le débit d'une façon trop brutale.

Les algorithmes historiques : Les algorithmes qui ont (ou non) été utilisés, mais désormais obsolètes.

1. TCP tahoe.
2. Reno.
3. New Reno.
4. Vegas.
5. Westwood(+).
6. SACK TCP.

Les implémentations actuelles :

1. CUBIC : il est basé sur **BIC**, cependant il a une phase montante plus douce que celle de BIC, ce qui le rend plus « amical » envers les autres versions de TCP. CUBIC est plus simple car ne possède qu'un seul algorithme pour sa phase montante, et n'utilise pas les acquittements pour augmenter la taille de la fenêtre, on préfèrera parler d'évènement de congestion. C'est ce qui rend CUBIC plus efficace dans les réseaux à bas débit ou avec un RTT court.
2. Compound TCP : Une des grandes particularités de CTCP est qu'il maintient deux fenêtres de congestion. La première est la fenêtre qui augmente de façon linéaire mais décroît en cas de perte via un certain coefficient. La seconde fenêtre est liée au temps de réponse du destinataire. On combine donc des méthodes liées à la perte de paquet (comme TCP Reno) et d'adaptation préventive à la congestion (comme TCP Vegas), d'où le nom « Compound » (composé). La taille de la fenêtre réellement utilisée est la somme de ces deux fenêtres. Si le RTT est bas, alors la fenêtre basée sur le délai augmentera rapidement. Si une perte de segments survient, alors la fenêtre basée sur la perte de paquet diminuera rapidement afin de compenser l'augmentation de la fenêtre basée sur le délai. Avec ce système, on cherchera à garder une valeur constante de la fenêtre d'émission effective, proche de celle estimée. Le but de ce protocole est de maintenir de bonnes performances sur des réseaux avec un haut débit et avec un grand RTT.

1.9 AQM : Active Queue Management

L'active queue management ou gestion active de file d'attente, est un ensemble de techniques utilisées pour traiter le problème de congestion dans les réseaux IP, tel que le réseau Internet. Ces techniques, telles que RED (random early detection) ou BLUE, permettent d'optimiser la gestion des files d'attente des routeurs qui constituent le réseau.

1.9.1 Algorithmes AQM

Les algorithmes AQM les plus utilisés sont :

1. Random early detection (RED).
2. Random Exponential Marking (REM).
3. Blue and Stochastic Fair Blue (SFB).
4. PI controller.
5. Robust random early detection (RRED).
6. RSFB : a Resilient Stochastic Fair Blue algorithm against spoofing DDoS attacks.
7. RED with Preferential Dropping (RED-PD).
8. Controlled Delay (CoDel).

2 Simulateur NS2

Network Simulator version 2 (NS2) [9] est un simulateur d'évènement discret gratuit et open source conçu pour aider à la conception et à la recherche de réseaux. Il fournit un support important pour la simulation de presque tous les protocoles existants sur les réseaux câblés et sans fil (locaux et par satellite) dans différentes configurations.

Le simulateur NS a vu le jour à partir du simulateur *REAL* développé par *Srinivasan Keshav* en 1989. Par la suite, il a été élargi par le Network Research Group au Laboratoire National Lawrence Berkeley par Floyd et McCanne en 1995. NS2 est ensuite issu de ce projet par le projet VINT Virtual InterNetwork Testbed) au Lawrence Berkeley National Laboratory.

Plus tard, la NS3 a été introduite pour remédier à certaines des lacunes de NS2. Le noyau du moteur de simulation NS2 est implémenté en C^{++}. Cependant, pour configurer un scénario de simulation, la topologie doit être décrite en utilisant oTCL, un langage de script. La raison en est que, depuis le simulateur a été conçu dans les années 1990, les ordinateurs de bureau ont été plutôt lent. La topologie devait être spécifiée plusieurs fois, et les erreurs de spécification de la topologie devaient être corrigées plusieurs fois. La compilation des programmes C^{++} prenait beaucoup de temps, alors qu'un langage interprété comme oTCL nécessitait une interprétation de la ligne modifiée. Toutefois, avec la disponibilité de matériel puissant, le temps de compilation est à peine un problème. Par conséquent NS3 a été implémenté en C^{++} entièrement. NS2 est livré avec un package appelé NAM (Network Animator), un système d'animation Tcl qui produit une représentation visuelle du réseau spécifié. D'autre part, NS3 emploie un paquet connu sous le nom de PyViz, qui est un paquet de visualisation en temps réel python.

Actuellement, l'outil a plus de 300K lignes de code, avec un manuel de 400 pages. Il a une très grande base d'utilisateurs qui se propage autour du globe et englobe les utilisateurs de l'industrie et du milieu universitaire. Il est devenu un standard de fait dans l'étude de réseau et de recherche en raison de sa disponibilité facile et libre et la flexibilité de l'incorporation de nouvelles conceptions selon les besoins des utilisateurs. Puisque les protocoles à différentes couches sont déjà intégrés dans NS2, un développeur de protocole développant le protocole à n'importe quelle couche peut simuler sans effort, puisque tous les protocoles de support ont déjà été programmés en lui.

2.1 Présentation du simulateur NS2

NS est essentiellement élaboré avec les idées de la conception par objets, de la réutilisation du code et de modularité. Il est aujourd'hui un standard de référence en ce domaine, plusieurs laboratoires de recherche recommandent son utilisation pour tester les nouveaux protocoles.

Le simulateur NS actuel est particulièrement bien adapté aux réseaux à commutation de paquets et à la réalisation de simulations de grande taille (le test du passage à l'échelle). Il contient les fonctionnalités nécessaires à l'étude des algorithmes de routage unicast ou multicast, des protocoles de transport, de session, de réservation, des services intégrés, des protocoles d'application comme FTP. A titre d'exemple la liste des principaux composants actuellement disponibles dans NS par catégorie est :

- Application : Web, ftp, telnet, générateur de trafic (CBR...) ;
- Transport : TCP, UDP, RTP, SRM ;
- Routage unicast : Statique, dynamique (vecteur distance) ;
- Routage multicast : DVMRP, PIM ;
- Gestion de file d'attente : RED, DropTail, Token bucket.

2.2 Structure du simulateur

NS2 a été programmé en utilisant deux langages orientés objet, C^{++} et Object Oriented Tool Command Language (OTcl). Les programmes OTcl sont écrits par l'utilisateur et agissent comme front-end, alors que les programmes C^{++} agissent comme l'arrière-plan qui exécute la simulation réelle. La figure 1 montre la structure de base de NS2. Comme on peut le voir sur cette figure, les protocoles réels ont été implémentés en interne à l'aide de C^{++}, alors que OTcl fournit un handle à l'utilisateur pour utiliser ces protocoles. La connexion entre OTcl et C^{++} a été faite par une interface appelée TclCL (Tcl avec Classes Library). Il fournit une couche de colle C^{++} sur OTcl.

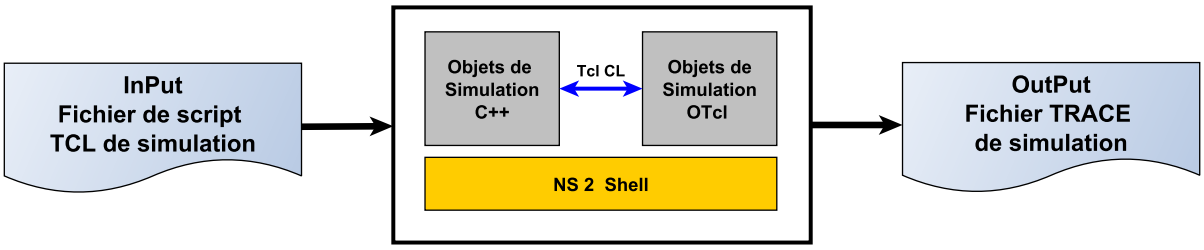


FIGURE 1 – Structure du Simulateur NS2

NS2 fournit un grand nombre d'objets C^{++} intégrés qui peuvent être utilisés pour configurer une simulation de base à l'aide d'un script de simulation Tcl. Cependant, un utilisateur avancé qui a l'intention de concevoir un nouveau protocole doit développer ses propres objets C^{++} et doit concevoir une interface de configuration OTcl pour accéder aux objets nouvellement conçus via un script Tcl.

Le code C^{++} s'exécute plus vite que le code OTcl, mais les modifications apportées au code C^{++} nécessitent plus de temps et d'effort que les modifications du code OTcl car le code C^{++} est compilé, alors que le code OTcl est interprété, c'est-à-dire un petit changement À $i = 1$) en code C^{++} compilerait le paquet complet, où un interprète ne peut exécuter que la ligne modifiée. En pratique, un utilisateur doit modifier fréquemment les valeurs d'entrée (configuration) lors du test d'une expérience de simulation. OTcl permet donc d'effectuer un grand nombre d'expériences de simulation avec différentes configurations de réseau, par exemple, un nombre différent de nœuds et des liens peuvent être créés à l'aide d'OTcl.

Les deux programmes C^{++} et OTcl se composent de deux hiérarchies de classes différentes, la hiérarchie liée et la hiérarchie autonome. Les classes de la hiérarchie liée des deux domaines sont liées ensemble à l'aide de TclCL comme le montre la figure 2. En d'autres termes, pour chaque classe disponible dans la hiérarchie C^{++}, une classe correspondante existe également dans OTcl, ce qui établit une correspondance un-un. La hiérarchie liée OTcl est appelée «hiérarchie interprétée» et la hiérarchie C^{++} de contrepartie est appelée «hiérarchie compilée» (voir la figure 2).

Le deuxième type de hiérarchie inclut des classes à partir des domaines qui ne sont pas liés ensemble. Ces classes ne font pas partie de la hiérarchie interprétée ni ne font partie de la hiérarchie compilée. Les utilisateurs peuvent avoir à programmer en utilisant n'importe quel (les deux) langue en fonction du problème de simulation de réseau spécifique à portée de main. Cependant, ce qui suit est une ligne directrice quant à quand utiliser OTcl et quand utiliser C^{++}.

- Utilisez OTcl :
  - Pour la configuration et la configuration du réseau.
  - Pour exécuter la simulation avec les modules NS2 existants.
- Utiliser C^{++} :
  - Modifier un ou plusieurs modules existants ou créer de nouveaux modules pour implémenter de nouveaux protocoles.

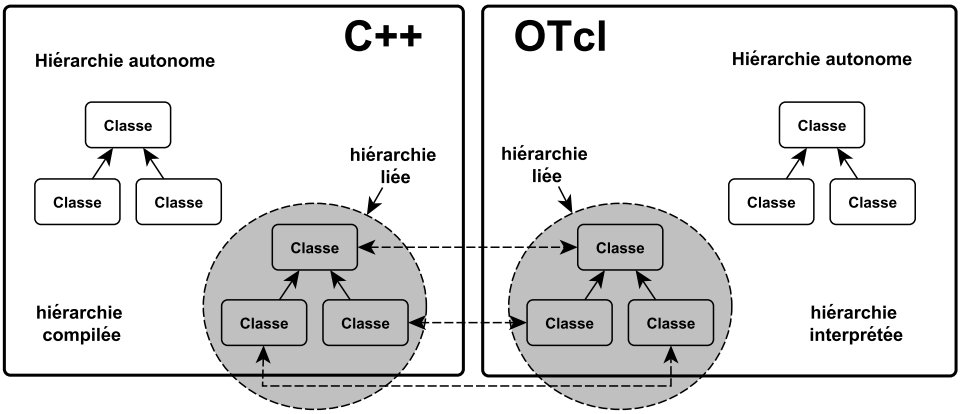


FIGURE 2 – Interconnection des domaines C^{++} et OTcl.

2.3 Entrée et sortie du simulateur

L'entrée dans le simulateur est un script TCL écrit en tant que fichier texte et interprété par le shell NS2. Le simulateur produit normalement deux fichiers (contenant du texte ASCII) en sortie : un fichier de trace NAM et un fichier de trace NS ou packet. Nous expliquons l'objectif de ces deux fichiers dans ce qui suit.

Le fichier de suivi NAM est utilisé comme entrée dans un autre package appelé Network Animator (NAM) qui anime la trace de simulation complète dans une interface utilisateur graphique. Un exemple de visualisation NAM est illustré à la figure 3, où deux nœuds communiquent via une liaison point à point. L'avantage d'utiliser l'animation dans la visualisation est double. L'utilisateur peut facilement vérifier que la topologie conçue par lui est réellement ce qu'il voulait, et il augmente la compréhensibilité du processus complet.

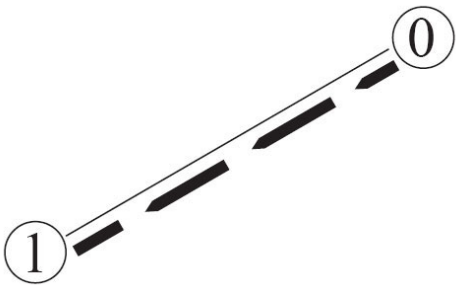


FIGURE 3 – Exemple de visualisation produit par NAM

Le deuxième fichier de sortie est appelé NS ou fichier de suivi de paquets. Il peut être utilisé pour étudier les paramètres de performance du réseau comme le retard, le débit, etc. Le fichier de trace contient généralement plusieurs lignes et chaque ligne représente un évènement qui s'est produit au cours de la simulation.

Ces lignes sont produites dans un format prédéfini à partir duquel les données requises peuvent être extraites à l'aide d'un outil de traitement de texte (écrit en utilisant un script ou un langage de programmation). Une fois que les données ont été collectées à l'aide de n'importe quel outil, l'un des outils graphiques peuvent être utilisés pour tracer les graphiques. Quelques outils qui sont populaires pour tracer les points de données rassemblés sont Xgraph, Gnuplot, MATLAB®, MS Excel, etc.

Même en utilisant ces outils, il est toujours fastidieux d'extraire les données des fichiers de trace, car aucun paquet prêt n'est fourni. Lors de vos simulations, vous pouvez utiliser des scripts shell, des scripts AWK, des programmes Perl ou $C/\ C^{++}$ pour extraire les informations requises d'un fichier de trace.

2.4 L'outil de visualisation NAM

NS2 ne permet pas de visualiser le résultat des expérimentations. Il permet uniquement de stocker une trace de la simulation, de sorte qu'elle puisse être exploitée par un autre logiciel, comme NAM.

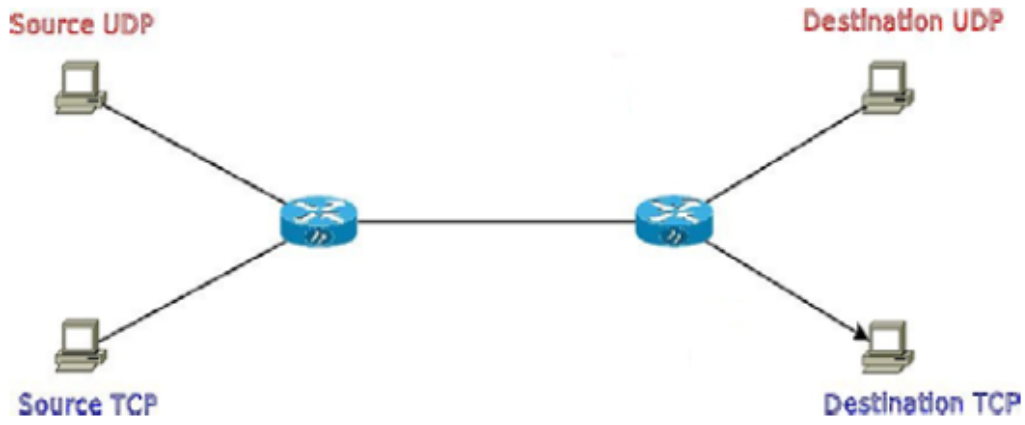
NAM est un outil de visualisation qui présente deux intérêts principaux : représenter la topologie d'un réseau décrit avec NS2, et afficher temporellement les résultats d'une trace d'exécution NS2.

Le paquet NAM est livré avec le logiciel appelé ns-allinone bundle (tarball). Il s'agit d'un outil de visualisation qui anime une simulation de réseau. Il a été développé à l'aide de Tcl/TK et il est utilisé pour afficher différents éléments de réseau tels que les nœuds, les liens et les évènements réseau tels que la transmission de paquets, la perte de paquets, le mouvement des nœuds, etc.

L'entrée de NAM est le fichier de trace NAM généré comme une sortie d'une simulation NS. Ce fichier de trace est en fait un simple fichier texte contenant différents champs, tels que des informations topologiques et des traces de paquets. L'outil NAM crée une visualisation basée sur les informations contenues dans le fichier de trace NAM. Il fournit également une interface utilisateur qui permet de contrôler différents aspects de l'animation, tels que la topologie, la vitesse de simulation, il est capable aussi de représenter des paquets TCP ou UDP, la rupture d'un lien entre nœuds, ou encore de représenter les paquets rejetés d'une file d'attente pleine. Ce logiciel est souvent appelé directement depuis les scripts TCL pour NS-2, pour visualiser directement le résultat de la simulation, Ou depuis la ligne de commande par la commande **nam**. Cela démarrera la fenêtre console NAM. Plusieurs animations peuvent s'exécuter sous la même instance nam.

2.5 Exemple de simulation avec NS2

Pour observer le fonctionnement de NS-2 et de son outil de simulation NAM, nous avons simulé un montage réseau simple, correspondant au diagramme suivant :



Pour cela, nous avons créé six nœuds NS2 (node), reliés par des liens full duplex (duplex-link) supportant des débits différents, un temps d'accès au medium de 10 ms, avec un algorithme de file d'attente DropTail2 pour la gestion des files d'attente.

La création des nœuds se fait par deux méthodes (explicite ou dynamique)
Méthode A :

```
set n0 [$ns node]
$n0 label "source UDP"
set n1 [$ns node]
$n0 label "source TCP"
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
$n0 label "destination UDP"
set n5 [$ns node]
$n0 label "destination TCP"
```

Méthode B :

```
set nbNodes 5
for {set i 0} {$i < $nbNodes } {incr i}
{
set n$i [$ns node]
lappend NodeList $n$i
}
```

Nous avons ensuite créé un agent UDP (Agent/UDP) attaché au noeud n0 (attach-agent) qui permet à n0 de transmettre des paquets UDP sur le réseau. Puis, nous avons créé un agent pour envoyer des paquets à débit constant (Constant Bit Rate -CBR-Application/Traffic/CBR) attaché à l'agent UDP défini précédemment (attach-agent).

```
# Cration de l'agent implant dans n0
set udp [new Agent/UDP]
$ns attach-agent $n0 $udp
# Traffic CBR de 500 octets toutes les 5 ms pour UDP0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp
```

Et de la même façon pour le trafic TCP

```
# Cration de l'agent implant dans n0
set tcp [new Agent/TCP]
$ns attach-agent $n1 $tcp

# Traffic CBR de 500 octets toutes les 10 ms pour tcp
set cbr1 [new Application/Traffic/CBR]
$cbr1 set packetSize_ 500
$cbr1 set interval_ 0.010
$cbr1 attach-agent $tcp
```

Nous avons enfin créé un agent vide, destiné à recevoir les paquets UDP sans les traiter (Agent/Null). On l'attache au noeud n4 (attach-agent) puis on connecte l'agent UDP et l'agent vide (connect).

```
# Cration d'un agent vide, destin   recevoir les paquets implants dans n3
set null0 [new Agent/Null]
$ns attach-agent $n4 $null0
$ns connect $udp $null0
```

Contrairement au UDP (protocole de transport sans connexion), TCP est un protocole orienté connexion ce qui demande une certaine coordination entre la source et la destination, pour cela nous avons mis un agent (Agent/TCPSink) au noeud de destination qui permet de communiquer avec le nœud source.

```
# Cration d'un agent vide, destin   recevoir les paquets implants dans n3
set tcpsink [new Agent/TCPSink]
$ns attach-agent $n5 $tcpsink
$ns connect $tcp $tcpsinkfloat
```

Enfin, on indique que les traces de simulation doivent être logées, qu'on souhaite lancer NAM à la fin de la simulation, que celle-ci va durer 10 secondes et que le CBR ne sera émis qu'à partir de la 0.5 ème seconde de simulation, et jusqu'à la 14.5 ème seconde.

Le code TCL de la simulation, commenté en détail, est le suivant :

```
# Cration du simulateur
set ns [new Simulator]
# Cration du fichier de traces NS-2
set nf [open out.nam w]
$ns namtrace-all $nf
# Dfinition de classes pour la coloration
$udp set class_ 1
$tcp set class_ 2
# Coloration des classes : bleu pour udp (classe 1) et rouge pour tcp (classe
    2)
$ns color 1 Blue
$ns color 2 Red
# Procdure de fin de simulation, qui crit les donnes dans le fichier
# et lance NAM pour la visualisation
proc finish {} {
global ns nf
$ns flush-trace
close $nf
exec nam out.nam &
exit 0
}
# Cration des noeuds
set n0 [$ns node]
$n0 label "source UDP"
set n1 [$ns node]
$n0 label "source TCP"
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
```

```
$n0 label "destination UDP"
set n5 [$ns node]
$n0 label "destination TCP"
# Cration des liens, tous en 1Mbps/10ms de TR/file d'attente DropTail
$ns duplex-link $n0 $n2 1Mb 10ms DropTail
$ns duplex-link $n1 $n2 1Mb 10ms DropTail
$ns duplex-link $n2 $n3 1Mb 10ms DropTail
$ns duplex-link $n3 $n4 1Mb 10ms DropTail
$ns duplex-link $n3 $n5 1Mb 10ms DropTail
# gestion du layout de la topologie
$ns duplex-link-op $n0 $n2 orient right-down
$ns duplex-link-op $n1 $n2 orient right-up
$ns duplex-link-op $n2 $n3 orient right
$ns duplex-link-op $n3 $n4 orient right-up
$ns duplex-link-op $n3 $n5 orient right-down
# Cration de deux agents implants dans n0
set udp [new Agent/UDP]
$ns attach-agent $n0 $udp

set tcp [new Agent/TCP]
$ns attach-agent $n1 $tcp
# Traffic CBR de 500 octets toutes les 5 ms pour UDP0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp

# Traffic CBR de 500 octets toutes les 10 ms pour UDP1
set cbr1 [new Application/Traffic/CBR]
$cbr1 set packetSize_ 500
$cbr1 set interval_ 0.010
$cbr1 attach-agent $tcp

# Cration d'un agent vide, destin   recevoir les paquets implants dans n3
set null0 [new Agent/Null]
$ns attach-agent $n4 $null0

# Cration d'un agent vide, destin   recevoir les paquets implants dans n3
set tcpsink [new Agent/TCPSink]
$ns attach-agent $n5 $tcpsink

# Le traffic issu des agents udp0 et udp1 est envoy vers null0
$ns connect $udp $null0
$ns connect $tcp $tcpsink
```

```
# Sc\'e nario de dbut et de fin de  gnration  des paquets par cbr0
$ns at 0.5 "$cbr0 start"
$ns at 0.5 "$cbr1 start"
$ns at 14.5 "$cbr0 stop"
$ns at 14.5 "$cbr1 stop"

# La simulation va durer 15 secondes et appelle la procdure finish
$ns at 15.0 "finish"
#  Dbut  de la simulation
$ns run
```

On lance ensuite la simulation qui permet de lancer la visualisation du résultat avec l'outil NAM à l'aide de la commande :

```
$ns <le nom du fichier>.tcl
```

Ce qui donne le résultat suivant :

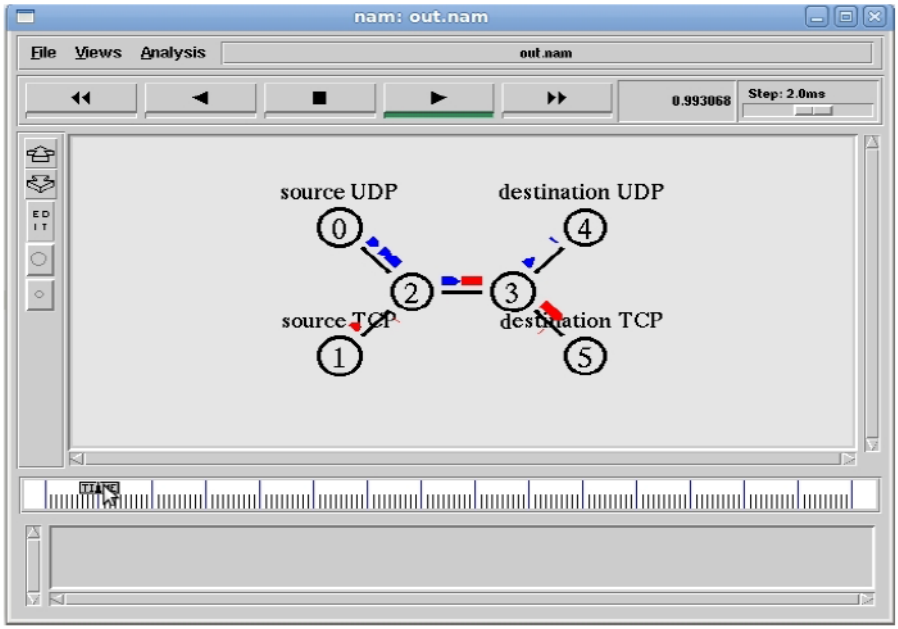


FIGURE 4 – ns2

Le logiciel NS2 et l'outil de visualisation NAM sont particulièrement adaptés à l'étude de réseaux complexes (filaires, sans fils) mettant en œuvre de nombreux types de files d'attente, protocoles de routage, de couches transport (UDP, TCP), et couches application.

3 Simulateur OPNET

OPNET (Optimized Network Engineering Tool) est un environnement de simulation orienté objet qui répond à toutes ces exigences et est le plus puissant simulateur de réseau polyvalent disponible aujourd'hui. L'outil d'analyse complet d'OPNET est particulièrement idéal pour interpréter et synthétiser les données de sortie. Une simulation d'évènement discret de la signalisation d'appel et de routage a été développée en utilisant un certain nombre de caractéristiques uniques d'OPNET telles que l'allocation dynamique de processus pour modéliser des circuits virtuels transitant par un commutateur ATM.

De plus, son support de langage Proto-C intégré lui permet de réaliser presque toutes les fonctions et tous les protocoles. OPNET offre un environnement de développement complet pour la spécification, la simulation et l'analyse des performances des réseaux de communication. Une large gamme de systèmes de communication d'un seul réseau local à des réseaux mondiaux par satellite peut être prise en charge. Des simulations d'évènements discrets sont utilisées comme moyen d'analyse de la performance et du comportement du système.

Les principales caractéristiques d'OPNET sont les suivantes :

1. Cycle de modélisation et de simulation : OPNET fournit des outils puissants pour aider les utilisateurs à parcourir trois des cinq phases dans un cercle de conception (c'est-à-dire la construction de modèles, l'exécution d'une simulation et l'analyse des données de sortie).
2. Modélisation hiérarchique : OPNET utilise une structure hiérarchique pour la modélisation. Chaque niveau de la hiérarchie décrit différents aspects du modèle complet en cours de simulation.
3. Spécialisé dans les réseaux de communication : Des modèles de bibliothèque détaillés fournissent un support aux protocoles existants et permettent aux chercheurs et aux développeurs de modifier ces modèles existants ou de développer de nouveaux modèles.
4. Génération automatique de simulation : les modèles OPNET peuvent être compilés en code exécutable. Une simulation exécutable d'évènements discrets peut être déboguée ou simplement exécutée, ce qui donne des données de sortie. Ce paquet sophistiqué est livré avec une gamme d'outils qui permettent aux développeurs de spécifier les modèles en détail, d'identifier les éléments du modèle d'intérêt, d'exécuter la simulation et d'analyser les données de sortie générées. Un schéma de simulation est illustré à la Figure 5.

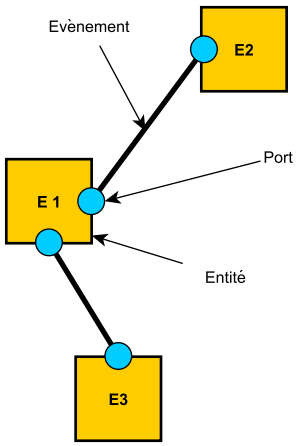


FIGURE 5 – Un schéma de simulation.

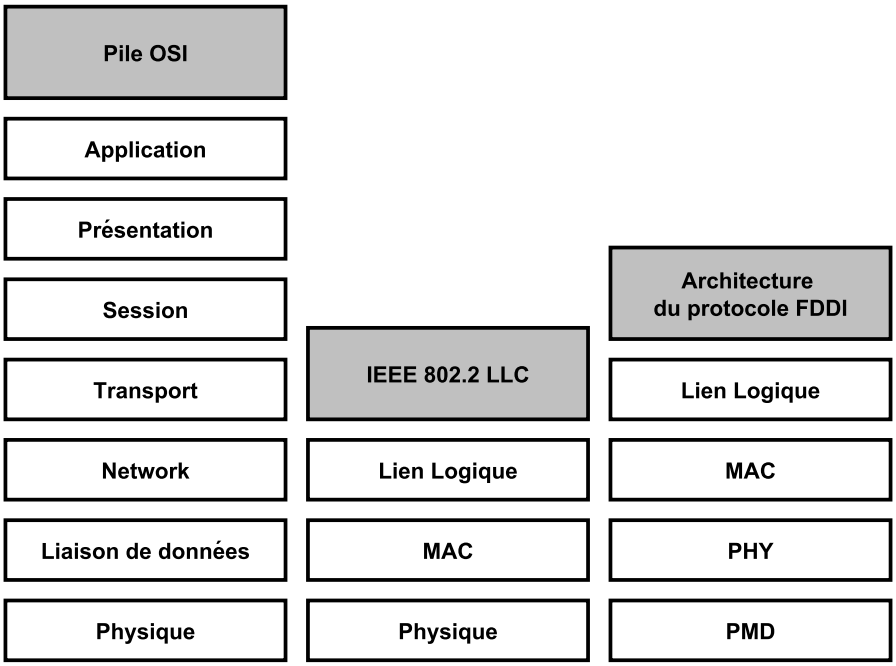


FIGURE 6 – Modèles et Architectures de référence.

3.1 OPNET : Un package de simulation de réseau

OPNET (Optimized Network Engineering Tools) a été largement utilisé dans les universités, l'industrie et le gouvernement. L'armée américaine a adopté l'OPNET comme norme sous les auspices de la Stratégie d'entreprise de l'Armée sous la direction du Bureau de l'armée américaine du Directeur des systèmes d'information pour le commandement, le contrôle, les communications et l'informatique. OPNET est largement utilisé dans les universités ainsi que de nombreuses parties du ministère de la Défense (DOD). OPNET prend en charge la visualisation à travers des objectifs de modélisation. Il peut être décrit comme un langage de simulation axé sur les communications. Le seul aspect le plus significatif d'OPNET est qu'il fournit l'accès direct au code source couplé à un front-end facile à utiliser.

Une approche générique de la modélisation du réseau peut être construite à partir du modèle de référence OSI, comme le montre la Figure 6. Cette approche permet la mise en œuvre de différents protocoles de réseau qui sont compatibles avec les limites de couche OSI. Sur le plan pédagogique, cette approche présente des limites. Comme l'illustre la Figure 6, toute implémentation détaillée d'un modèle Ethernet ne s'alignera pas directement Avec le modèle de référence OSI. D'autres protocoles tels que Fiber Distributed Data Interface (FDDI) ne s'alignent pas parfaitement avec le modèle de référence OSI.

Les modèles OPNET sont composés de trois couches de modèles hiérarchiques imbriqués primaires : la couche de processus, la couche de nœud et la couche de réseau. La couche de modélisation la plus basse est la couche de processus. Cette hiérarchie de modélisation est illustrée à la Figure 7. Le modèle de processus de la Figure 8 montre un diagramme de transition d'état (STD) pour la génération de paquets. Les modèles de processus sont construits à l'aide de machines à états finis (MFF) décrites par les MST.

Les FSM sont un moyen efficace de définir des systèmes à événements discrets qui maintiennent l'information d'état. La conception basée sur FSM permet de gérer la complexité. Les réseaux complexes peuvent être décomposés en états individuels, puis chaque état est défini et mis en œuvre.

Le code source de chaque état est facilement accessible et modifiable par l'utilisateur. Chaque état a des execs d'entrée et des execs de sortie. Le terme execs est utilisé pour décrire le code exécuté lorsqu'un état est entré et quand un état est sorti. Le code définissant la transition d'état entre les

Modèle des Réseaux	Réseaux et sous Réseaux
Modèle des nœuds	nœud individual et station
Modèle des processus	STD

FIGURE 7 – La hiérarchie du modèle OPNET.

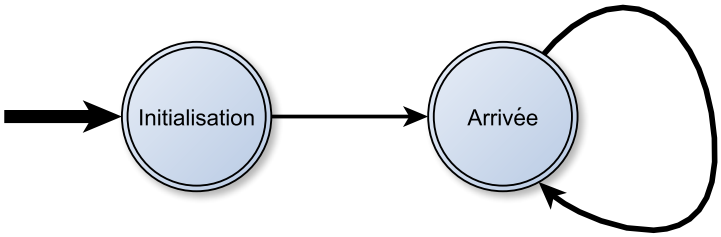


FIGURE 8 – Diagramme de transition d'état dans un modèle de processus.

états est également accessible. Le niveau d'abstraction suivant à partir du modèle de processus est le modèle de nœud. Chaque élément du modèle de nœud est soit un artéfact OPNET prédéfini, soit défini par sa propre STD. Un double clic sur un élément de modèle de nœud fait apparaître son modèle de processus sous-jacent. La Figure 9 est un exemple de modèle de nœud qui définit une station sur un réseau FDDI. Les paquets sont générés à partir de la source llc src, traités dans le module mac, et sont mis sur l'anneau par le module phy tx. Le trafic provenant de l'anneau est reçu via le module phy rx traité dans le module mac et finalement reçu et rejeté par le module d'évier llc.

Le cœur d'un modèle de nœud est soit un module de processeur, soit un module de file d'attente. Les modules de processeur sont utilisés pour effectuer le traitement général des paquets de données comme spécifié dans le protocole applicable. Les modules de file d'attente sont des super-ensembles de modules de processeur avec des capacités de collecte de données supplémentaires intégrées. Le module mac de la Figure 5.9 est une instanciation d'un module de file d'attente.

Le modèle de réseau est la couche de modélisation la plus élevée dans la hiérarchie du modèle OPNET. Le modèle de réseau peut représenter une hiérarchie de sous-réseaux. Un modèle de réseau est illustré à la Figure 10. Chacune des tations (nœuds) de la Figure 10 est définie par un modèle de nœud tel que celui de la Figure 9. De nouveau, chaque module dans un modèle de nœud est défini par un STD comme montré dans la Figure 8, ce qui correspond à la hiérarchie de modélisation

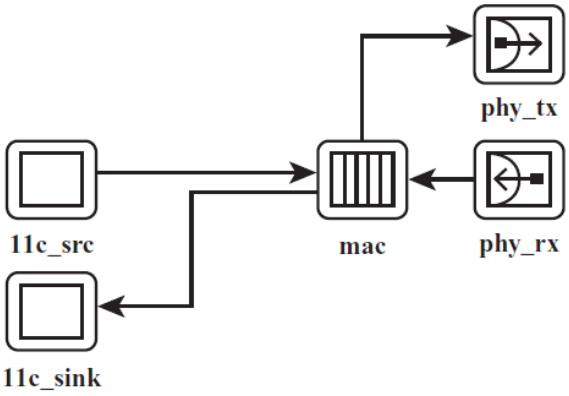


FIGURE 9 – Modèle de nœud (FDDI).

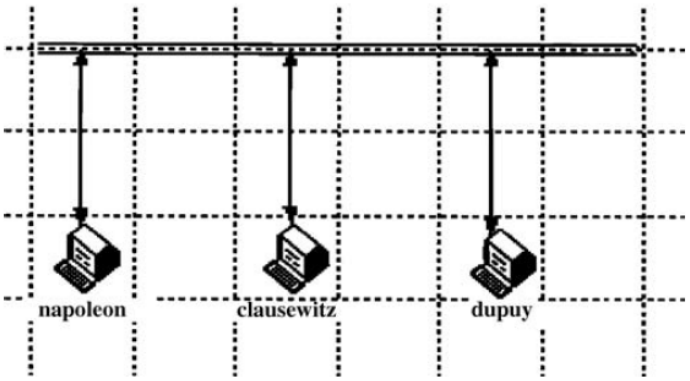


FIGURE 10 – Modèle de réseau à trois nœuds.

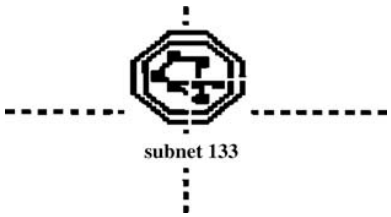


FIGURE 11 – Sous-réseau de segments agrégés.

illustrée à la Figure 7.

Le modèle de réseau peut être utilisé pour modéliser un réseau, un sous-réseau ou un segment unique ou une hiérarchie de réseaux, de travaux de sous-réseau ou de segments. Le segment de la Figure 10 peut être joint à d'autres segments et agrégé en une seule icône de sous-réseau, comme le montre la Figure 11. Le fonctionnement d'un seul segment de réseau peut maintenant être étudié. Les fonctionnalités implémentées des couches physique et de liaison du modèle de référence OSI sont suffisantes pour modéliser le fonctionnement d'un seul segment. A ce stade, les stations individuelles du segment peuvent être personnalisées si une représentation plus détaillée est souhaitée.

Les postes de travail individuels ou types de postes de travail peuvent être spécialement modélisés. Des caractéristiques particulières pourraient être mises en œuvre en modifiant les modules individuels de la station d'intérêt ou la ligne de réseau physique reliant les stations. De nombreuses modifications peuvent être effectuées via les menus intégrés. Toutefois, des modifications peuvent être apportées au niveau du code source si les choix de menu ne sont pas entièrement satisfaisants. L'ajout de services réseau dans un réseau TCP / IP nécessite la mise en œuvre d'Internet Protocole (IP). Pour simuler le fonctionnement de plus d'un segment, la fonctionnalité des services de couche réseau doit être ajoutée au modèle. Le modèle de nœud de la Figure 10 peut être étendu en ajoutant des modules pour implémenter le protocole IP et le protocole de résolution d'adresse (ARP). Les tables ARP sont implémentées statiquement avec des entrées correspondant à des adresses IP à des adresses MAC.

3.2 Exemple d'utilisation d'Opnet

3.2.1 Objectif

Utiliser [10], [11] OPNET pour analyser les performances du protocole RIP (Routing Information Protocol), un protocole de routage basé sur un algorithme de routage à vecteur de distance. Étudier la création et l'échange des tables de routage et analyser le comportement du RIP dans le cas d'une faute d'un lien de communication.

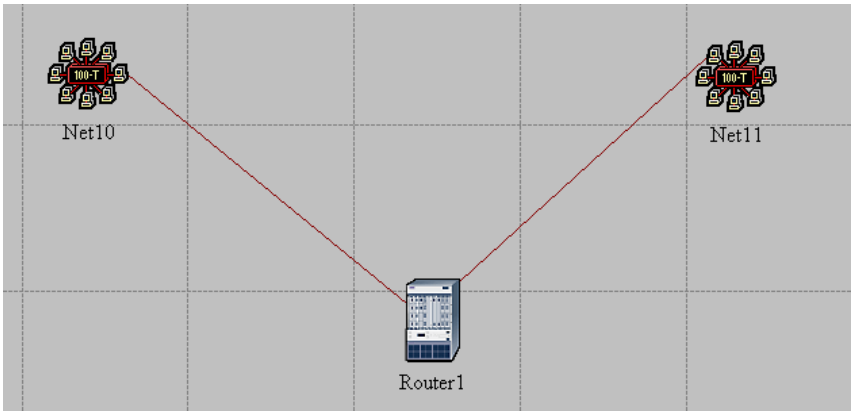


FIGURE 12 – Réseaux Opnet

3.2.2 Comment faire ?

❶ Créer un nouveau projet :

① Démarrer **Opnet IT Guru Academic Edition**.

② Choisissez **New** dans le menu **File**.

③ Sélectionnez **Project** et cliquez sur **OK**.

④ Nommez le projet et le scénario puis cliquez sur **OK**.

⑤ Dans la boîte de dialogue **Initial Topology dialog box**, Assurez-vous que **Create Empty Scenario** est sélectionné.

⑥ Cliquez **Next**.

⑦ Sélectionnez **Campus** à partir de **Network Scale list**.

⑧ Cliquez trois fois sur **Next**.

⑨ Cliquez **OK**.

❷ Créer et configurer le réseau :

- Initialiser le réseau :

① La boîte de dialogue **Object Palette** doit maintenant se trouver au-dessus de votre espace de travail de projet. Si ce n'est pas le cas, cliquez sur l'icône dans le menu. Assurez-vous que la case **Internet toolbox** est sélectionnée dans le menu déroulant de la **Object Palette**.

② Ajoutez à l'espace de travail du projet les objets suivants de la palette : un Routeur**ethernet-slip8-gtwyet** et deux objets **100BaseT-LAN**.
- Pour ajouter un objet à la palette, cliquez sur son icône dans la palette d'objets puis déplacez votre souris vers l'espace de travail, ensuite cliquez pour placer l'objet, enfin faites un clic droit pour cesser de créer des objets de ce type.

③ Utiliser des liaisons bidirectionnelles **100BaseT** pour connecter les objets. (voir la figure 12)

④ Pour renommer un noeud : Cliquez avec le bouton droit sur le nœud puis **Set Name**.

⑤ Fermez la boîte de dialogue **Object Palette**.

⑥ Enregistrez le projet.

- Configurer le routeur :

① Faites un clic droit sur **Router1**, puis **Edit Attributes** et enfin **Reports**.

② • **IP Forwarding Table = Export at End of Simulation :** pour demander au routeur d'exporter sa table de routage à la fin de la simulation dans le fichier de journalisation.

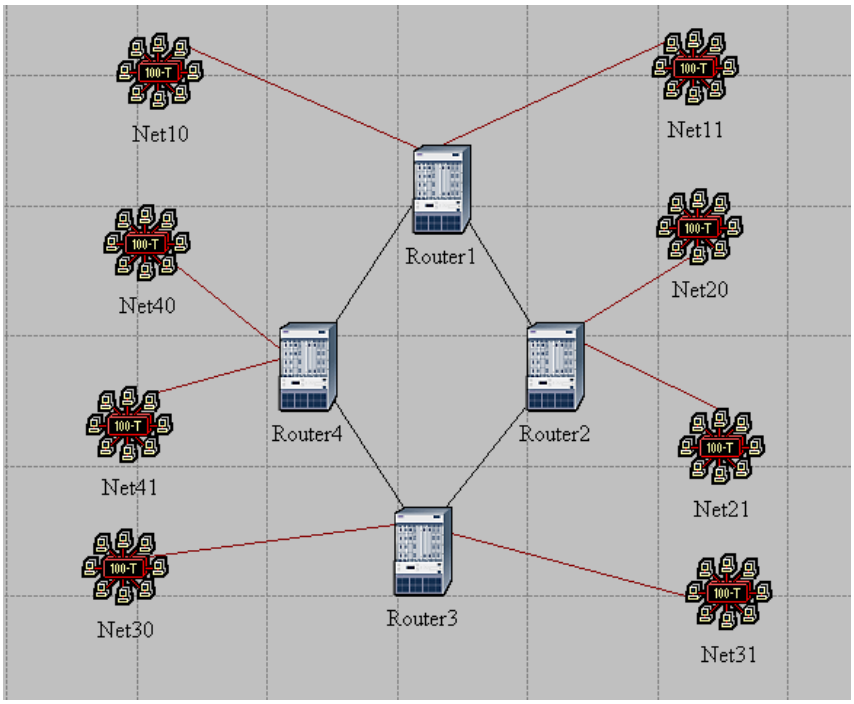


FIGURE 13 – Réseaux Opnet2

- **RIP Routing Table = Export at End of Simulation.**

③ Cliquez **OK** et Enregistrez le projet.

- Ajoutez les LAN restants :

① Sélectionnez simultanément les 5 objets que vous avez actuellement dans l'espace de travail du projet (un routeur, deux LAN et deux liens).

② Appuyez sur Ctrl + C pour copier les objets sélectionnés puis sur Ctrl + V pour les coller.

③ Répétez l'étape 2 trois fois pour générer trois nouvelles copies des objets et les disposer comme indiquer dans la figure 13. Utiliser les mêmes noms pour les différents objets.

④ Connectez les routeurs, en utilisant les liens **PPP-DS3** (ce lien à un débit de transfert de données égale à **44.736Mbps**.

- Le choix des statistiques :

① Cliquez avec le bouton droit n'importe où dans l'espace de travail du projet et sélectionnez **Choose Individual Statistics** dans le menu contextuel.

② Dans la boîte de dialogue Choisir les résultats, vérifiez les statistiques suivantes :

i. **Global Statistics** ⇒ **RIP** ⇒ **Traffic Sent (bits/sec)**.

ii. **Global Statistics** ⇒ **RIP** ⇒ **Traffic Received (bits/sec)**.

iii. **Nodes Statistics** ⇒ **Route Table** ⇒ **Total Number of Updates**.

③ Cliquez **OK**, puis enregistrez votre projet.

- Configurer la simulation :

① Cliquez sur **START** pour avoir la fenêtre **Configure Simulation**.

② Définissez la durée de la simulation à 10 minutes.

③ Sur le panneau de droite, cliquez sur l'onglet **Input** ⇒ **Global Attributes** et modifiez les attributs suivants :

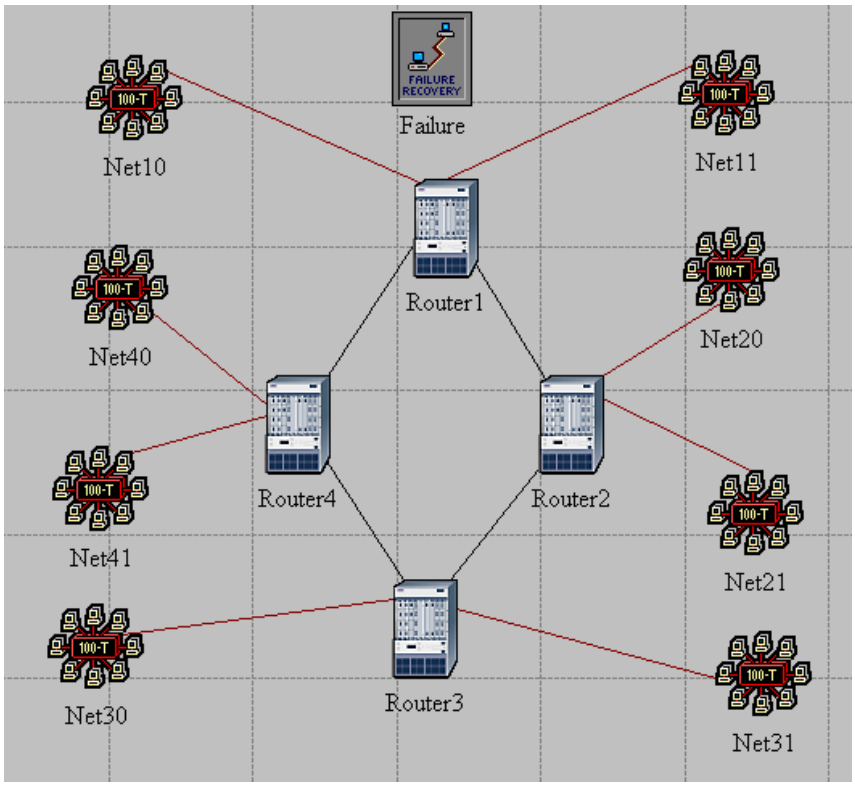


FIGURE 14 – Réseaux Opnet

- **IP Dynamic Routing protocol = RIP**, Cela configure le protocole RIP comme étant le protocole de routage de tous les routeurs du réseau.
- **IP Interface Addressing Mode = Auto Addressed/Export**.
- **RIP Sim Efficiency = Disabled** : Si cet attribut est activé, RIP s'arrêtera après le "RIP Stop Time". Mais nous avons besoin du RIP pour maintenir la mise à jour de la table de routage au cas où il y aurait un changement dans le réseau.

④ Cliquez **OK**, puis enregistrez votre projet.

Auto Addressed signifie que toutes les interfaces IP font automatiquement attribuées des adresses IP pendant la simulation. La classe d'adresses (A, B ou C) est déterminée sur la base du nombre d'hôtes dans le réseau conçu. Les masques de sous-réseau attribués à ces interfaces sont les masques de sous-réseau par défaut pour cette classe.
Export provoque l'exportation des interfaces IP attribuées automatiquement dans un fichier nommé $< net - name > ip - address.gdf$ qui est enregistré dans le répertoire du modèle principal.

- Un deuxième scénario : Dans le réseau que nous venons de créer, les routeurs vont construire leurs tables de routage, puis ils n'auront plus besoin de les mettre à jour car nous n'avons pas simulé de fautes de nœud ou de lien. Dans ce second scénario, nous simulerons des fautes afin que nous puissions comparer le comportement des routeurs dans les deux cas.

① Sélectionnez **Duplicate Scenario** dans le menu **Scenarios** et nommez-le **Failure** ⇒ Click **OK**.

② Ouvrez la palette d'objets en cliquant sur l'icône. Sélectionnez la palette **Utilities** dans le menu déroulant.

③ Ajoutez **Failure Recovery** à votre espace de travail et faites-le échouer comme indiqué ⇒ Fermez la boîte de dialogue Palette d'objets.

④ Cliquez avec le bouton droit de la souris sur l'objet **Failure** ⇒ **Edit Attributes** ⇒ Cliquez

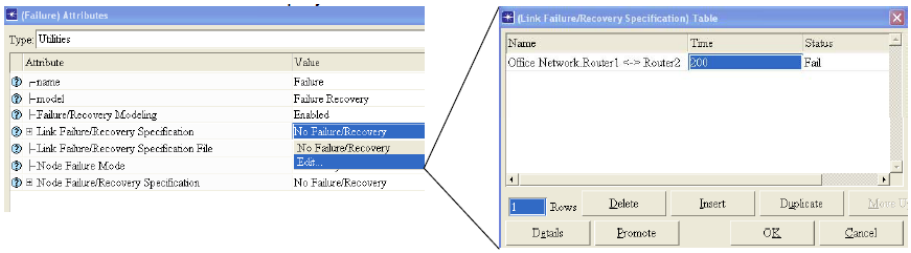


FIGURE 15 – Réseaux Opnet

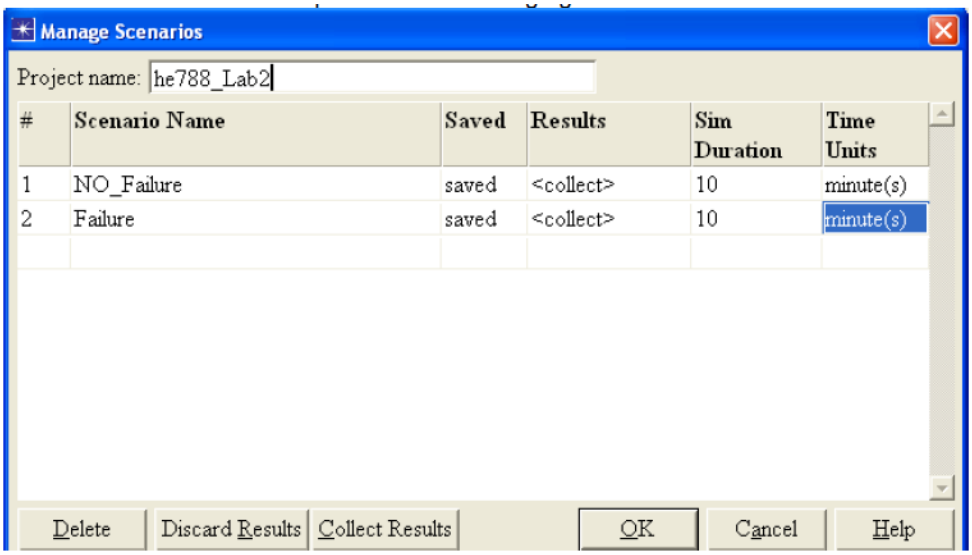


FIGURE 16 – Réseaux Opnet

avec le bouton droit sur la hiérarchie **Link Failure/Recovery Specification** ⇒ Sélectionnez **Edit** ⇒ Mettre 1 dans les rangées, comme indiqué dans la figure ci-dessous : Cela va "faire échouer" le lien entre **Router1** et **Router2** 200 secondes durant la simulation.

⑤ Cliquez **OK**, puis enregistrez votre projet.

- Exécution de la simulation : Pour exécuter la simulation pour les deux scénarios simultanément :

① Allez dans le menu **Scenarios** ⇒ Sélectionnez **Manage Scenarios**.

② Modifiez les valeurs sous la colonne **Results** à <**collect**> (ou <**recollect**>) pour les deux scénarios. Comparer à la figure suivante..

③ Cliquez sur **OK** pour exécuter les deux simulations.

④ Une fois les deux simulations terminées, une pour chaque scénario, cliquez sur **Close** ⇒ Enregistrer votre projet.

- Visualisation des résultats : Comparaison des nombres des mises à jour :

① Sélectionnez **Compare Results** dans le menu **Results**.

② Modifiez le menu déroulant dans la partie inférieure droite de la boîte de dialogue Comparer les résultats avec **Stacked Statistics** comme indiqué.

③ Sélectionnez les statistiques de Nombre total de mises à jour pour le **routeur1 Total Number of Updates**et cliquez sur **Show**.

④ Vous devriez obtenir deux graphiques, un pour chaque scénario. Cliquez avec le bouton droit sur chacun d'eux et sélectionnez **Draw Style** ⇒ **Bar**.

- Obtenir les adresses IP : Avant de vérifier le contenu des tables de routage, nous devons déterminer les informations d'adresse IP pour toutes les interfaces du réseau actuel. Rappelons que ces adresses IP sont attribuées automatiquement pendant la simulation, et nous définissons l'attribut global *IP Addressing Mode Interface* pour exporter ces informations vers un fichier.
  ① Dans le menu **File**, choisissez **Model Files** ⇒ **Refresh Model Directories**. Cela amène Opnet IT Guru à rechercher les répertoires de modèle et à mettre à jour la liste des fichiers.
  ② Dans le menu **File**, choisissez **Open** ⇒ dans le menu déroulant, choisissez **Generic Data File** ⇒ sélectionnez le fichier **Failureip- addresses** (l'autre fichier créé à partir du scénario d'échec doit contenir les mêmes informations)
  ③ cliquez sur **OK**.
  ④ Le fichier **gdf** doit afficher les adresses IP attribuées à l'interface de **Router1** dans notre réseau. Le masque de sous-réseau associé à chaque interface indique que l'adresse du sous-réseau au quelle l'interface est connectée.
  ⑤ Dessinez la topologie du réseau et notez les adresses IP associées à **Router1** ainsi que les adresses affectées à chaque sous-réseau.
- Comparer le contenu des tables de routage :
  ① Pour vérifier le contenu des tables de routage dans **Router1** pour les deux scénarios : Aller au menu **Results** ⇒ Ouvrir le journal de simulation **Open Simulation Log** ⇒ ? Développer la hiérarchie à gauche comme illustré ci-dessous ⇒ cliquer sur le champ **COMMON ROUTE TABLE**.
  ② Effectuer l'étape précédente pour les deux scénarios.

Références

[1] Mohsen Guizani, Ammar Rayes, Bilal Khan, and Ala Al-Fuqaha. *Network modeling and simulation : a practical perspective*. John Wiley & Sons, 2010.

[2] Adarshpal S Sethi and Vasil Y Hnatyshin. *The Practical OPNET User Guide for Computer Network Simulation*. CRC Press, 2012.

[3] Mahbub Hassan and Raj Jain. *High performance TCP/IP networking*, volume 29. Prentice Hall, 2003.

[4] Georges Fiche and Gérard Hèbuterne. *Trafic et performances des réseaux de télécoms*. Hermès science publ., 2003.

[5] William Stallings. *High-speed networks and internets : performance and quality of service*. Pearson Education India, 2002.

[6] Halsall Fred. Data communications, computer networks and open systems, 1996.

[7] Andrew S Tanenbaum et al. Computer networks, 4-th edition. *ed : Prentice Hall*, 2003.

[8] Wikipedia, url = http ://fr.wikipedia.org.

[9] Nayak Ajit Kumar Rai Satyananda Champati and Mall Rajib. Computer network simulation using ns2.

[10] Adarshpal S Sethi and Vasil Y Hnatyshin. *Laboratory Assignment : Network Simulation using OPNET*. City University, School of Engineering and Mathematical Sciences, 2006.

[11] Emad Aboelela. *Network simulation experiments manual*. Academic Press, 2003.