الجمهوريـة الجزائـريـة الديمقراطيـة الشعبيـة
République Algérienne Démocratique et Populaire
وزارة التعليـم العالـي والبحـث العلمـي
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

N° Réf :……

Centre Universitaire
Abd elhafid Boussouf Mila

Institut des sciences et de la technologie **Département de Mathématiques et Informatiques**

Mémoire préparé en vue de l'obtention du diplôme de Master

En : Informatique

Spécialité: Sciences et Technologies de l'Information et de la Communication (STIC)

Conception et réalisation d'une extension au mécanisme de multi-streaming du protocole SCTP sous NS3

Préparé par : **- KHELATOU Amel**
- CHEBBAH Asma

Soutenue devant le jury

Encadré par : Meriem TALAI……...M . A. A
Président : Samira ABDERREZAK ……. M . A. A
Examinateur : Meriem BOUMASSATA ……M . A. A

Année universitaire : 2015/2016

بسم الله الرحمن الرحيم

REMERCIMENT

اللّهم صلّ وسلم على سيدنا مُحمّد أشرف المرسلين وعلى اله و صحبه و سلّم

الحمد لله الذي وفقنا فيما نحب ويرضاه وكتب لنا فيه الخير الكثير .

Nous tenons particulièrement à remercier Allah le tout puissant de nous avoir donnés courage et force pour réaliser ce mémoire, qui n'aurait jamais été réalisé sans sa bénédiction.

Nous tenons remercier aussi nos parents, qui ont toujours tout mis en œuvre pour qu'on s'épanouisse dans tous ce qu'on entreprend.

Nous adressons nos remerciements à notre encadreur Madame Talai meriem, pour son aide consistante, ses conseils judicieux, et pour ses remarques objectives.

Nous remercions les membres du jury, Melle Boumassata Meriem et Mme Abderrezak samira d'avoir accepté d'examiner notre travail.

Nous n'oublions pas tous les enseignants de l'informatique de notre centre universitaire, surtouts Mr Bencheikh Elhosine Madjed, Mr Boukhchame Nadir et Mme Bouchmel Nerdjes pour leurs motivations durant nos études universitaires.

Dédicace

Je dédie cette Mémoire …✍

– A ma très chère mère « fatima », mon très cher père «filali »
Pour leur soutien, leur patience et leur amour.

– A ma chère frère « Samir ».

– A mes précieuses sœurs « Houda », « Karima » et « Hala ».

– A mon binôme «Asma» et sa famille.

– A mes amies et mes sueurs « Amira, Samira et Meriem».

– A tous mes amies **« »** .

– A toutes la promotion de l'informatique.

– A tous ce qui m'aime et que j'aime.

Amel

Dédicace

Je dédie cette Mémoire …✍

– A ma très chère mère « Saliha », mon très cher père «habid »
Pour leur soutien, leur patience et leur amour.

– A mes chères frères
« Bakir»,« Ramzi» ,« Mostafa » et « Ihab ».

– A ma précieuse sœurs « Hadil».

– A tout ma grande famille.

– A mon binôme «Amel» et sa famille.

– A mes amies et mes sueurs « Amira, Samira et Meriem».

– A tous mes amies **« »**.

– A toutes la promotion de l'informatique.

– A tous ce qui m'aime et que j'aime.

Asma

Résumé

En télécommunication, Le transport de signalisation du RTC (Réseau Téléphonique Commuté) sur un réseau IP (Internet Protocol) et en utilisant TCP (Transmission Control Protocol) met en évidence des problèmes tel que le HOL (Head-Of-Line) blocking. Le nouveau protocole de transport SCTP (Stream Control Transmission Protocol) est venu avec deux caractéristiques spéciales (multi-streaming et multi-homing) pour résoudre ces problèmes. Nous nous intéressons au mécanisme de multi-streaming.

Dans ce mémoire nous avons essayé d'élaborer une conception et réalisation d'une extension au multi-streaming du protocole SCTP sous NS2 (Network Simulator 2).Nous avons activé le processus de multi-streaming au niveau du SAP (Service Access Point) entre couche application et transport de SCTP. Nous avons également modifié l'algorithme qui permet à l'application de classer les données afin d'utiliser le multi-streaming de SCTP. Notre objectif principal est d'optimiser le transfert de données les plus contraignantes dans les réseaux.

Les performances de notre extension sont évaluées et comparées avec le protocole originel sous le simulateur NS2.

Mots clés: Couche transport, SCTP, Multi-streaming, Ordonnancement, Simulation, NS2.

Table des matières

Table des matières

Chapitre 1: Introduction à l'architecture des réseaux.

Chapitre 2: Outils de simulation.

Chapitre 3: Extension au multi-streaming de SCTP.

Chapitre 4: Simulation et analyse des résultats.

Tables des illustrations

1. Table des figures

2. Table des tableaux

Table des abréviations

A	ACK	Acknowledgement.
	API	Application Programming Interface.
	A_Rwnd	Advertised Receiver Window Credit.
	AWK	Aho Weinberger Kernighan.
C	CSMA	Carrier Sense Multiple Access.
	CWND	Congestion WiNDow.
D	DARPA	Defense Advanced Research Projects Agency.
	DoS	Denial of Service.
F	FIN	FINal.
	FCFS	First Come First Served.
	FTP	File Transfer Protocol.
G	GUI	Graphic User Interface.
H	HOL	Head-Of-Line.
	HTML	Hypertext Markup Language.
I	IETF	Internet Engineering Task Force.
	IP	Internet Protocol.
	IPv4	Internet Protocol version 4.
	IPv6	Internet Protocol version 6.
	ISO	International Standardization Organization.
M	MTU	Maximum Transmission Unit.
N	NAM	Network AniMator.
	NS	Network Simulator.
	NS2	Network Simulator 2.
	NS3	Network Simulator 3.
O	OPNET	Optimum Network Performance.
	OTcl	Object Tools command language.
	OSI	Open System Interconnection.

P	Perl	Practical Extraction and Report Language.
	PMTU	Path MaximumTransmission Unit.
	PSTN	Public Switched Telephone Network.
	PSH	PuSH.
Q	QoS	Quality of Service.
R	RFC	Request For Comment.
	RR	Round Robin.
	RST	ReSeT.
	RTC	Réseau Téléphonique Commuté.
	RTO	Retransmission Time Out.
	RTT	Round Trip Time.
S	SAP	Service Access Point.
	SACK	Selective Acknowledgement.
	SCTP	Stream Control Transmission Protocol.
	SID	Stream IDentifier.
	SIGTRAN	SIGnaling TRANsport.
	SSN	Stream Sequence Number.
	SSTHRESH	Slow Start Threshold.
	SS7	Système de Signalisation numéro 7.
	SYN	Synchronization.
T	Tclcl	Tcl classes.
	TCP	Transmission Control Protocol.
	TSN	Transmit Sequence Number.
U	UDP	User Datagram Protocol.
	UMTS	Universal Mobile Telecommunications System.
	UPL	UPper Layer.
	URG	URGent.
V	VoIP	Voice over IP.
W	WiFi	Wireless Fidelity.

Introduction générale

Les réseaux de télécommunication modernes dépendent fortement de l'échange rapide et fiable de leurs paramètres de fonctionnement entre les différentes entités. En effet, la signalisation ou l'échange de message de contrôle supporte des services de télécommunication de base. Mais de plus, les critères de qualité de service sont également fortement influencés par la performance du système de signalisation [1].

Pendant beaucoup d'années, SS7 (le système de signalisation numéro 7) a été le porteur dominant de la signalisation pour des réseaux de télécommunications. Il a l'inconvénient d'exiger une infrastructure de réseau dédiée et très chère [1].

En 1998, un groupe de travail SIGTRAN (SIGnaling TRANsport) de l'IETF (Internet Engineering Task Force »[2] a été formé pour concevoir un mécanisme pour le transport fiable de la signalisation d'appel sur Internet (via TCP).

Deux problèmes majeurs apparus dans l'utilisation de TCP: head-of-line-blocking où l'envoi de messages indépendants sur une connexion TCP préservant l'ordre est retardé dans les tampons d'un récepteur jusqu'à ce qu'un message perdu soit retransmis et reçu. Le 2ème problème c'est l'adressage de connexion TCP basé sur une adresse unique IP. Une connexion TCP ne se lie qu'à un point de fixation unique, alors que des chemins alternatifs peuvent être utilisés avec d'autres adresses IP pour communiquer des messages critiques [3].

Le protocole SCTP (Stream Control Transmission Protocol) [RFC2960] (Request for comment.) est une solution standard à ces problèmes, établi en 2000 par le groupe SIGTRAN [1]. Il permet l'utilisation commune du cœur d'un réseau pour le transport de signalisation et de données.

SCTP est un protocole de transport caractérisé par deux nouveaux mécanismes. Le mécanisme de multi-homing permet d'utiliser plusieurs adresses IP pendant une association. Le mécanisme multi-streaming permet d'utiliser plusieurs streams indépendants de messages pour la transmission de données.

Problématique

Sur Internet, des recherches montrent que le trafic est très différent par sa composition ou par les contraintes exigées : En plus des messages de signalisation, les données utilisateurs de type audio ou vidéo nécessitent des traitements particuliers surtout en termes de délais de transmission pour éviter des problèmes bien connus tels que la gigue (la variation du temps de transmission des paquets de données).SCTP, par ces nouveaux mécanismes, est alors considéré comme protocole intéressant pour pallier aux contraintes des messages applicatifs.

Cependant, dans ses spécifications théoriques et encore plus dans son implémentation sous NS2, le protocole sollicite des recherches scientifiques et expérimentations pour l'améliorer. Nous nous intéressons au mécanisme de multi-streaming dont l'algorithme de transmission est encore à l'étude et montre des opportunités importantes en matière de gestion de la qualité de service ou QoS (Quality of Service) des transmissions pour les différents types de données.

À titre indicatif, le multi-streaming dans les spécifications théoriques de SCTP ne fait aucune distinction entre type de données applicatives alors que sous NS2, il est implémenté séparément pour traiter les données applicatives de FTP et les autres types de données. De plus, la gestion de l'ordonnancement entre les streams reste précaire pour les types d'application à contraintes.

Nous notons alors que la distinction entre les types de données FTP et autres n'est basé sur aucun principe de la qualité de service, encore moins la gestion du multi-streaming pour le transfert des données de SCTP à la couche réseau, au niveau de l'extrémité émettrice ou dans la réception et envoi des données de SCTP à l'application de l'autre extrémité.

Objectifs et contribution

Dans notre travail présenté par ce rapport, nous nous intéressons à l'étude et la compréhension du protocole de la couche transport SCTP, en particulier à son mécanisme de multi-streaming. L'objectif étant d'apporter des extensions à ce mécanisme afin de supporter la qualité de service des transmissions.

Nous opérons sur deux axes : la distinction entre les types de données et le traitement de leur transmission. Dans la conception et la réalisation de notre travail, nos propositions sont faites à un niveau théorique mais surtout de sorte à les intégrer cohére ment au code de l'agent SCTP. Nous pouvons considérer cette extension comme un terrain de base pour l'implémentation de la QOS via SCTP sous NS2.

Organisation du mémoire

Notre rapport de mémoire est réparti sur quatre chapitres présentant l'enchaînement suivi pour résoudre la problématique posée ci-dessus, au moyen des contributions proposées. Les principaux chapitres sont énumérés ci-dessous.

- **Chapitre 1:** Nous présentons dans ce chapitre un aperçu général sur l'architecture des réseaux leurs concepts et leurs caractéristiques. Nous avons mis l'accent sur la couche transport en présentant les deux protocoles TCP et SCTP.
- **Chapitre 2 :** Ce chapitre est réservé à la présentation de différents simulateurs des réseaux en particulier le simulateur NS3 et NS2 puis une étude détaillé sur le simulateur NS2, ces composants et leur fonctionnement.
- **Chapitre 3 :** Dans ce chapitre, nous détaillons les différentes conceptions existantes des transmissions parallèles. Ensuite nous présentons notre conception de l'extension.
- **Chapitre 4 :** Nous présentons, dans ce dernier chapitre, la réalisation de l'extension par les modifications au code C++ de NS, la simulation utilisée et les résultats obtenus par simulation de l'extension proposée sous la plateforme NS2.

Chapitre 1

Introduction à l'architecture des réseaux

1. Généralités

Du fait du grand nombre de fonctionnalités implémentées dans les réseaux, les architectes réseau ont décomposé les processus réseaux en couches protocolaires ou niveaux, portant chacune son propre nom qui caractérise ses fonctionnalités dans le transfert de données à travers le réseau [4]. L'ensemble des couches et leurs interactions constituent des modèles de référence : le modèle OSI (Open System Interconnexion) dont dérive le modèle TCP /IP (Transmission Control Protocol / Internet Protocol).

Chaque couche va rendre des services à la couche immédiatement supérieure et utiliser les services de la couche immédiatement inférieure. Un service désigne la fonctionnalité qui doit être rendu par la couche N à la couche supérieure (N+1) [4].

2. Modèles OSI et TCP /IP

2.1.Modèle OSI

Le modèle OSI a été développé par l'organisation ISO (International Standardisation Organisation) pour la normalisation en 1984 [5]. Les fonctionnalités sur ses couches se résument comme suite :

Couche physique : cette couche s'occupe de normaliser les caractéristiques électriques, mécaniques, et les caractéristiques fonctionnelles des circuits de données et les procédures d'établissement, de maintien et de libération du circuit de données.

Couche liaison de données : la fonction principale de cette couche est la délimitation des paquets, lors de l'arrivée des éléments binaires, le récepteur ayant le pouvoir de connaître le début et la fin de la trame, d'autres fonctions comme la correction d'erreurs, la synchronisation … etc. sont assurés par cette couche.

Couche réseau : c'est la couche qui permet d'acheminer les données vers le récepteur en choisissant le bon chemin, l'autre fonction est l'interconnexion des différents réseaux.

Couche transport : est chargée de la segmentation des données applications en paquets (datagrammes) adaptés pour la transmission jusqu'au destinataire sur les infrastructures réseau, ainsi elle procure des mécanismes permettant le contrôles de la quantité des données injectées dans le réseau.

Couche session : cette couche organise et synchronise les échanges entre tâches distantes. Elle établit également une liaison entre deux programmes d'application devant coopérer et commande leur dialogue.

Couche présentation : cette couche assure l'unicité de langage c'est-à-dire que l'information envoyée par la couche application d'un système est lisible par la couche application d'un autre système.

Couche application: Cette couche est le point de contact entre l'utilisateur et le réseau, elle offre à l'utilisateur les différents services apporter par le réseau, comme par exemple le transfert de fichier, la messagerie...etc [4].

2.2. Modèle TCP/IP

Quant au modèle TCP/IP, acronyme de Transmission Control Protocol / Internet Protocol, il définit les couches et règles de communications sur Internet [6]. Ces couches sont plus ou moins différentes en nombres et services fournis, mais sont basés sur le modèle OSI.

Couche accès réseau: regroupe les services des couches physiques et liaison de données d'OSI. Elle permet à plusieurs types de réseaux de s'intégrer à l'architecture globale TCP/IP.

Couche Internet : cette couche assure principalement la fonction d'adressages et de routage entre les différents sous-réseaux.

Couche transport : sa fonction se résume dans l'établissement et la fermeture des connexions, résoudre les problèmes liées aux transmissions des paquets dans les niveaux inférieurs tel la perte, la duplication et les erreurs sur les paquets.

Couche application : le modèle TCP /IP regroupe en une seule couche tous les protocoles de haut niveau de OSI, relatifs au contrôle de dialogue [4].

2.3. Mécanisme d'encapsulation

Le modèle TCP/IP utilise lors du transfert des données un mécanisme d'emboîtement des messages les uns dans les autres, appelé l'encapsulation.

La couche **N** de l'ordinateur source communique avec sa couche homologue **N** de l'ordinateur de destination suivant des règles et conventions utilisées appelées Protocole de couche N.

Lors de l'émission, les données traversent toutes les couches, chacune ajoute aux données des informations relatives au protocole (en-têtes, queues, …) et puis les retransmet. Ces ajouts correspondent à l'opération d'encapsulation.

En réception, l'opération est effectuée à rebours, le protocole de chaque couche de destination rétablit la forme originale des informations c'est la dés-encapsulation.

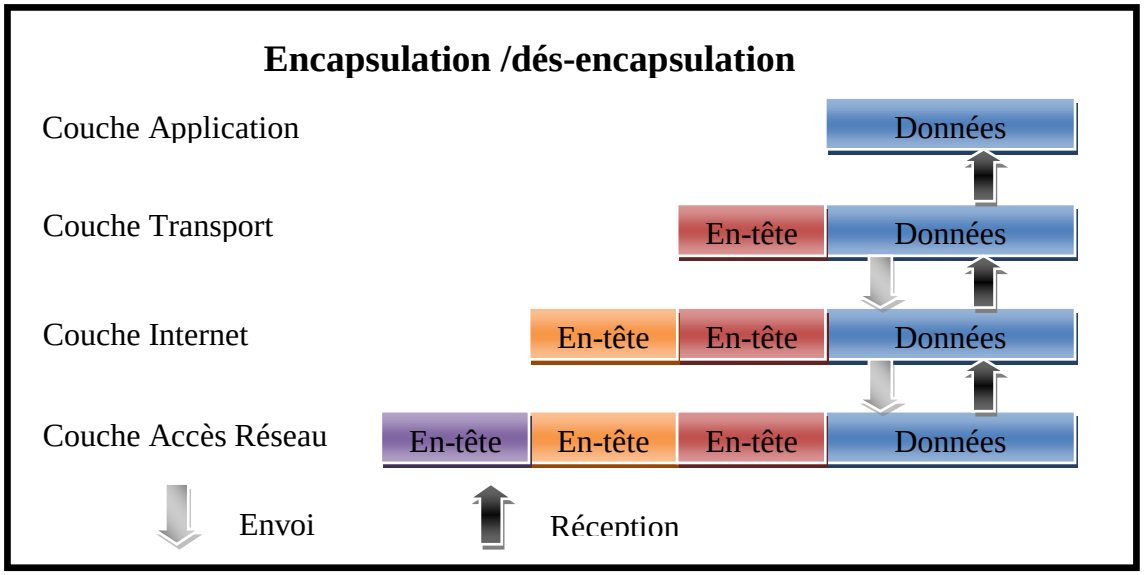


Figure 1.1: Encapsulation de données.

Notons que des points d'accès au service N ou SAP (Service Access Point) sont situés à la frontière entre les couche N+1 et N [1].

Parmi les couches présentées précédemment on s'intéresse dans ce qui suit d'avantage à la couche transport.

3. Principes de la couche transport

Selon les modèles cités, un réseau est organisé d'une manière hiérarchique, où chaque couche assure un aspect spécifique de la communication à travers les protocoles déployés.

Les protocoles usuels de transport de l'information dans les réseaux IP, sont TCP et UDP (User Datagram Protocol). SCTP est relativement un nouveau protocole de transport qui présente des mécanismes particuliers pour la couche transport. Dans ce qui suit, on s'intéresse aux protocoles TCP et SCTP. UDP n'est cité qu'à titre comparatif.

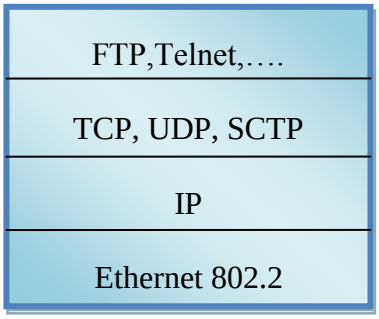


Figure 1.2:SCTP, TCP dans la pile IP.

3.1. Communication bout en bout

La couche transport a pour but de fournir des solutions de bout-en-bout pour le transfert des données depuis une application émettrice vers une application réceptrice : contrairement aux couches inférieures dont les protocoles opèrent entre les nœuds immédiatement adjacents, [7].

Elle joue ainsi un rôle intermédiaire critique entre les couches basses (1,2,3) et les couches hautes (5,6,7) du modèle OSI. En effet, elle récupère et traite les détails d'implémentation des sous-réseaux mais elle les masque aux couches hautes.

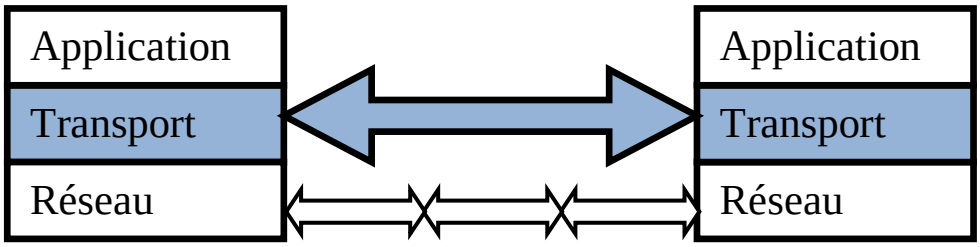


Figure 1.3: Communication Transport de bout à bout.

Les extrémités communicantes via le protocole de transport sont considérées respectivement client/serveur selon l'initiative/acceptation de la connexion transport.

3.2. Identification de connexion transport (adresse IP, port)

Les adresses IP désignent les machines sur le réseau entre lesquelles d'éventuels processus communications sont établis. Lorsqu'un processus, d'une machine identifiée par son adresse IP, désire entrer en communication avec le processus d'une autre machine, il doit adresser le processus s'exécutant sur cette machine. L'adressage de ce processus est effectué selon un concept abstrait, c'est le concept du port [8].

C'est pourquoi, dans le but d'échanger des informations entre deux hôtes d'un réseau**,** une connexion de transport lie une paire d'extrémités identifiée chacune par un couple (adresse IP, port). La combinaison adresse IP + port est alors une adresse unique, elle est appelée socket. Le quadruple « adresse source, N° de port », « adresse destination, N° de port » définie la connexion entre les deux extrémités utilisant un protocole transport.

Les valeurs des ports étant limitées et numériques, on en distingue 3 catégories : les ports réservés (de 0 à1023), des ports inscrits (de 1024 à 49151) et des ports dynamique ou privé (de 49152 à 65535), se référé à [9] pour plus de précisions.

3.3. Services transport

Les principaux services fournis par la couche transport se résument en les points suivants. Plus de détails sont apportés dans la présentation des protocoles TCP et SCTP :

- **Segmentation des messages :** une opération de division des messages longs provenant de la couche application en paquets de taille moindre et acceptable au niveau réseau.
- **Regroupement des messages :** lors de la phase d'encapsulation, les messages courts reçus de l'application sont regroupés en un seul paquet à transmettre à la couche réseau.
- **Reconstitution du message d'origine:** lors la phase de dés- encapsulation, les paquets reçus de la couche inférieure sont rassemblés en un seul message transmis à la couche applicative.
- **Acquittements ACK (Acknowledgments) :** des accusés de réception sont utilisés pour valider et confirmer la bonne réception des paquets entre les bouts communicants. Ils peuvent être immédiats ou cumulatifs.
- **Contrôle d'erreurs :** en utilisant le mécanisme de checksum pour chaque segment.

4. Protocole TCP (Transmission Control Protocol)

4.1. Introduction

Le protocole TCP est sans doute considéré comme le plus important des protocoles au niveau transport, regroupant les fonctionnalités de niveau 4 du modèle de référence. Le protocole TCP est en mode orienté connexion. TCP a été développé pour assurer des communications fiables entre deux hôtes sur un même réseau physique, ou sur des réseaux différents [10].

Notons que même si UDP se positionne également au niveau transport, il opère en un mode non connecté et avec pratiquement aucune fonctionnalité. Il est non fiable car son utilisation présuppose que l'on n'a pas besoin ni du contrôle de flux, ni de la conservation de l'ordre de remise des paquets. Dans la suite, on ne s'intéresse qu'au protocole TCP.

4.2. Format d'un paquet TCP

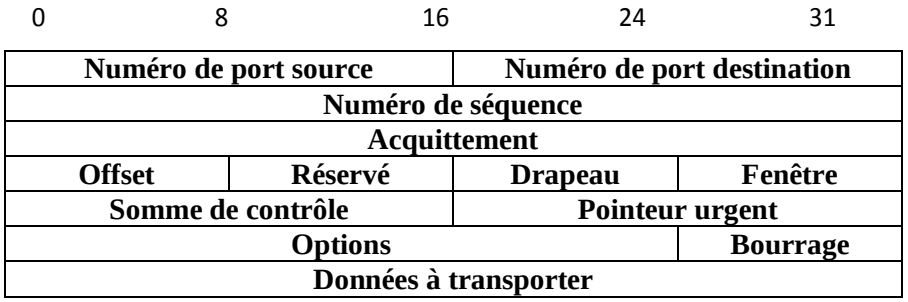


Figure 1.4: Format d'un paquet TCP.

- **Numéro du port source:** sur16 bits, ce numéro correspond au point de communication utilisé par le service de la couche application de l'émetteur.
- **Numéro du port destination:** sur16 bits, il correspond au point de communication utilisé par le service de la couche application du destinataire.
- **Numéro de séquence:** permet de rétablir l'ordre des paquets reçus et d'écarter les paquets dupliqués. Ce numéro est incrémenté d'une unité chaque fois qu'un octet est envoyé.
- **Acquittement :** Si le flag ACK est présent, ce champ désigne le prochain numéro de séquence attendu. Il constitue donc un acquittement de tous les segments dont le numéro de séquence est inférieur.
- **Offset données :** champ de 4 bits qui indique le nombre de mots de 32 bits dans l'en-tête TCP.
- **Réservé :** Cette zone de 6 bits est toujours à zéro.
- **Drapeau :** Cette zone contient sous forme binaire un drapeau concernant l'association ou le segment lui-même (URG , ACK, PSH, RST, SYN, FIN) dont la signification est détaillé en [10].
- **Fenêtre :** Nombre d'octets disponibles dans la fenêtre de réception, c'est-à-dire le nombre d'octets qui peuvent être reçus avant acquittement.
- **Somme de contrôle :** La somme de contrôle est réalisée en faisant la somme des champs de données de l'en-tête, afin de pouvoir vérifier l'intégrité de l'en-tête [11].
- **Pointeur urgent :** Champ valide si le drapeau URG est positionné. Pointeur sur l'octet de donnée urgente, exprimé en déplacement par rapport au numéro de séquence du segment.
- **Options :** Facultative.

- **Bourrage :** On remplit l'espace restant après les options avec des zéros pour avoir une longueur multiple de 32 bits.

- **Données :** Ce sont les données à acheminer, en provenance ou à destination de la couche supérieure de l'émetteur et encapsulées par des segments TCP [10].

4.3. Cycle des sessions TCP

Une session TCP fonctionne en trois phases :

4.3.1. Établissement d'une connexion

Pour ouvrir une connexion, le mécanisme le plus commun est l'ouverture passive où un système ouvre un socket et se met en attente passive de demandes de connexion d'un autre système. Ce fonctionnement est utilisé par le côté serveur de la connexion. Le côté client de la connexion effectue une ouverture active en 3 temps, comme indiqué sur la figure ci-après.

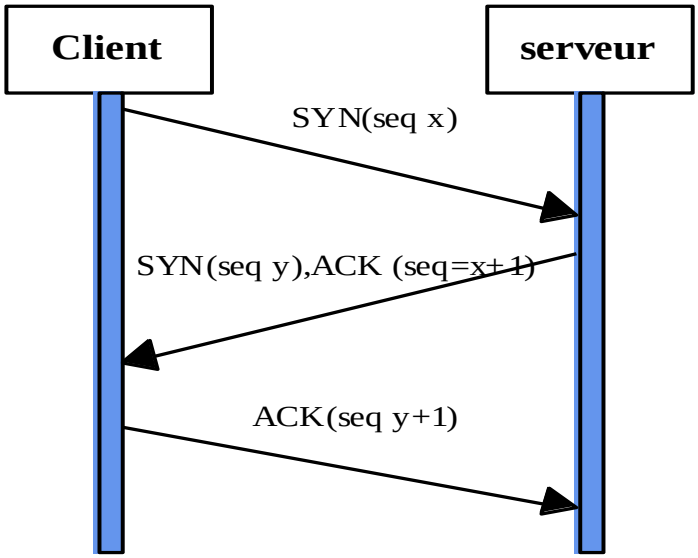


Figure 1.5: Établissement d'une connexion.

Ce processus de connexion permet entres autres d'initialiser et de synchroniser quelques valeurs de la connexion entre extrémités, telles que le numéro d'ordre (appelé aussi numéro de séquence) des segments :

- ✓ Le client envoie une séquence de synchronisation, avec le numéro de séquence. Le Flag "SYN" est positionné.
- ✓ Le serveur répond par une acceptation dans laquelle il renvoie :
  - un numéro d'acquittement égal au numéro de séquence qu'il a reçu+1.

• un numéro de séquence les flags SYN et ACK sont positionnés.

✓ Le client acquitte la réponse en envoyant :

•un numéro d'acquittement égal au numéro de séquence envoyé par le serveur +1.

•un numéro de séquence égal au numéro d'acquittement envoyé par le serveur [12].

Notons qu'il est possible pour deux systèmes d'établir une connexion entre eux simultanément.

4.3.2. Transfert de données

Le protocole TCP possède un système d'accusé de réception permettant au client et au serveur de s'assurer de la bonne réception mutuelle des données. Lors de l'émission d'un segment, le numéro de séquence lui est associé. A réception d'un segment de donnée, la machine réceptrice va retourner un segment de donnée dont le drapeau ACK est à 1 (afin de signaler qu'il s'agit d'un accusé de réception) accompagné d'un numéro d'accusé de réception égal au numéro d'ordre précédent [11].

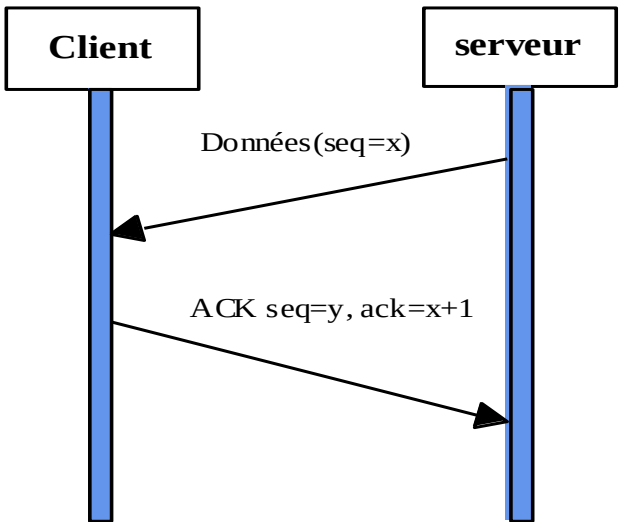


Figure 1.6: Transfert des données.

De plus, grâce à une minuterie déclenchée dès l'envoi d'un segment au niveau de la machine émettrice, le segment est réexpédié dès que le temps imparti est écoulé, car dans ce cas la machine émettrice considère que le segment est perdu.

Toutefois, si le segment n'est pas perdu et qu'il arrive tout de même à destination, la machine réceptrice saura grâce au numéro d'ordre qu'il s'agit d'un doublon de segment et ne conservera que le dernier segment arrivé à destination [11].

Pendant cette phase de transfert de données, certains mécanismes permettent d'assurer la robustesse et la fiabilité de TCP, nous citons particulièrement : [4]

- **Contrôle d'erreurs**

Pour éviter des erreurs éventuelles de transmissions, TCP déploie un mécanisme de contrôle d'erreurs. Le checksum que porte chaque segment TCP, calculé par l'émetteur avant transmission, est utilisé par le récepteur pour vérifier la validité des données transmises. Une fois que ces données vérifiées intactes, le destinataire doit acquitter les segments reçus en indiquant qu'il a reçu toutes les données du flux d'octets jusqu'à un certain numéro de séquence.

- **Acquittements ACK**

En effet, TCP utilise la technique de l'accusé de réception, lorsque un destinataire reçoit un paquet valide il retourne un accusé ACK qui confirme la bonne réception du paquet, ainsi l'émetteur sait si l'information qu'il voulait transmettre est bien parvenu à destination, ces ACK peuvent être immédiat ou cumulatif.

- **Contrôle de flux**

Pour éviter des engorgements du récepteur, l'émetteur adapte le nombre de paquets envoyés selon la taille du buffer de réception, cette information parvient à l'émetteur à travers les acquittements. Quand le récepteur retourne un ACK confirmant la bonne réception, cet ACK contient le nombre d'octets qu'il peut recevoir dans le prochain fragment TCP.

- **Temporisation et retransmission**

Afin d'atteindre une bonne fiabilité, une entité TCP envoie un paquet et attend son acquittement, si au bout d'un temps RTO (Retransmission Time Out) l'émetteur ne reçoit pas un accusée de réception, il décide de retransmettre le paquet, ce délai doit être supérieur au RTT (Round Trip Time) qui correspond au temps écoulé entre l'émission d'un paquet de données et la réception de son acquittement respectif.

- **Contrôle de congestion**

Le contrôle de congestion est une fonctionnalité majeure du protocole TCP. Il est utilisé pour atteindre de hautes performances et éviter l'effondrement du réseau. Le mécanisme de contrôle de congestion essaie de maintenir le taux des données entrant dans le réseau en dessous du taux qui cause une congestion.

La congestion est résolue à l'aide des algorithmes [1] : CWND (Congestion Window) et SSTHRESH (Slow Start Threshold).

• CWND: qui limite le nombre de bytes que l'émetteur des données peut avoir en suspend.

• SSTHRESH: permet de choisir le bon algorithme de congestion au bon moment .

4.3.3. Terminaison d'une connexion

La fermeture d'une connexion se fait en deux manières : normale ou brutale.

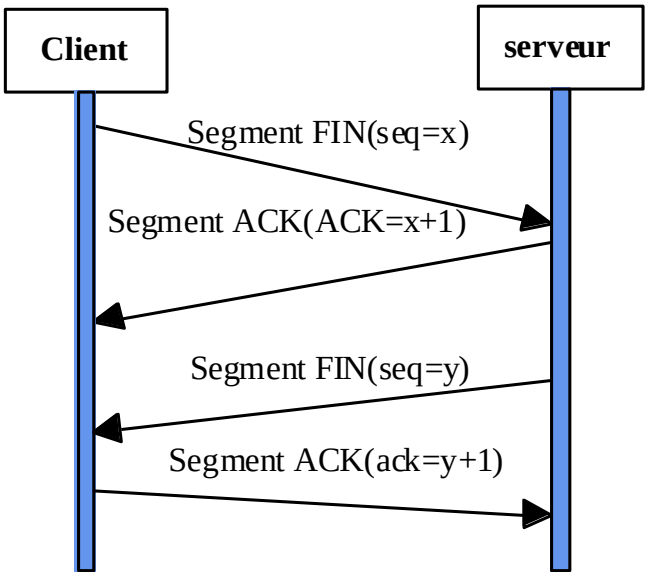


Figure 1.7: Libération de connexion.

Dans une terminaison normale de la connexion, le client peut demander à mettre fin à une connexion au même titre que le serveur, en suivant les étapes suivantes :

- ✓ Une des machines envoie un segment avec le drapeau FIN à sa valeur 1, et l'application se met en état d'attente de fin, c'est-à-dire qu'elle finit de recevoir le segment en cours et ignore les suivants.
- ✓ Après réception de ce segment, l'autre machine envoie un accusé (Ack) de réception avec le drapeau FIN à 1 et continue d'expédier les segments en cours. Suite à cela la machine informe l'application qu'un segment FIN a été reçu, puis envoie un segment FIN à l'autre machine, ce qui clôture la connexion [11].

Pour une terminaison brusque :

• Envoi de la primitive ABORT (flag RST=1)

• Toutes les transmissions ou réceptions sont interrompues et les tampons sont vidés [13].

4.4. Conclusion

TCP offre un service de transport fiable, en mode connecté qui garantit le respect de l'ordre des transmissions. Il gère les données en un flux d'octets (flot, ou stream) et autorise l'échange bidirectionnel simultané (Full duplex).

Notons que des améliorations sont ajoutées à TCP dont on cite une extension qui évite des retransmissions inutiles (si un paquet perdu d'un bloc), nommée acquittement sélectif (Selective ACK ou SACK) RFC 2018, la mise en œuvre de cette option est négociée lors de l'établissement de connexion et elle permet au destinataire de sélectionner et préciser le paquet à retransmettre [4].

5. Protocole SCTP (Stream Control Transmission Protocol)

5.1. Présentation

Pour répondre aux nouveaux besoins des transmissions de données sur Internet, que les protocoles TCP ou UDP ne peuvent satisfaire qu'en intégrant des extensions majeurs, l'IETF élabore le protocole SCTP .Une première version a été proposée en octobre 2000 par le groupe de travail IETF SIGRAN et a été spécifié dans RFC 2960[1]. SCTP est ensuite devenu un protocole d'utilisation générale du niveau transport, comme TCP et UDP.

SCTP implémente les mécanismes communs de la couche transport, il garde les principaux points forts de TCP tel que le contrôle de flux, la détection des erreurs et la retransmission, tandis que de nouvelles fonctionnalités ont été ajoutées, particulièrement le multi-homing, le multi-streaming.

Dans la terminologie SCTP, la connexion entre émetteur et destinataire est appelée association et les éléments qui communiquent entre eux sont appelés points terminaux (endpoint) [14].

5.2. Mécanismes spécifiques à SCTP

5.2.1. Multi-homing

La propriété essentielle du protocole de transport SCTP est le support des nœuds avec des connexions multiples, c'est-à-dire des nœuds qui peuvent être atteints via plusieurs

adresses IP. Le multi-homing spécifié dans le protocole SCTP sert uniquement pour la redondance, la répartition de charge ne fait pas partie de la spécification actuelle du protocole et reste toujours à l'étude [4].

Dans la terminologie IP, un nœud de communication ou une machine est appelé "multi-homed" si il peut être adressé par plusieurs adresses. "Multi-homed" signifie que la machine a été installée avec plusieurs cartes d'interface, chacune ayant une adresse IP différente [15].

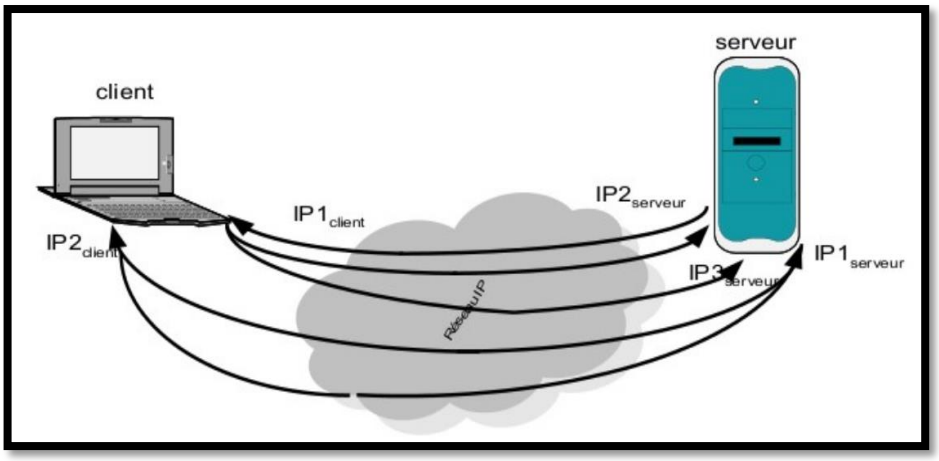


Figure 1.8: Exemple de nœuds IP multi-homed [16].

Via SCTP, l'application 1 du nœud A communique avec l'application 1 du nœud B.L'association établie sera décrite de la manière suivante :

Association = {[@IP1, @IP2 : Port 1] : [@IP : Port 1]} [17].

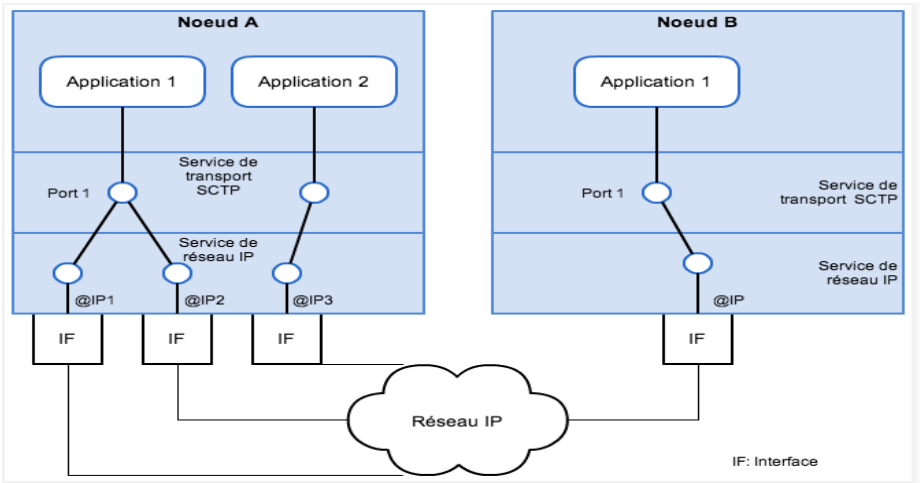


Figure 1.9: Association SCTP via mécanisme multi-homing [17].

5.2.2. Multi-streaming

Le multi-streaming est un nouvel apport du protocole SCTP. Il va permettre aux processus en communication de s'échanger des messages, au sein d'une même association, par l'intermédiaire de plusieurs flux (streams) de transmission.

C'est une caractéristique qui rend SCTP différent des protocoles de transport existants. Contrairement à TCP ou les données sont transmises en flot d'octets, dans SCTP les données sont envoyées en flot de messages. Mais de plus, SCTP permet d'avoir plusieurs flots de données dans une association.

Comme la gestion d'acquittement est basée sur des flots, la transmission de chaque flot est indépendante. En cas de blocage ou de perte d'un message d'un flot, les autres flots peuvent toujours être délivrés sans contraintes. Ainsi des problèmes de transmission sont résolus au niveau du flot considéré [18] comme le problème de *Head-of-Line Blocking* sur l'unique flux de TCP induit par la nécessité d'une livraison par ordre [19].

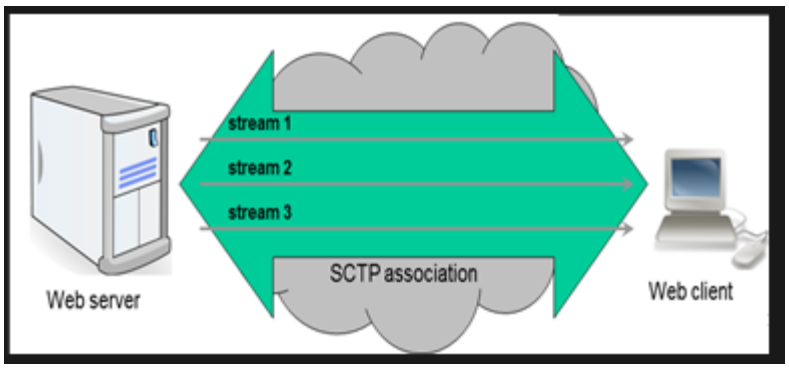


Figure 1.10: Multi-streaming [20].

Le nombre de streams en émission et en réception est négocié lors de la mise en place de l'association.

Parmi des applications qui peuvent bénéficier du multi-streaming, on peut mentionner des protocoles de signalisation dans des réseaux téléphoniques PSTN (Public Switched Telephone Network) ou des navigateurs web.

Dans le premier cas, ce qui est essentiel, c'est l'indépendance des flots dans une association et la possibilité de livraison de messages en dehors de séquence. Si un message est bloqué ou retardé, les autres peuvent être délivrés au destinataire.

Dans le deuxième cas, le navigateur peut télécharger une page entière dans une association. Tous les objets de la page (fichiers html, images, sons, styles, etc.) peuvent être téléchargés en même temps dans des flots indépendants. Cela permet d'économiser des ressources, du côté du client aussi bien que du côté du serveur [18].

En d'autres termes, le but initial de SCTP étant de transporter la signalisation, il trouve aujourd'hui son intérêt dans le transport de VoIP (la Voix sur les réseaux IP)[4]. C'est pourquoi, vu que les contraintes en termes de temps de livraison de ce type de données sont plus strictes que celles liées aux données classiques, SCTP gère des temporisateurs plus courts que ceux de TCP.

5.2.3. Acquittement sélectif

Pour la transmission des données, SCTP assure un acquittement sélectif qui correspond à acquitter la réception de tous les chunks reçus en séquence sans erreurs en indiquant la dernière valeur des paquets de données correctement reçues ou les plages de messages de données reçues séparément (hors séquence).

L'acquittement sélectif permet également une indication des séquences de paquets dupliqués. L'exploitation des informations d'acquittement sélectif reçus, permet à l'émetteur SCTP de retransmettre les messages qui n'ont pas été reçus, identifiés par leur numéro de séquence. Cette retransmission s'effectue après l'expiration du temporisateur ou la réception de 4 paquets d'acquittement indiquant la perte du même message de données [16].

5.3. Format de l'unité de protocole

La structure d'une unité de protocole SCTP ou de message SCTP est modulaire: elle est composée d'un entête commun et d'un ensemble de modules appelés chunks [4].

Un chunk est un ensemble de bits de tailles variables qui sont divisés en deux classes : chunk de contrôle et chunk de données. Un chunk de contrôle est par définition un chunk qui contient des informations propres au contrôle et au maintien de l'association SCTP. Les chunks de données sont utilisés pour transporter des messages utilisateurs à travers l'association [21].

Chaque chunk est lui-même formé d'un entête suivi de la charge utile (des données utilisateur ou bien des données de contrôle).Plusieurs chunks peuvent être multiplexés dans un seul paquet SCTP (Figure 1.11)

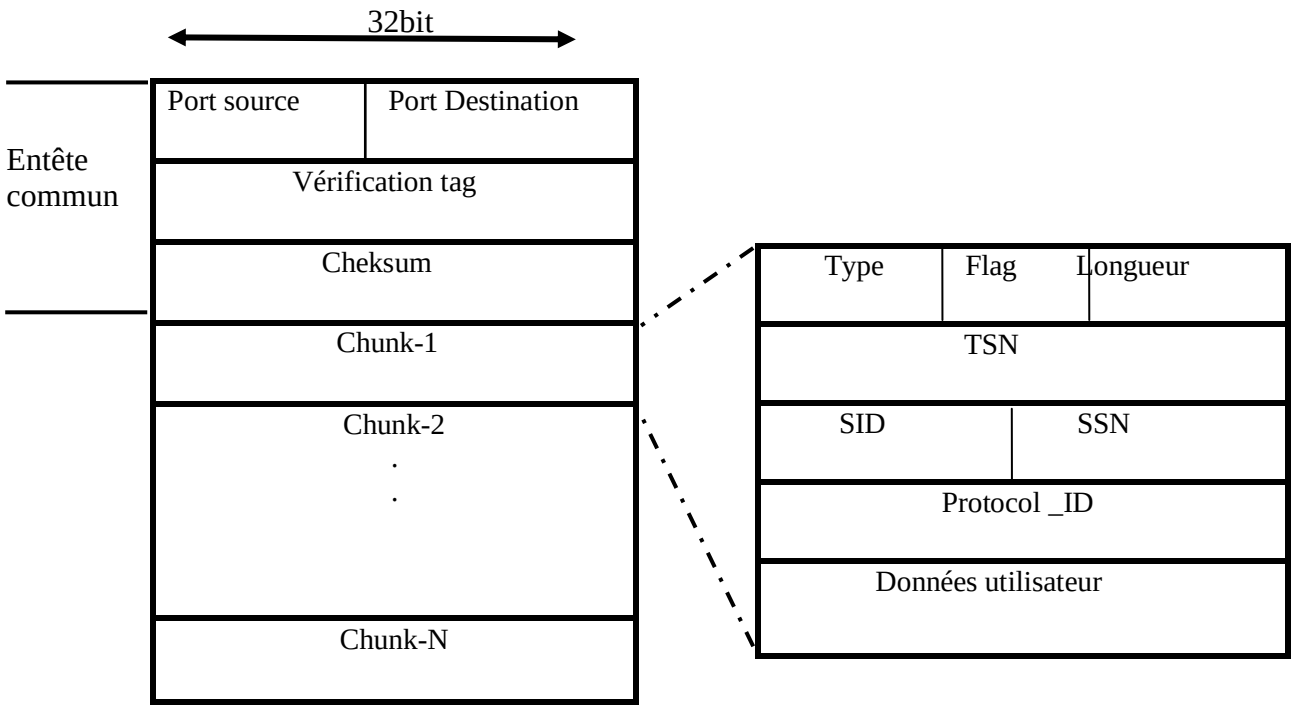


Figure 1.11: Format d'un paquet SCTP.

✓ **En-tête commun**

Chaque message SCTP contient un en-tête commun qui a 4 champs différents. L'en-tête comporte 12 octets [21].

- **Port Source et Destination :** les mêmes que dans TCP.
- **Tag de vérification :** une information spécifique attaché à un message d'une association et qui fournit une clé au récepteur afin de vérifier qu'un message SCTP appartient à l'association courante et pas d'un message d'une ancienne association.
- **Checksum** (32 bit contre 16 bits dans TCP) **:** ce champ est utilisé pour corriger les erreurs éventuelles qui peuvent subir sur un message SCTP lors de son trajet [4].

✓ **Chunks**

Le nombre N de chunks est déterminé par la taille du MTU (Maximum Transmission Unit), c'est la taille maximale d'un message SCTP qui ne subit à aucune fragmentation.

- **Type de chunk** sur 8 bits**:** permet de déterminer le type de la charge utile du chunk (données ou informations de contrôle...) [16].Les principaux chunks définis dans SCTP sont récapitulé dans la table suivante :

ID valeur	Chunk type	Description
0	DATA	Les données d'utilisateur.
1	INIT	Initiation d'une association.
2	INIT ACK	Acquittement de la portion INIT.
3	SACK	Acquittement de la réception de données.
4	HEARTBEAT	Vérifier l'accessibilité via des chemins alternatifs.
5	HEARTBEAT-ACK	Acquittement de la portion HEARTBEAT.
6	ABORT	Fermeture immédiate d'une association.
7	SHUTDOWN	Début de la fermeture normale d'une association.
8	SHUTDOWN ACK	Acquittement de la portion SHUTDOWN.
9	ERROR	Erreur d'opération
10	COOKIE ECHO	Porte le COOKIE pendant l'initialisation d'une association.
11	COOKIE ACK	Acquittement de la portion COOKIE ECHO.

Tableau 1.1: Listes des types des chunks [4].

- **Chunk Flag** sur 8 bits**:** ce champ est utilisé différemment selon le type de chunk. Il apporte des indicateurs concernant le type de livraison, soit en séquence ou que le séquencement de données non requis [16].

– U, vient de "(U)nordered" : si cette valeur égale 1, alors ceci est un chunk de données non ordonné. Il n'y a pas de numéro de séquence de Stream pour cette donnée.

– B, vient de "(B)eginning fragment" : indique le premier fragment d'un message utilisateur.

– E, vient de "(E)nding fragment" : indique le dernier fragment d'un message utilisateur[21].

- **ChunkLength** sur 16 bits: la taille du chunk est variable et dépend de la taille du champ des données utilisateur.

- **TSN** (Transmit Sequence Number) sur 32 bits : numéro de séquence des fragments. Il permet de reconstruire un message fragmenté [16].

- **SID** (Stream IDentifier) sur 16 bits**:** identifie les flux des chunks de données Ceci n'est utilisé que lorsque SCTP transporte plusieurs flux dans une association (multi-streaming) [21].

- **SSN** (Stream Sequence Number) sur 16 bits **:** il s'agit du numéro de séquence du chunk dans le stream. Le SSN permet le ré-ordonnancement des données par le récepteur sur chaque flot.

- **Protocol_ID** (protocole IDentifier) sur 32 bits **:** sert à indiquer le type d'information contenue dans les chunks de données. C'est un champ non utilisée par SCTP mais par la couche application [16].

5.4. Étape d'association SCTP

Le mode de connexion implémenté dans la spécification du SCTP, est similaire au TCP, avant tout échange des données, les deux entités doivent établir une association entre eux, cette association SCTP passe par les états : établissement d'association, échange des messages et fermeture d'une association.

5.4.1. Établissement d'une association

Un des points faibles de TCP est la possibilité d'effectuer une attaque de DOS (Denial of Service) de type SYN-flood. Il s'agit d'une attaque où un attaquant commence plusieurs connexions avec de fausses adresses source. Comme ces adresses sont fausses, voire n'existent pas, le serveur TCP ne peut pas répondre ni établir la connexion complète. Dans cette situation, le serveur consomme des ressources (mémoire) pour maintenir les connexions semi-ouvertes. Cela provoque un épuisement de la mémoire et finalement un blocage du serveur.

Pour se protéger contre ce type d'attaques, le protocole SCTP prévoit un mécanisme d'initialisation sécurisée basé sur un COOKIE. C'est une valeur calculée par le serveur pour vérifier le client qui a demandé l'initialisation d'une association [18].

Le schéma d'établissement d'une association en quatre temps entre deux hôtes SCTP est présenté dans la figure 1.12.

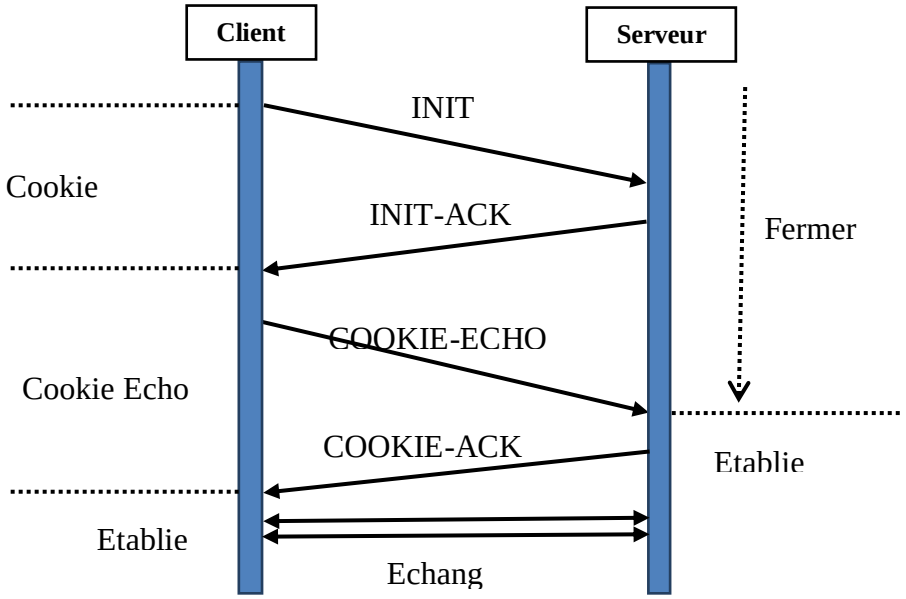


Figure 1.12: Etablissement d'association SCTP.

INIT

L'association étant à l'état **CLOSED** du côté du serveur qui attend un message du client avec un chunk INIT, ce chunk signale la requête qu'un client demande la création d'une association. Le client envoie un chunk INIT au serveur et déclenche un temporisateur lui permettant de réémettre le chunk en l'absence de réponse. Ensuite le client se met en attente du message de réponse contenant le cookie [16].

Le message INIT contient obligatoirement les informations suivantes [18]:

- Initiate Tag : la valeur qui sera utilisée par le client comme étiquette de vérification (Verification Tag) dans tous les messages émis pendant l'association.
- A_RWND (Advertised Receiver Window Credit):le nombre d'octets que le client peut recevoir du serveur.
- Number of Outbound Streams : le nombre de flots sortants proposés par le client pour cette association.
- Number of Inbound Streams : le nombre de flots entrants proposés par le client pour cette association.
- Initial TSN : la valeur initiale du TSN pour le client. La valeur du Initiate Tag peut être utilisée comme le TSN initial.

Dans le message INIT le client peut aussi mettre optionnellement comme informations :

- Cookie Préservative : la durée de validité des COOKIES. Par défaut un COOKIE est valable pendant 60 secondes.

Après avoir émis le *chunk* INIT l'association, du côté client, aura un état **COOKIEWAIT.**

INIT-ACK

Un serveur qui reçoit un message INIT se base sur les données reçues et calcule un COOKIE. Ce COOKIE est important pour le serveur, car c'est lui qui doit le reconnaitre dans l'étape suivante. Le client ne traite pas de COOKIE, mais le renvoie simplement au serveur. L'association garde l'état **CLOSED** du côté serveur [16].

Dans le message INIT-ACK le serveur met aussi obligatoirement les informations suivantes[18] :

- Initiate Tag: la valeur qui sera utilisée par le serveur comme étiquette de vérification (Verification Tag) dans tous les messages émis pendant l'association.
- A_RWND: le nombre d'octets que le client peut recevoir du serveur.
- Number of Outbound Streams : le nombre de flots sortants proposés par le serveur pour cette association.
- Number of Inbound Streams : le nombre de flots entrants proposés par le serveur pour cette association.
- Initial TSN : la valeur initiale du TSN pour le serveur. La valeur du Initiate Tag peut être utilisée comme le TSN initial.

Le serveur peut ajouter des paramètres optionnels comme :

- Error : si le serveur n'a pas reconnu un des paramètres dans le message INIT, il envoie un message d'erreur.

COOKIE-ECHO

Après la réception de INIT-ACK du serveur, le client renvoie au serveur le COOKIE trouvé dans le message INIT-ACK, quitte l'état COOKIE-WAIT et bascule à l'état de COOKIE-ECHOED.

Après la réception du message COOKIE-ECHO, le serveur calcule encore une fois le COOKIE de la même façon qu'après la réception du message INIT. Si le COOKIE calculé est identique au COOKIE reçu dans le message COOKIE-ECHO, le serveur peut être sûr que l'adresse du client est réelle et qu'il peut établir une association avec ce client.

Dans le cas contraire, le serveur supprime le message COOKIE-ECHO et termine l'initialisation sans établir une association [18].

COOKIE-ACK

Dès que le serveur reçoit un message COOKIE-ECHO, il signale au client que l'association est établie par l'envoie d'un chunk COOKIE-ACK. L'association du côté serveur, passe à l'état **ESTABLISHED**.

Le client à la réception d'un chunk COOKIE-ACK, conclut l'ouverture de l'association SCTP. C'est à partir de cet instant que le client commence à émettre des paquets à destination du serveur. L'association du côté client passe donc à l'état **ESTABLISHED** [22].

5.4.2. Transfert des paquet SCTP

Le protocole SCTP permet, grâce à son mécanisme de multi-streaming, l'envoie de données à partir de plusieurs flots (streams) dans un même paquet grâce à une technique d'identification de stream pour chaque chunk [22].

Les données d'une application peuvent être découpées sur plusieurs flux, SCTP pouvant chacun remettre les messages à la couche supérieure de manières différentes, alors que TCP devrait établir plusieurs sessions en parallèle pour gérer ce type de transmission [21].

Lors du transfert, si un message utilisateur est de taille supérieure au PMTU (Path Maximum Transmission Unit) d'une association, il sera fragmenté en plusieurs chunks. Les chunks formant un même message ont des TSNs successifs, et sont identifiés par le même SSN. L'identification de l'ordre des différentes parties d'un message utilisateur est assurée au moyen des flags B/E (premier chunk, dernier chunk) [22].

SCTP est donc un protocole qui assure optionnellement un ordre total au sein d'un même flux et n'offrant aucune garantie sur l'ordre entre les flux. Ce qui lui permet de délivrer les données d'un flux même si des pertes ou des hors-séquences sont détectées sur un autre flux [21].

SCTP est un protocole qui assure la fiabilité des transmission via un chunk de contrôle : le SACK chunk. Ce dernier est utilisé par le récepteur SCTP qui l'envoie en retour afin d'acquitter les DATA chunks reçus. Le SACK chunk permet à l'émetteur d'identifier les chunks qui ont été perdus, les chunks qui ont été reçus dupliqués, ainsi que toutes les informations nécessaires à d'éventuelles retransmissions ou à la gestion de streams.

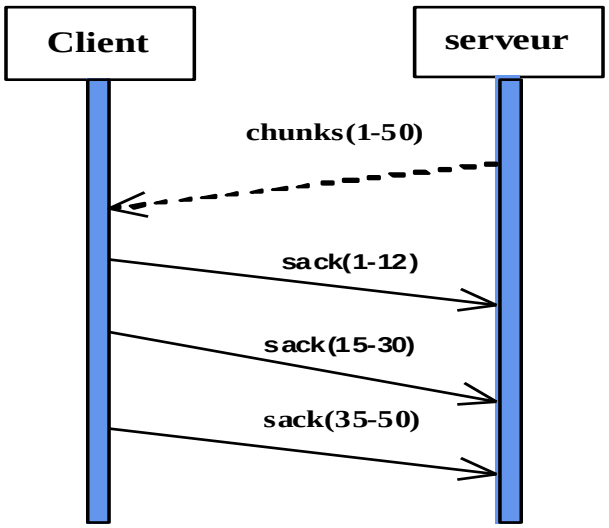


Figure 1.13: Transfert fiable de donnée.

Pendant cette phase de transfert, certains mécanismes permettent d'assurer la robustesse et la fiabilité de SCTP, dont on cite[4] :

- **Contrôle de flux**

Le champ A_RWND qui se trouve dans le SACK chunk, indique à l'émetteur combien de bytes le récepteur est prêt à recevoir. Utilisé avec les champs Cumulative TSN acknowledgment et Gap Ack Blocks (qui se trouve sur le chunk sack), l'émetteur peut avoir une vue précise de la taille du buffer de réception.

- **Contrôle de congestion**

Dans le réseau, la congestion peut se trouver à deux endroits : au récepteur (taille des buffers insuffisante) ou dans le réseau (bande passante de la liaison atteinte).Dans le premier cas, le problème est résolu avec le champ A_RWND, qui se trouve dans les chunks de type INIT, INIT-ACK et SACK.

Le deuxième cas, plus complexe à gérer, est résolu à l'aide de plusieurs algorithmes : CWND, SSTHRESH (définit précédemment dans TCP).

5.4.3. Fermeture d'une association

La connexion est finalement fermée soit par un bloc SHUTDOWN (*graceful shutdown*) de manière élégante soit par un bloc ABORT (*abortive shutdown*) avec lequel, la connexion est supposée être terminée de façon forcée ou suite à un évènement inattendu [14].

- Avec le *graceful shutdown*, SCTP permet une fermeture en trois étapes**[**18]. comme le montre la figure 1.14

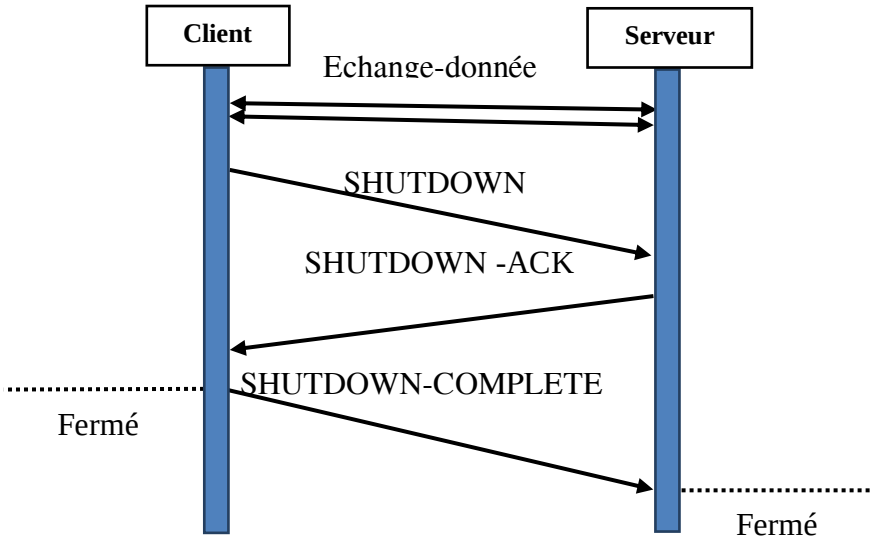


Figure 1.14: Fermeture normale d'une association SCTP.

SHUTDOWN

Le client envoie un message SHUTDOWN au serveur quand il n'a plus de données à envoyer et quand toutes les données envoyées ont été déjà acquittées par le serveur. Après ce message le client n'envoie plus des données au serveur.

SHUTDOWN-ACK

En réponse d'un message SHUTDOWN, le serveur envoie un message SHUTDOWN ACK. Le serveur peut envoyer ce message quand il a terminé d'envoyer toutes ses données au client. Après ce message, le serveur n'envoie plus de données au client.

SHUTDOWN-COMPLETE

Le client termine la procédure de fermeture en envoyant un message SHUTDOWNCOMPLETE. Cette méthode de fermeture d'une association permet d'éviter des associations semi-ouvertes.

6. Comparaison entre SCTP et TCP

Les principales différences entre SCTP et TCP sont présentées sur le tableau 1.2 ci-dessous:

Caractéristique/Services	**SCTP**	**TCP**
Orienté connexion	Oui	Oui
Full duplex	Oui	Oui
ACK sélectif	Oui	Optionnel
Transfert de données fiable	Oui	Oui
Multiplexage de message	Oui	Oui
Reconstitution de messages d'origine	Oui	Oui
Découvert MTU	Oui	Oui
Contrôle de congestion	Oui	Oui
Protection contre les SYN attaques	Oui	Non
Permet des connexions demi-fermées	Non	Oui
Support du multi-homing	Oui	Non
Support du multi-streaming	Oui	Non
Livraison ordonnée sur un stream	Optionnel	Oui
Livraison ordonnée entre streams	Indépendants	/
Sécurité	Oui	Non

Tableau 1.2: Synthèse comparative entre SCTP et TCP [3].

7. Conclusion

Dans ce chapitre nous avons présenté les deux modèles de référence OSI et TCP/IP, leurs différentes couches et leur fonctionnement. Ensuite, nous avons présenté en détail la couche de transport via les protocoles SCTP et TCP et identifié leurs mécanismes, en particulier le multi-streaming.

Chapitre 2

Outils de simulation

1. Introduction

Dans les réseaux informatique, il est très couteux de valider et vérifier un protocole ou un certain algorithme spécifique, c'est pour cela que les simulateurs de réseaux sont utilisés.

Les simulateurs du réseau offrent beaucoup d'économie, de temps et d'argent pour l'accomplissement des taches de simulation et sont également utilisés pour que les concepteurs des réseaux puissent tester les nouveaux protocoles ou modifier les protocoles déjà existants d'une manière contrôlée et productrice [24].

La mise en œuvre d'une simulation nécessite une étape qui décrit la topologie du réseau et le comportement prévu de ses composants, une seconde étape de simulation proprement dite qui permet enfin la récolte et l'interprétation des résultats.

Le but de ce chapitre est de présenter les caractéristiques de quelques simulateurs, à titre comparatif.

2. Simulation des réseaux informatiques

Simuler c'est reproduire le fonctionnement d'un système réel à partir de la modélisation des éléments qui le constitue et de leur fonctionnement. L'observation réelle de ces systèmes est généralement complexe, à risques ou couteuse. Il s'agit d'une approche permettant de représenter le fonctionnement d'un système réel constitué de plusieurs entités, de modéliser les différentes interactions entre elles, et enfin d'évaluer le comportement global du système et son évolution dans le temps afin de prévoir son comportement dans le monde réel ou d'en définir les meilleurs paramètres de fonctionnement [25].

Toutefois, il est nécessaire de bien identifier les caractéristiques du système afin de le représenter, le plus finement possible, par des modèles abstraits.

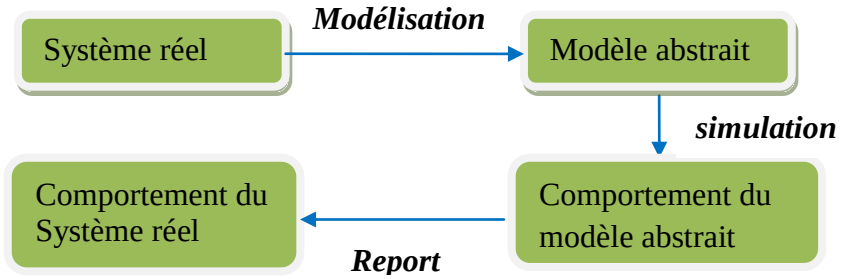


Figure 2.1: Modélisation et simulation d'un système réel.

Si la représentation du système réel par des modèles abstraits et leur simulation sont suffisamment réaliste et précise, il est alors possible de reporter les paramètres relatifs aux résultats obtenus suite à leur simulation sur le système réel [25]. Le cycle correspondant aux étapes de modélisation, simulation et report des résultats est illustré sur la figure 2.1.

Les simulateurs réseaux sont utilisés, entres autres, par les chercheurs universitaires, pour concevoir, simuler, vérifier et analyser les performances des protocoles de différents réseaux. Ils peuvent également être utilisés pour évaluer l'effet des différents paramètres des protocoles étudié [26].

En général, un simulateur de réseau est composé d'un large éventail de technologies et de protocoles réseaux et aide les utilisateurs à construire des réseaux complexes à partir de blocs de construction de base comme des grappes de nœuds et de liens. Avec leur aide, nous pouvons concevoir différentes topologies de réseau en utilisant différents types de nœuds tels que les nœuds terminaux, concentrateurs, ponts de réseau, routeurs, et des unités mobiles[26].

3. Divers simulateurs de réseaux informatiques

Il existe plusieurs simulateurs de réseaux qui diffèrent principalement par la simplicité d'utilisation, le principe de fonctionnement et le volume et précision des éléments et résultats de simulation. Nous présentons dans ce qui suit quelques un, tel que : OPNET, Packet Tracer, NS2 et NS3et leurs principales caractéristiques.

3.1.Simulateur OPNET

OPNET (Optimum Network Performance) est un outil de simulation de réseaux très puissant et très complet. Basé sur une interface graphique intuitive, son utilisation et sa prise en main est relativement aisée [27]. Il présente une très bonne interface graphique relativement complète avec une librairie suffisamment fournie pour une très large gamme d'utilisation et d'application. Evidemment, l'éditeur graphique de ce simulateur ou GUI (Graphic User Interface) nous permet, entre autre, de construire différentes topologies et architectures de réseaux pour différentes applications et avec différents protocoles [28].

Le simulateur OPNET dispose de trois niveaux de domaine hiérarchique, respectivement de niveau plus haut au niveau plus bas : network domain, node domain et process domain[29].Il se compose d'interface utilisateur de haut niveau, qui est construite à partir de blocs de code source C et C++ [30].

Il est l'un des meilleurs logiciel de simulation de réseaux présent sur le marché, le seul problème d'OPNET c'est qu'il est payant mais ce problème est résolu avec la version académique (la version 9.1) [31].

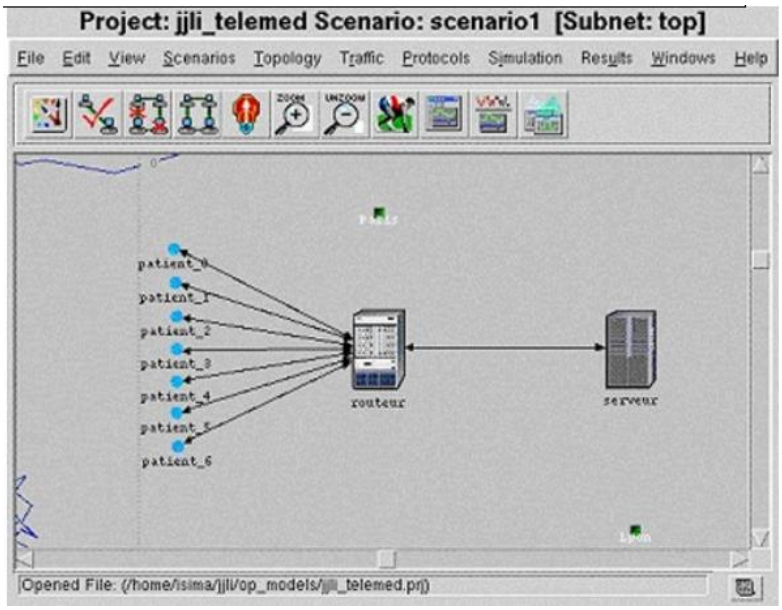


Figure 2.2 : Interface conviviale du simulateur OPNET [27].

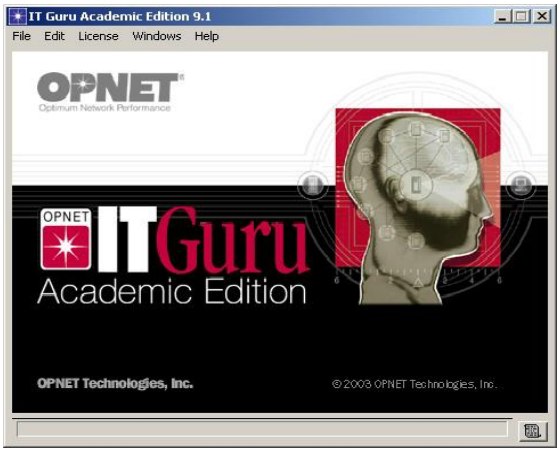


Figure 2.3 : Version académique [28] .

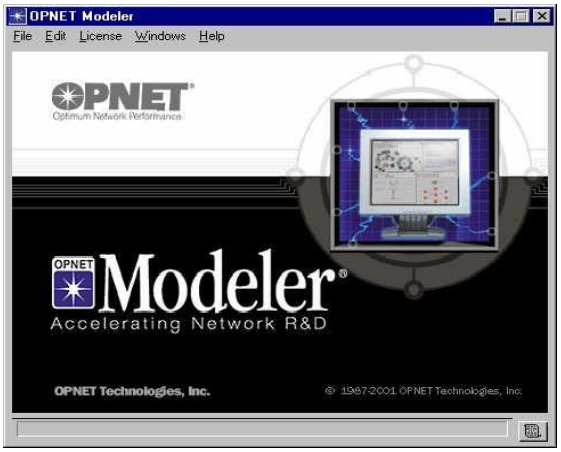


Figure 2.4 : Version commerciale[28].

3.2. Simulateur Packet Tracer

C'est un simulateur de routeur Cisco didactique, généralement utilisé dans la formation et l'éducation, mais parfois même dans la recherche pour les simulations de réseaux informatiques. L'outil est créé par Cisco Systems et prévoit la distribution gratuite aux professeurs, aux étudiants et anciens étudiants qui sont ou ont participé à la Cisco Networking Academy. Le but de Packet Tracer est d'offrir aux étudiants et aux enseignants un outil pour apprendre les principes de mise en réseau ainsi que de développer des compétences technologiques Cisco spécifiques [27].

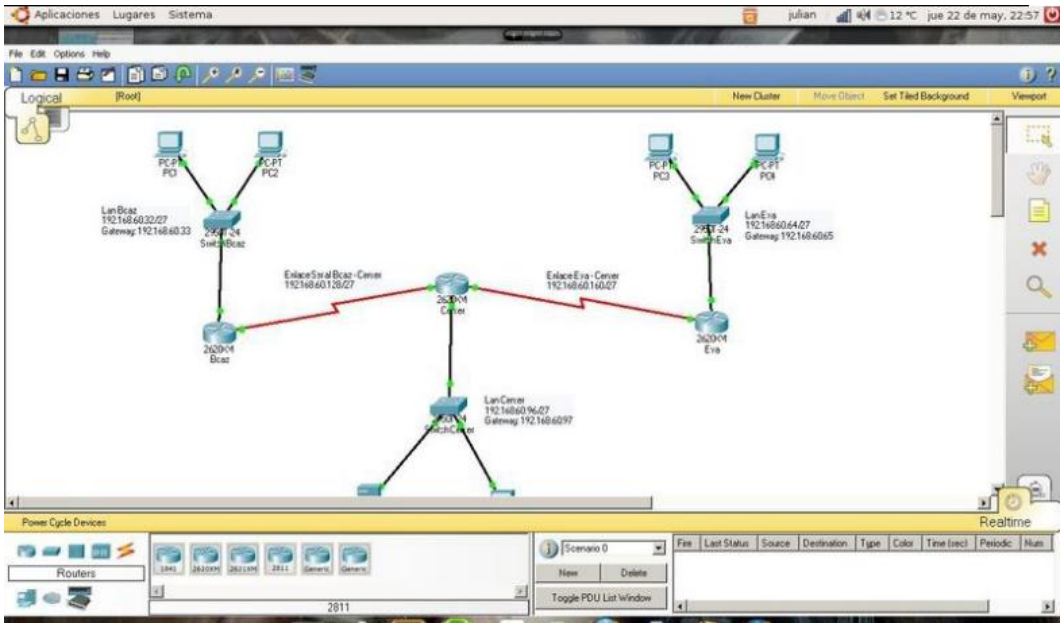


Figure 2.5 : Interface du simulateur didactique packet Tracer.**[27]**

3.3. Simulateur NS2

Network Simulator 2 est un logiciel de simulation de réseaux informatique développé en 1989, lors d'un projet de la DARPA (Defense Advanced Research Projects Agency) [32]. Il est essentiellement élaboré avec les idées de la conception par objets, de la réutilisation du code et de modularité.

NS2 est totalement Open Source et gratuit [32] disponible sur le site de référence, http://www.isi.edu/nsnam/ns.

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 (File Transfer Protocol) [33].

Ce simulateur 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 (Network AniMator) [34].

NS2 est exécutable tant sous Unix que sous Windows. Il combine le langage de script OTcl (Object Tools Command Langage) avec C++ dans lequel la plupart des protocoles réseaux ont été implémentés.

Bien qu'il implémente un noyau puissant en C++ et qui permet d'effectuer des calculs très précis et fiables, quelques lacunes de NS2 font obstacles à son déploiement étendu. Il s'agit principalement de sa mauvaise gestion et utilisation des traces ou des contraintes liées à son interface (textuelle via script Tcl (Tools command langage), utilisation de multiples interfaces sur un nœud,..)[27].

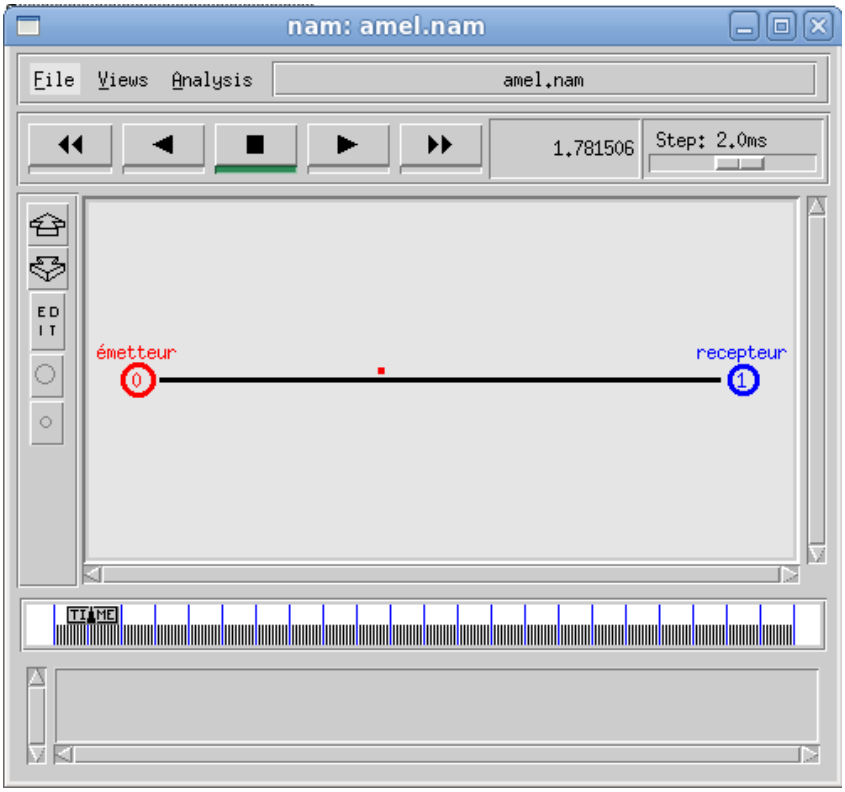


Figure 2.6: Interface du logiciel NAM pour la visualisation des résultats NS2**.**

3.4. Simulateur NS3

Network Simulator 3 est vise à remplacer son prédécesseur NS2, écrit en C++ et OTcl, pour tenter de remédier à ses limites.

Cependant, contrairement à NS2, il peut être utilisé sur les plateformes Linux/Unix, OS X (Mac) et Windows. Les développeurs de NS3 ont décidé suite à l'expérience tirée de NS2 d'associer les progrès des langages de programmation et du génie logiciel au développement de l'architecture de NS3. C'est pourquoi, l'idée de la rétrocompatibilité avec NS2 a été abandonné dès le départ. Cela libère NS3 de contraintes héritées de NS2 et permet la construction d'un simulateur qui est bien conçu depuis le début [26].

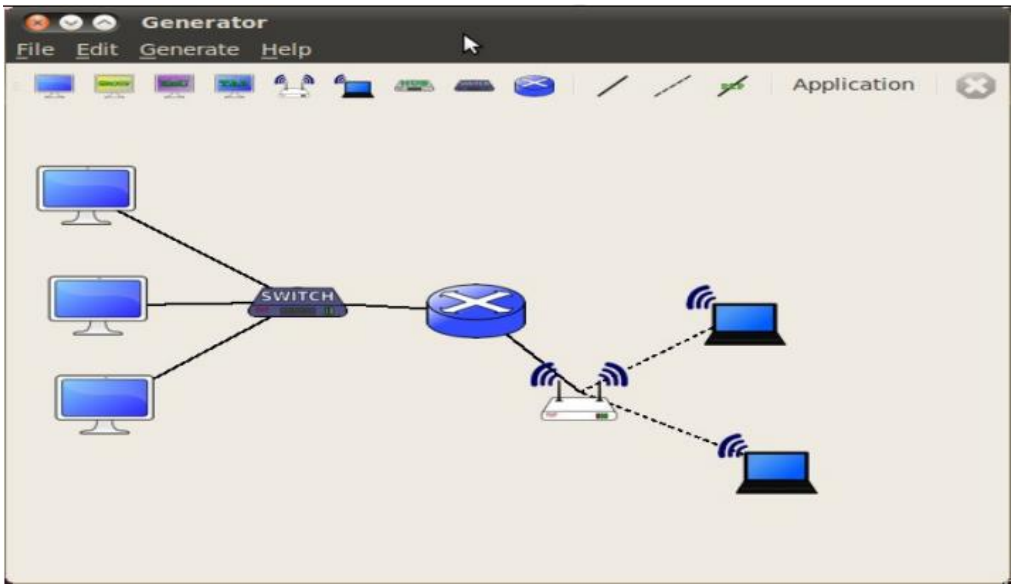


Figure 2.7 : Interface de NS3 pour générer les éléments de simulation [27].

4. Apports de NS3 comparé à NS2

Dans ce qui suit, nous nous intéressons aux deux simulateurs NS2 et NS3 qui représentent les outils utilisés concrètement dans notre travail. Mais avant de s'approfondir sur leurs caractéristiques, nous comparons les deux outils, notamment:

NS2: caractérisé par :

- Logiciel multicouches de simulation.
- Interface de programmation en Tcl et noyau écrit en C++.
- Développement orienté objet.
- Documentation ambigüe.
- Adapté aux petits réseaux.
- Analyse des résultats :

  NS2: stockage d'une trace de la simulation.

  NAM: visualisation de la simulation.

  Gnuplot, Xgraph, Tracegraph, …: Interprétation de la simulation.

NS3: caractérisé par :

- Peut être utilisé sur les plateformes Linux/Unix, OS X(Mac), et Windows (via Cygwin ou une machine virtuelle).
- Deux langages de programmation: C + +, Python.
- Contrairement à NS2, tout est écrit en C++ sous NS3.
- Beaucoup plus rapide en termes d'exécution (tout est préalablement compilé).
- Le code est bien documenté
- NS3 plus performant que NS2 en terme de gestion de mémoire.
- Visualisation NS3: ns3-viz, pyviz, nam,…[35].

5. Présentation détaillé de NS3

5.1.Terminologie

Il est important de bien comprendre le sens des termes employés au sein du simulateur, ainsi que les abstractions qui ont été faites. NS3 utilise des termes largement employés dans le domaine des réseaux, mais qui peuvent avoir une signification particulière au sein du simulateur. Voici les principaux : [26]

- **Une application**

Représente un code exécuté par un utilisateur. Ce code peut être nécessaire au déroulement d'une simulation. L'échange de paquets durant une simulation nécessite par exemple la description d'une Application au sein des nœuds participants. Par exemple, pour réaliser une application en mode client/serveur, il faut créer Udp Echo Client Application d'un côté et Udp Echo Server Application de l'autre. NS3 ne fait pas de distinction entre les "applications système" (souvent exécutées par le noyau) et les applications des utilisateurs (exécutées dans le user-space). Les applications peuvent ensuite être attachées à un Node.

- **Un canal de communication**

Représente le lien qui relie des nœuds, ou plus exactement les NetDevices installés dans les nœuds. Des spécialisations de cette classe sont définies, comme par exemple Csma Channel pour modéliser un lien Ethernet utilisant CSMA (Carrier Sense Multiple Access), ou encore WifiChannel pour modéliser un lien WiFi (Wireless Fidelity).

➢ **Une interface de communication**

Appelée NetDevice, qui modélise à la fois l'équipement (carte réseau ou interface réseau) et le pilote dont un ordinateur a besoin pour pouvoir communiquer avec d'autres. Des spécialisations de NetDevice sont fournies, comme par exemple CsmaNetDevice qui simule une carte réseau Ethernet et peut être reliée à un CsmaChannel, ou encore WifiNetDevice pour un lien à un canal de type WifiChannel.
Pour établir une connexion entre deux nœuds, il faut alors :

- ✓ Équiper chaque nœud de NetDevice.
- ✓ Configurer la couche protocolaire sur chaque nœud (élément non représenté qui se situe entre les applications et les NetDevice).
- ✓ Dans le cas d'une couche protocolaire TCP/IP, configurer les adresses MAC (Media Access Control) et IP sur chaque NetDevice.
- ✓ Créer le canal de communication correspondant.
- ✓ Connecter chaque NetDevice au canal de communication.

➢ **Topology Helpers**

S'il s'agit de connecter un grand nombre de nœuds les uns avec les autres, ce processus peut être très lourd. NS3 fournit des Topology Helpers pour faciliter ce genre de tâches. Les classes représentant ces objets sont dans le dossier NS 3.21/build/ns3. On y trouve par exemple :

- **unCsmaHelper** pour instancier des CsmaNetDevice sur un nœud, instancier un CsmaChannel et les connecter.

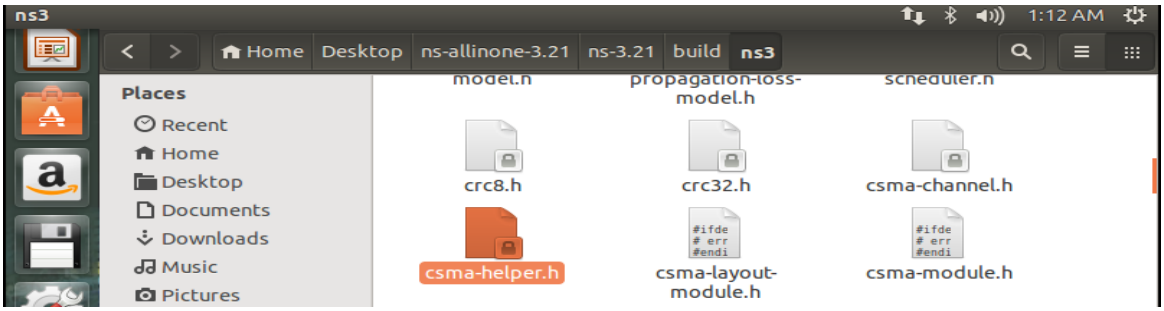


Figure 2.8: CsmaHelper sous NS3.21.

- **un WifiHelper**, qui a la même fonction que le CsmaHelper, mais pour une interface et un canal qui utilisent le WiFi.
- **un Internet StackHelper** pour instancier au sein d'un nœud la couche protocolaire IP/TCP/UDP et des fonctions de routage IPv4/IPv6.

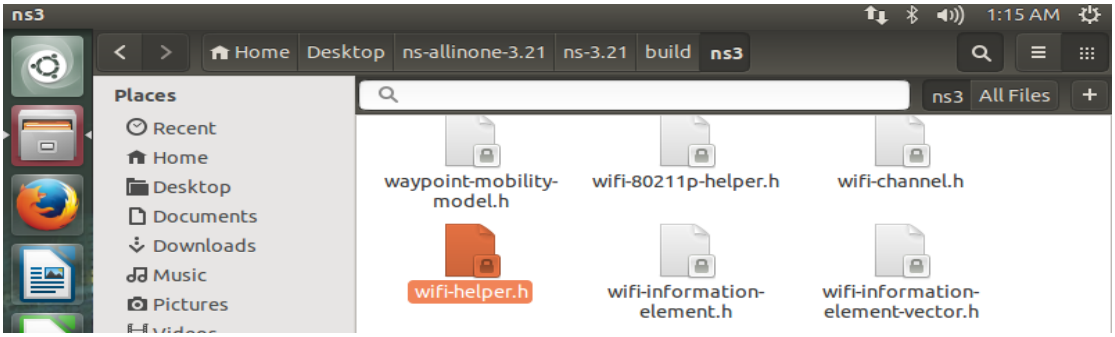


Figure 2.9: WifiHelper sous NS3.21.

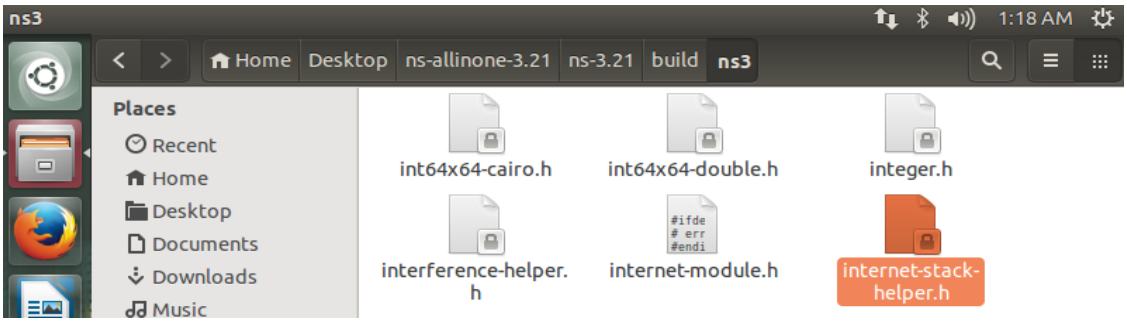


Figure 2.10: Internet StackHelper sous NS 3.21.

5.2. Modules du simulateur NS3

NS3 fournit différents modules qui peuvent être modifiés et effectivement utilisés. L'organisation du simulateur NS3 est illustrée dans la figure2.11. Les modules de base de NS3 sont :

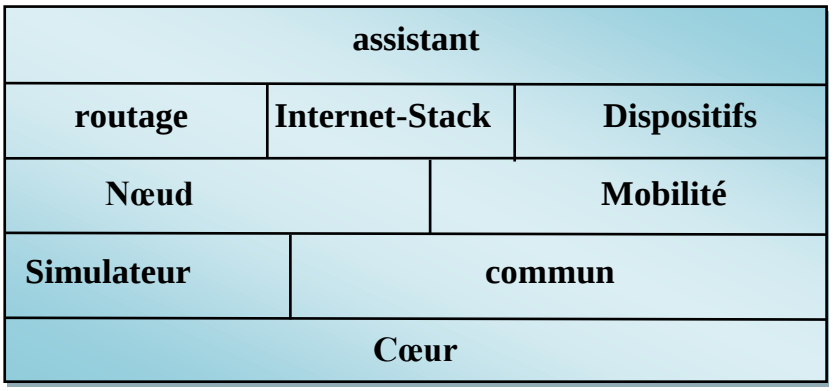


Figure 2.11 : Principaux modules de NS3.

- **Module cœur**

Permet la création d'une structure de classes hiérarchiques dérivant de la classe Object. Cette hiérarchie possède plusieurs fonctionnalités conçus spécialement pour répondre aux besoins de la simulation. Parmi lesquels : l'agrégation d'objets, l'enregistrement actif(TypeId) avec les attributs publiques… etc. Ce module de base est indépendant [37].

- **Module commun**

Ce module permet la génération des actions liées à l'émission et la réception des paquets [37].

- **Module Simulateur**

Gérer Les simulations des événements. Il prévoit explicitement la possibilité de planifier des événements à des moments différents et ensuite d'exécuter ces événements. La classe Time représente le temps simulé à haute résolution en utilisant un entier de 128 bits. Cette classe est la classe la plus importante du simulateur [26].

- **Module Mobilité**

Ce module permet de définir la position des nœuds et associe un modèle de mouvement aux agents de la simulation [26].

- **Module Nœud**

Le module Node simule un nœud réseau contenant des piles de protocoles. il peut contenir plusieurs NetDevices. Chaque NetDevice est attaché à un channel par lequel il envoie et reçoit des paquets. La figure12 illustre cette architecture pour deux nœuds Un nœud peut contenir plusieurs gestionnaires de protocole, qui acceptent les paquets reçus par le NetDevice. Pour lancer la transmission des paquets, chaque nœud peut également contenir une liste d'applications [36].

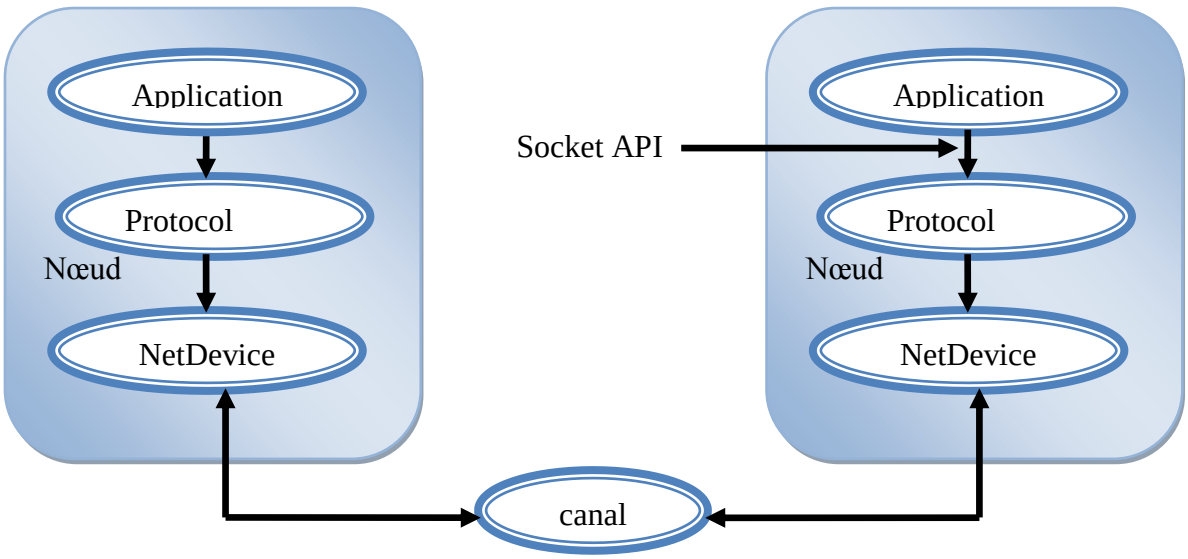


Figure 2.12 : Architecture du nœud NS3.

- **Module assistant**

Considéré comme un emballage de haut niveau. Il facilite la construction des scénarios complexes de simulation et l'installation des différents modules dans différents agents [26].

- **Module Internet-Stack**

Les classes de ce module définissent les protocoles TCP/IP des couches réseaux trois et quatre (TCP/IP) [36].

- **Module Dispositifs**

Les composants de ce type représentent des périphériques réseaux et transmettent des paquets via un canal virtuel [37].

6. Présentation détaillé de NS2

Bien que nous ayons commencé le travail en utilisant le simulateur NS3 (voire les étapes d'installation dans Annexe A) on était obligé par la suite de passer au simulateur NS2 pour la réalisation effective de notre travail. En effet, suite à l'installation de NS3, nous avons remarqué qu'il n'implémente pas encore le protocole SCTP. En recherchant plus auprès des forums, on a pu déduire qu'il y'avait un début d'implémentation de ce protocole mais non finalisée et son utilisation provoque des bugs. C'est pourquoi, on a opté pour NS2, vu que c'est un logiciel libre et relativement le plus proche des caractéristiques de NS3.

6.1. Architecture du simulateur NS2

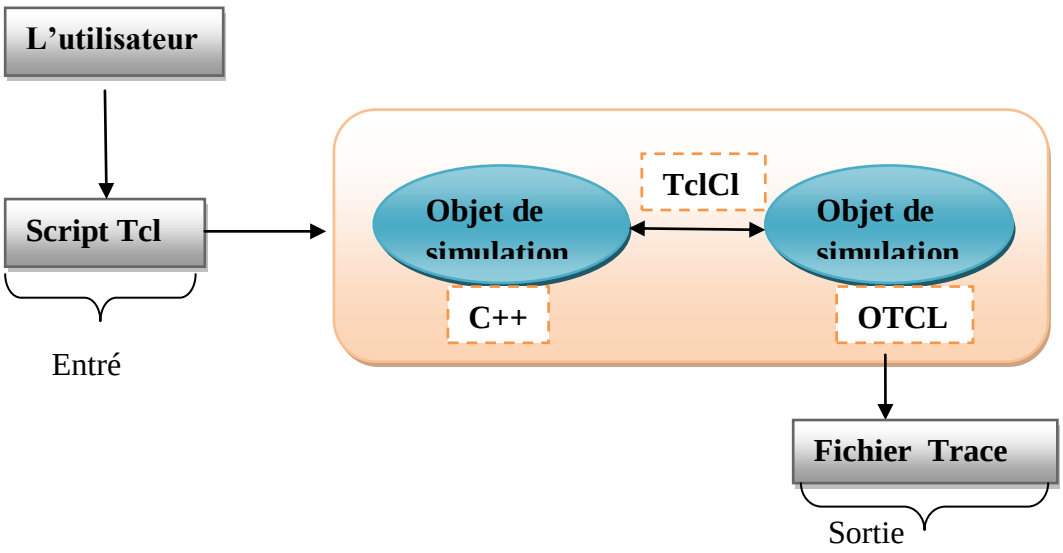


Figure 2.13 : Composants de système NS2**.**

L'architecture générale du NS2 est présentée sur la (figure 2.13), il consiste en deux type de Langage de programmation : le C++ et l'OTcl Le C++ est utilisé pour programmer les entités internes des systèmes simules, alors que l'OTcl est utilisé pour définir les scenarios des simulations et les paramètres de configuration. Ces deux types de langages sont, ensuite, lies via le Tclcl qui permet le passage des codes C++ vers les codes en OTcl.
Une fois la simulation terminée, NS2 produit deux fichiers de traces qui visualisent la dynamique des systèmes simules, et qui peuvent être interpréter en utilisant les outils: NAM et Xgraph [38].

6.1.1. Tcl

Tcl est un langage de commande. Il offre des structures de programmation telles que les boucles, les procédures ou les notions de variables. Il y a deux principales façons de se servir de Tcl:

Comme un langage autonome interprété ou comme une interface applicative d'un programme classique écrit en C ou C++. En pratique, l'interpréteur Tcl se présente sous la forme d'une bibliothèque de procédures C qui peut être facilement incorporée dans une application. Cette application peut alors utiliser les fonctions standards du langage Tcl mais également ajouter des commandes à l'interpréteur [39].

Toutes les applications qui utilisent Tcl créent et utilisent un interpréteur Tcl. Cet interpréteur est le point d'entrée standard de la bibliothèque. [27].

6.1.2. Langage de script OTcl

OTcl est une extension du langage de commande Tcl, qui utilise une programmation structurée (boucles, procédures, notions de variables) [40].

À travers ce langage, l'utilisateur décrit les paramètres de la simulation : topologie du réseau, nombre de nœud mobiles, le nombre de connexion, caractéristiques des liens physiques, protocoles utilisés, etc. La simulation doit d'abord être saisie sous forme de fichier texte que NS utilise pour produire un fichier trace contenant les résultats [41].

6.1.3. Langage C++

C++ qui constitue la partie centrale du simulateur (le noyau) et qui définit tout le mécanisme interne des objets de simulation.

6.1.4. Tclcl

(Tcl avec classes) est une interface Tcl/C++. C'est une couche de liaison entre le C++ de NS2 et le Tcl. Il permet de manipuler le script Tcl en C++ [40].

6.2. Composants du NS2

Pour créer une simulation il faut tout d'abord définir la topologie du réseau c'est à dire les nœuds et les liens. Puis définir les agents et les applications. Un agent est un objet de couche 4 du modèle OSI, cela peut être du TCP, UDP ou TCP Sink. Une application est génératrice de trafic sur le réseau. Nous pouvons par exemple utiliser du FTP, Telnet ou autre. L'étape suivante consiste à créer les connexions entre les agents puis de lancer les générateurs de trafic. Enfin la dernière étape consiste à définir le début et la fin de la simulation [40].

La liste des principaux composants actuellement disponible dans NS2 sont :

- **Application** (générateur de trafic) : Classe mère de toutes les applications (FTP, Telnet...).
- **Agent** (protocoles) : Classe mère des protocoles de niveau 3 et 4 (TCP, UDP).
- **Node** (nœud du réseau) : Représente l'ensemble des nœuds du réseau. Chaque nœud contient un queue de données (classifier) pour envoyer un paquet arrivant d'un agent ou d'une interface vers la bonne sortie.
- **Link** : lien entre les nœuds.
- **Monitor** : Elaboration de statistique sur un lien particulier [42].

Parmi ces composants on va voire la structure du Nœud :

Comme le montre la figure 2.14, un nœud est composé d'une entrée. Cette entrée reçoit tous les messages à destination du nœud. Lors de la réception d'un message l'élément appelé Addr Classifier permet de trier les messages en plusieurs destinations. Si le paquet a pour destination ce nœud dans ce cas celui-ci est transféré au module Port Classifier. Ce dernier a pour rôle de transférer le paquet à l'agent (et ensuite parfois l'application) pour laquelle il est destiné. Dans le cas ou le paquet n'a pas pour destination notre nœud, il est alors redirigé vers le lien approprié en fonction de la table de routage construite en début de la simulation [40].

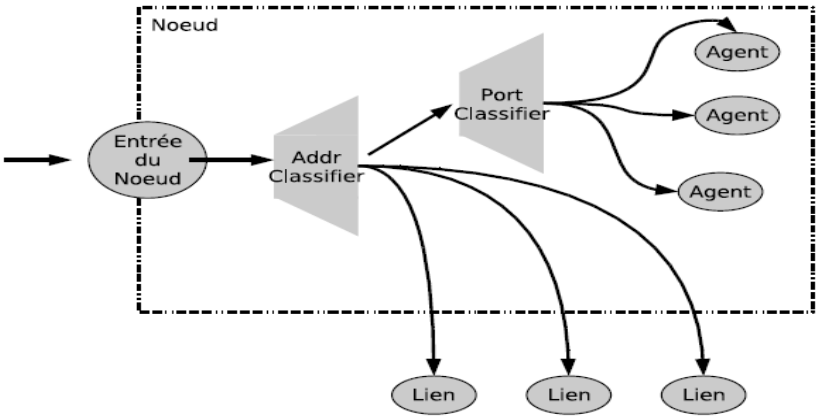


Figure 2.14 : Vue d'un nœud NS2 [40].

- **Classifier**

Un classifier est une sorte de démultiplexeur. Il a pour objectif de retrouver la référence d'un objet à partir du contenu du paquet. Par exemple le Port Classifier permet de transmettre le paqueta l'agent approprié [40].

- **Description d'un lien**

Un lien dans NS2 est un objet qui relie deux nœuds. Pour que les paquets puissent aller et venir dans les deux sens, il est nécessaire de créer un lien d'un nœud vers un autre et inversement. Fort heureusement des routines permettent de créer facilement ce type de lien double. Un lien a donc pour propriétés :

– un délai de propagation : c'est à dire le temps de parcours du lien par un paquet (latence).

– une capacité : la bande passante.

– un traitement de la file d'attente (DropTail par exemple).

6.3. Lien entre C++ et OTcl

NS est écrit en C++, Le package de NS2 inclus une hiérarchie de classes compilée dont les objets sont écrits en C++ et une hiérarchie de classes interprétée, dont les objets sont écrits en OTcl.

Il existe des passerelles entre le C++ et le Tcl. Comme le montre l'image, les objets créés dans OTcl sont également présents dans le C++. Il existe également un mécanisme de partage des variables entre le langage de script et le code C++ appelée bind. En utilisant ce procédé nous pouvons modifier directement la valeur d'une variable en C++ depuis le script [40].

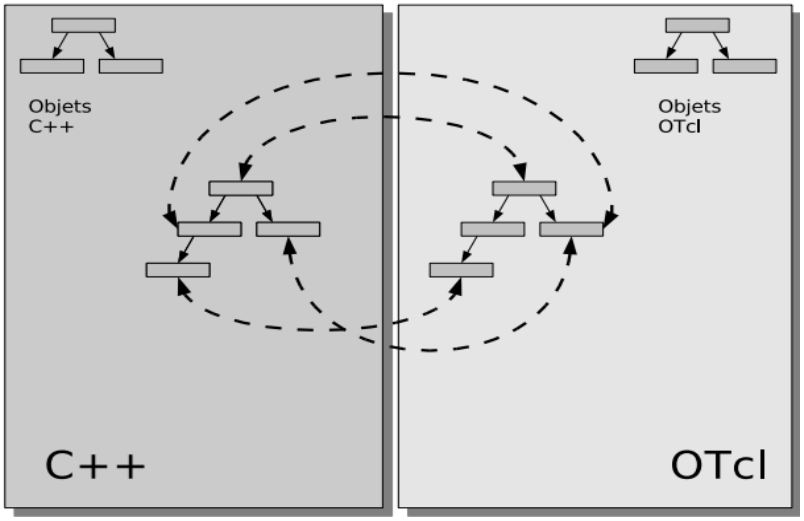


Figure 2.15 : C++ et OTcl, Dualité. **[40]**

6.4. Outils annexes

6.4.1. Outil de visualisation 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. Par exemple, il est capable 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 NS2, pour visualiser directement le résultat de la simulation [27].Xgraph, PERL, AWK, Gnuplot sont également des outils complémentaires à NS. Ils permettent une interprétation des fichiers de trace sous forme de graphes et de diagrammes.

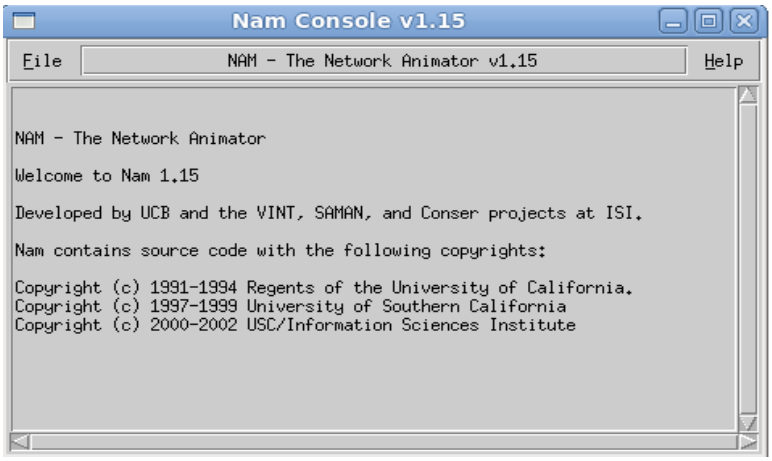


Figure 2.16: Interface graphique de NAM.

6.4.2. Outils d'analyse Awk, Xgraph

- **AWK** (Aho Weinberger Kernighan) : est un langage de traitement de lignes, disponible sur la plupart des systèmes Unix et sous Windows. Il est principalement utilisé pour la manipulation de fichiers textuels pour des opérations de recherches, de remplacement et de transformations complexes.

Le langage AWK permet d'extraire plusieurs informations à partir de fichier de trace comme par exemple le temps, le débit, le délai, le nombre de paquets reçus, etc [43].

La syntaxe de la commande AWK est :

gawk --file=[awk file name].awk [trace file name].tr

- ✓ --file :le code AWK est programmé dans un fichier.
- ✓ [awk file name]: c'est le programme AWK contenu dans un fichier .awk.
- ✓ [trace file name] : c'est le fichier de sortie .tr généré par NS2.

- **Xgraph :** est un programme d'analyse des résultats de NS. En effet, il modélise les informations existantes sur les fichiers trace(.tr) sous forme de diagramme de traçage (courbes, histogrammes,….).

Pour obtenir les graphes, à partir du fichier généré par la commande précédente, Il suffit d'exécuter la commande suivante, pour obtenir les graphes :

Xgraph [nom du fichier]

6.5. Chronologie du processus de simulation sous NS2

La mise en œuvre de ce simulateur se fait via une étape de programmation (configuration) qui décrit la topologie du réseau et le comportement de ses composants, puis vient l'étape de simulation proprement dite et enfin l'interprétation (analyse) des résultats.

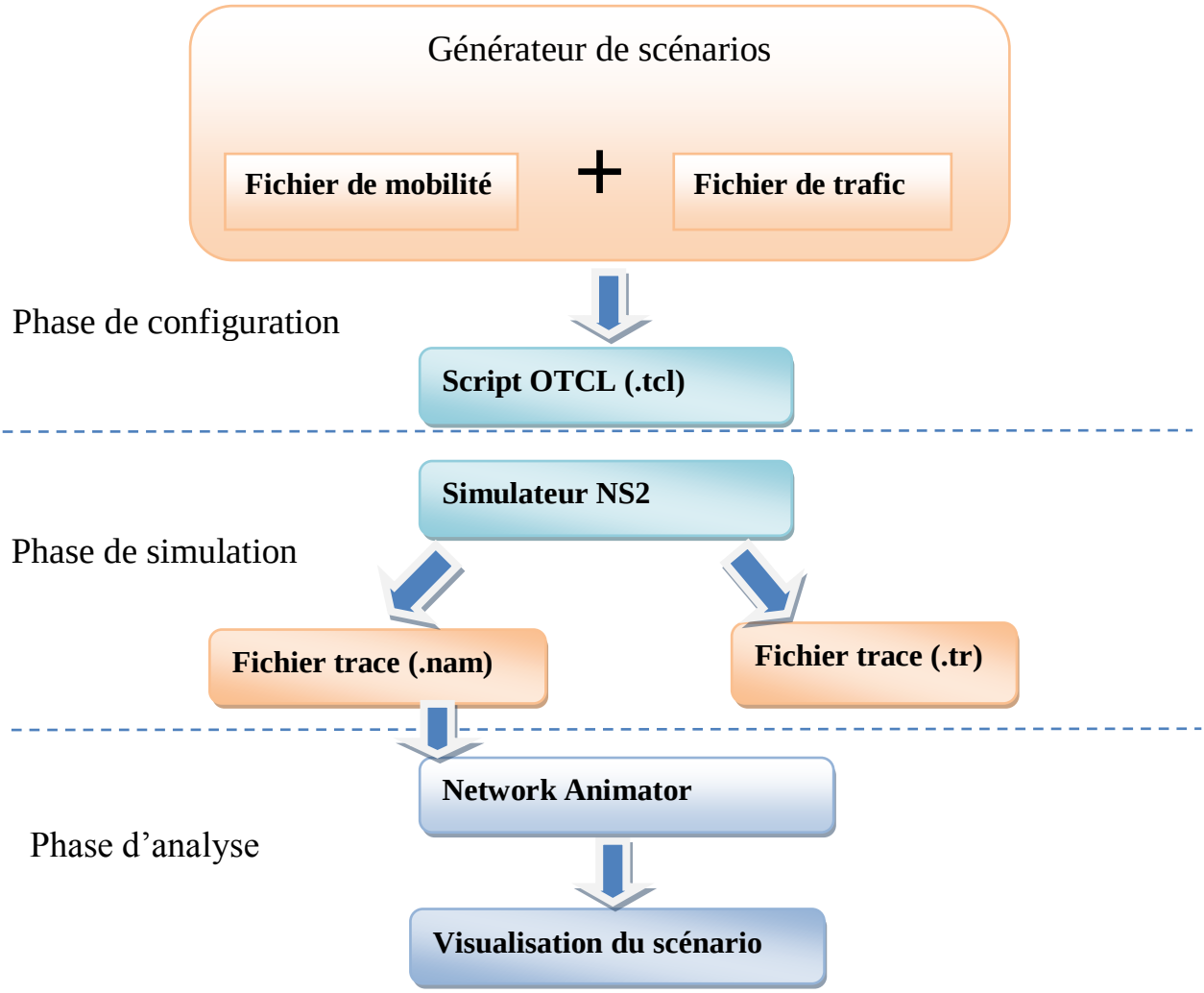


Figure 2.17 : Étapes de simulation avec NS2.

Phase de Configuration du réseau

Il s'agit de générer les fichiers d'entrées. Donc est une phase pour créer et configurer les composants du réseau.

L'utilisateur crée des scripts OTcl contenant les commandes nécessaires pour la mise en place de la simulation : le nombre des unités mobiles, le déplacement des unités mobiles, le choix des différents protocoles, le nombre, le type et la durée de divers transferts de données entre ces nœuds, etc.

Les fichiers d'entrée sont souvent classés en deux catégories : les fichiers de mobilité (qui décrivent le nombre de nœuds, leurs vitesses de déplacement, le temps de pause et les coordonnées du terrain) et les fichiers de communication (qui décrivent le trafic dans le réseau) [41].

Phase de simulation

S'occupe de l'exécution de scénario de simulation configuré dans la phase précédente. Pour cela, NS2 offre aux utilisateurs une commande exécutable (ns) qui prend le nom du script de lancement <nom_script>.tcl comme argument en entrée et produit en sortie deux fichiers de traces : <nom_script>.nam et <nom_script>.tr [41].

Phase d'analyse

Il s'agit de collecter et de compiler les résultats de la simulation. L'analyse du premier fichier trace <nom_script>.nam est effectué au moyen de l'outil de visualisation Network AniMator « NAM » fourni avec NS2. La tâche d'analyse du deuxième fichier de trace <nom_script>.tr est laissé à l'utilisateur [41].

7. Traces de NS2

Les fichiers traces avec l'extension **.tr** constituent la partie utile à l'étude des performances du réseau. Ils rapportent les résultats des simulations sous forme de colonnes qui représente chacune un résultat bien précis. Les valeurs des résultats par colonne varient selon des instants différents :

+ - d r événement	temps	à partir du nœud	Au nœud	Type de paquet	Taille du paquet	drapeaux	Id de flux

Adresse source	Adresse destination	numéro de séquence	Id unique paquet

Chaque ligne de trace commence par un événement (+, -, d, r) suivi du temps de simulation (en secondes) de cet événement, et à partir de et vers le nœud qui identifient le lien sur lequel l'événement a eu lieu. La prochaine information dans la ligne avant des drapeaux (est apparu comme «------» puisque aucun indicateur est défini) est le type de paquet et la taille (en octets). Le champ suivant est id de flux (fid) qu'un utilisateur peut définir pour chaque flux à l'entrée on script Tcl. Le champ fid est également utilisé lors de la spécification de couleur de flux pour l'affichage du NAM. Les deux champs suivants sont la source et l'adresse de destination dans les formes de "node.port" [44]. Le champ de numéro de séquence du paquet relatif au flux auquel il appartient et le champ Id unique paquet est un numéro unique qui permet de repérer un paquet au sein de l'ensemble des paquets de la simulation.

8. Conclusion

Dans ce chapitre, nous avons présenté quelques outils de simulation et leurs caractéristiques principales. Parmi ces simulateurs, on s'intéresse aux détails des simulateurs installés NS2 et NS3.

Chapitre 3

Extension au multi-streaming de SCTP

1. Introduction

Pour présenter notre extension au mécanisme de multi-streaming de SCTP, nous procédons comme suit :

- Étude approfondie des spécifications théoriques des transmissions parallèles au niveau transport. L'objectif étant d'introduire le principe de l'extension.
- Ensuite, nous présentons comment NS2 conçoit le mécanisme originel théorique du multi-streaming dans le code de SCTP.
- On définit enfin la conception de notre extension et comment l'intégrer à celle de SCTP sous NS2 de manière cohérente.

2. Diverses conceptions du multi-streaming

Dans ce qui suit, nous explorons comment cette idée de transmission parallèle a été traité, conçu et utilisée sur divers protocoles de transport.

De manière générale, la transmission via différents flux de données est basée sur l'une des approches [45] présentées ci-après. Dans chacune, les transmissions parallèles s'opèrent sur un exemple de trois streams.

2.1. Transmission parallèle sous TCP/UDP

Dans une première approche, l'émetteur A ouvre plusieurs connexions, vers le récepteur B. Bien que cette approche permet une séparation logique des données, généralement basée sur leur type, ce genre de connexions multiples affecte le mécanisme de contrôle de congestion de TCP et attribue à une application une bande passante plus large et ce au détriment des autres applications et d'autres flux de données sur le réseau [45].

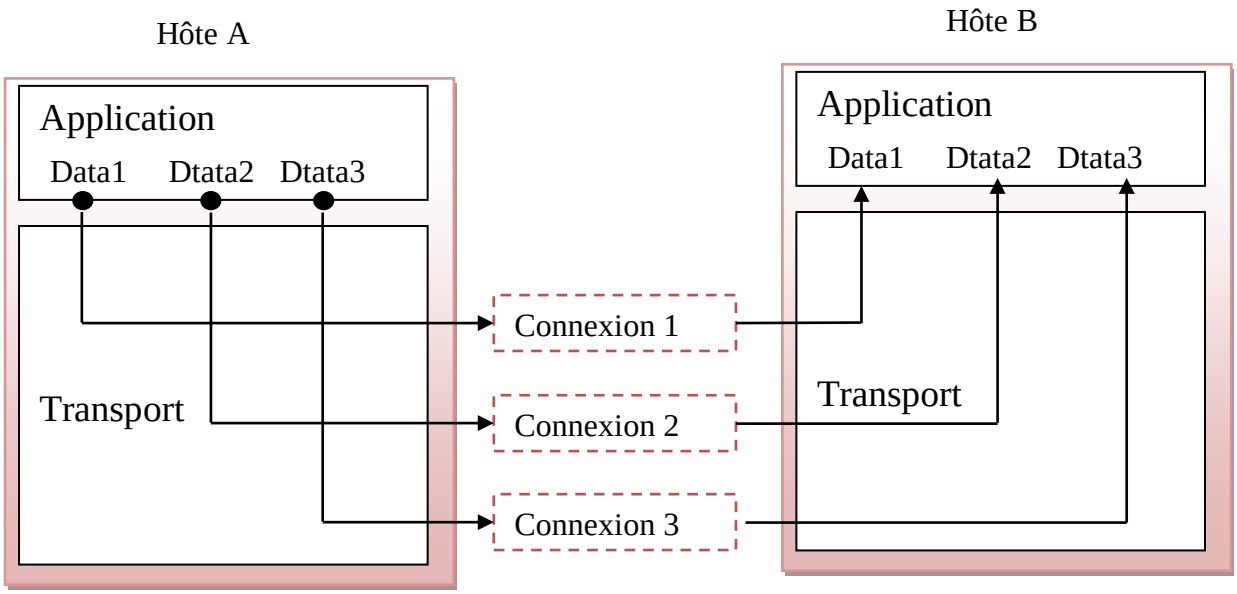


Figure 3.1 : 1ère Approche traditionnelle de transmission parallèle de données.

Dans une deuxième approche, l'hôte A opère un multiplexage/démultiplexage des trois types de données sur une seule connexion au niveau de l'application. Cette approche n'affecte pas le mécanisme de contrôle de congestion de TCP. Cependant, elle augmente la complexité pour le programmeur de l'application, puisque cette dernière doit, elle-même, assurer efficacement la procédure de transmission de données ainsi que sa bonne gestion [45].

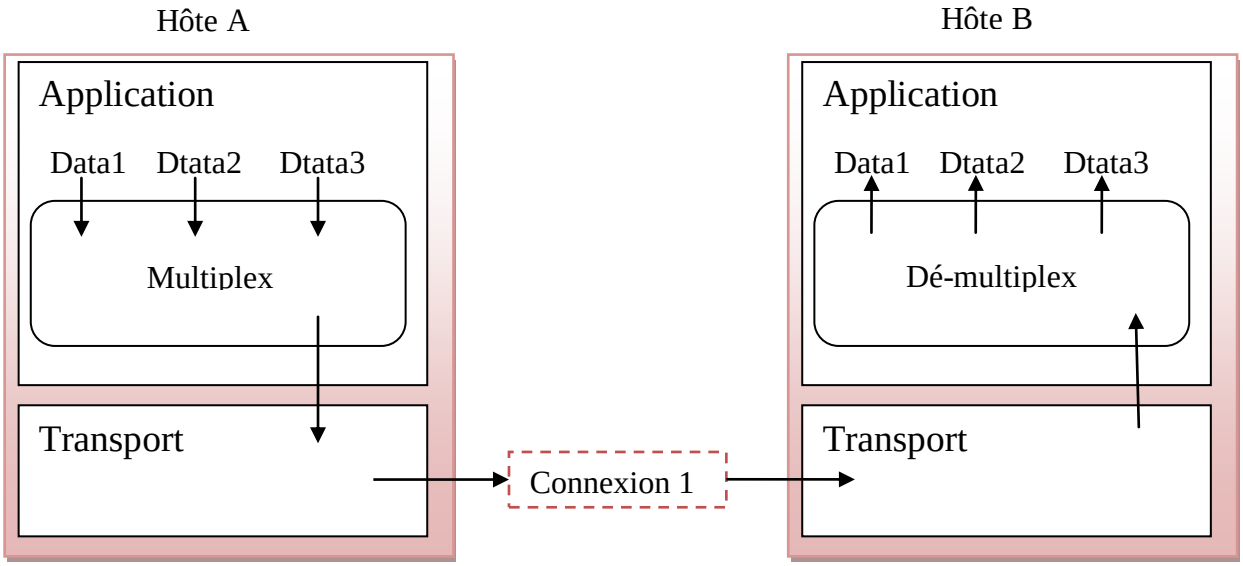


Figure 3.2 :2ème Approche traditionnelle de transmission parallèle de données.

Quant qu'à la troisième approche, elle concerne plutôt le protocole UDP. Cette approche est similaire à la deuxième. Cependant, les programmeurs d'application doivent assurer leur propre service de fiabilité aussi bien que leur propre multiplexage/démultiplexage. Ceci est dû au fait que UDP offre un service sans connexion [45].

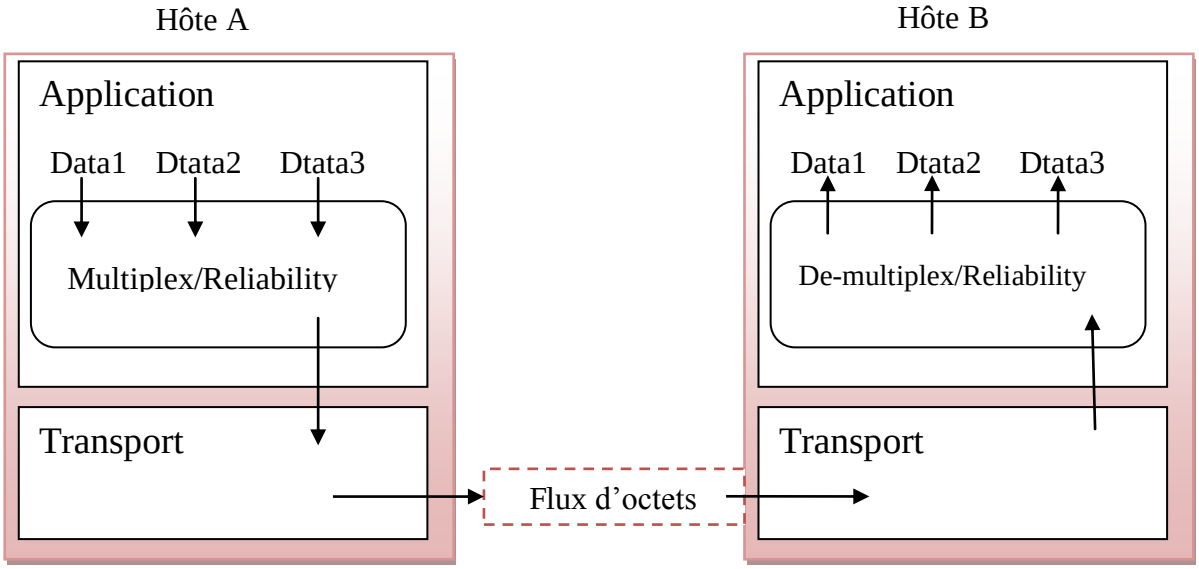


Figure 3.3 : 3ème Approche traditionnelle de transmission parallèle de données.

2.2. Multi-streaming originel du SCTP

L'introduction du protocole SCTP et du concept de stream, permet aux applications une transmission parallèle des données sur une même association. Cette nouvelle approche combine les avantages de connexions multiples de bout en bout et le multiplexage/démultiplexage des applications.

Un point terminal SCTP (client ou serveur) peut activer plusieurs streams sortants ou entrants lors de l'établissement d'une association (Figure3.4). Chaque stream est muni de sa propre mémoire tampon (buffer) indépendamment des autres streams pour envoyer et recevoir des données. Les streams sont actifs tout au long de la durée de vie de l'association SCTP [50].

Cependant, les concepteurs du protocole SCTP dans leurs recommandations ont juste précisé qu'une implémentation de SCTP doit utiliser un algorithme de gestion de l'allocation des streams et de leur utilisation avec comme unique considération le respect des délais de transmission. Deux algorithmes ont été proposés dans ces recommandations : le RR (Round-Robin) et le FCFS (First-Come First-Serve).

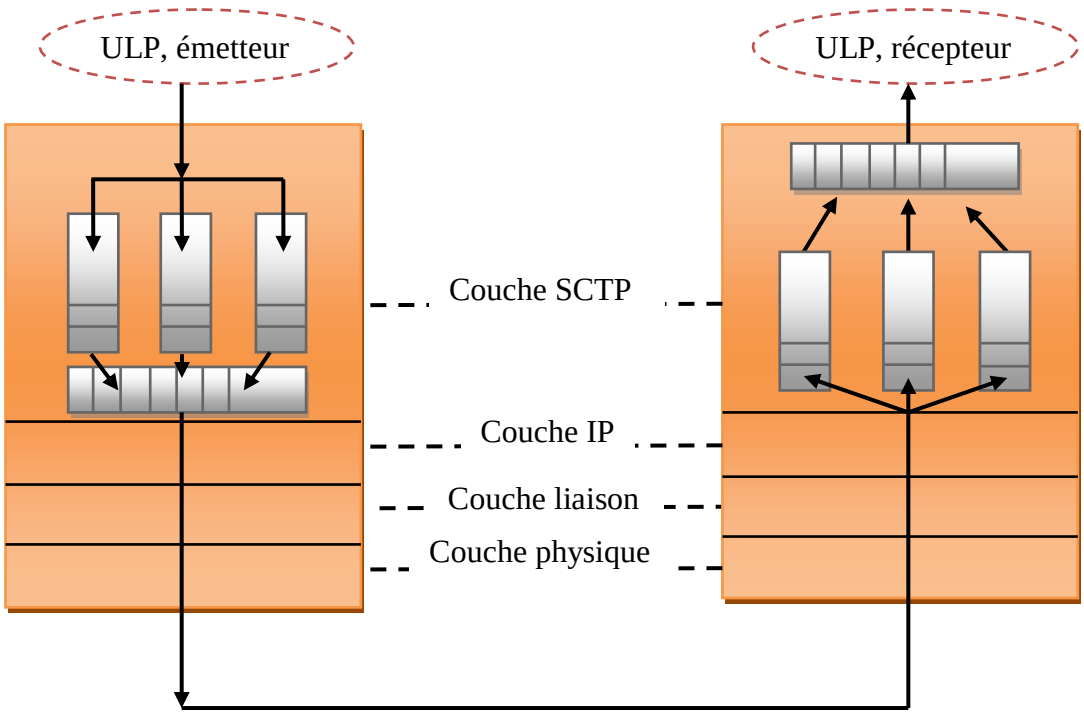


Figure 3.4:Exemple d'une association SCTP (*multi-streaming* activé).

- **Politique RR**

Cette politique est utilisée pour gérer le passage des chunks de la couche transport vers la couche réseau(Figure3.5). Le principe de RR établit qu'un élément est pris de la tête de chaque file d'attente d'émission à tour de rôle. Ceci correspond sous SCTP à prendre un chunk du premier stream, ensuite, si le client peut envoyer encore des données, tel le permettent la taille de la fenêtre de réception du serveur et la taille de la fenêtre de congestion du client, il envoie un chunk de données à partir de la file d'attente d'émission du stream 1 et ainsi de suite jusqu'à atteindre le dernier stream. Le RR reprend alors sur le premier stream. Ce processus de round-robin continue tout au long de la durée de vie de l'association entre les points terminaux de SCTP.

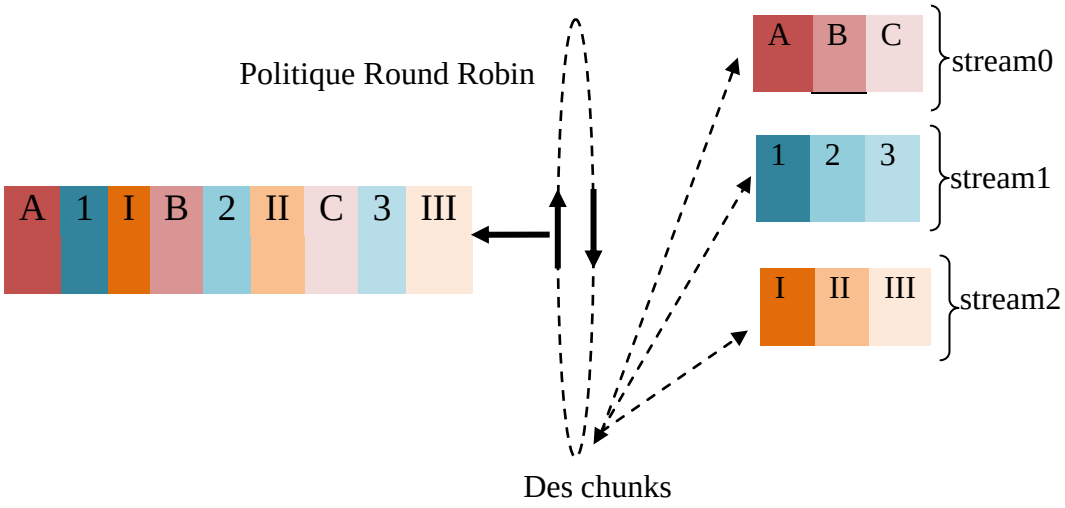


Figure 3.5 : Politique d'ordonnancement RR.

- **Politique FCFS**

Dans l'autre sens, à l'arrivée des données de la couche réseau vers la couche transport, les chunks sont indiqués par les streams identifiés dès leur envoi .Avec une politique FCFS, SCTP transmet chaque chunk dans l'ordre de sa réception dans la file d'attente du stream correspondant.

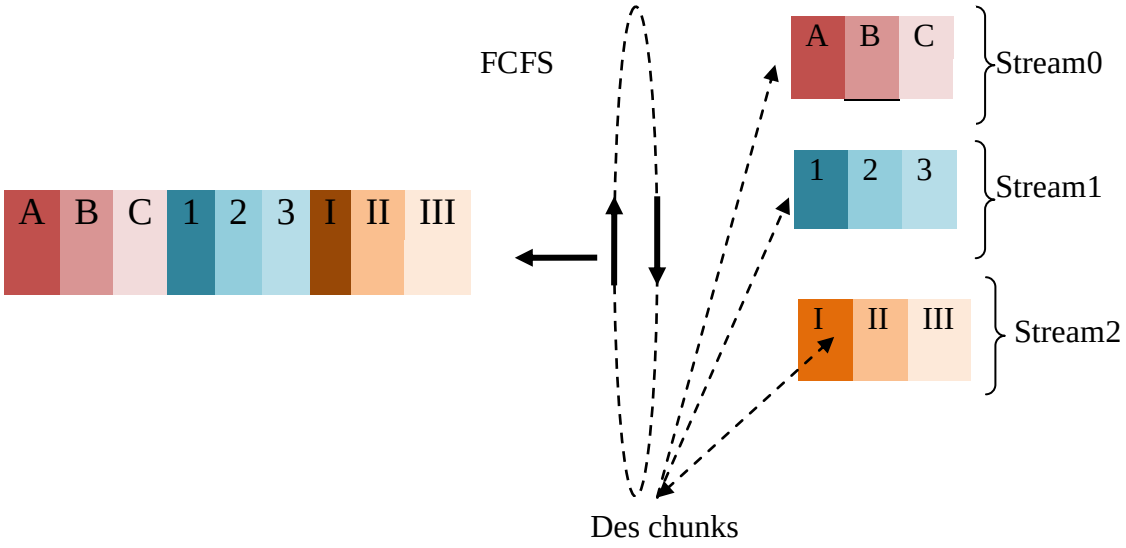


Figure 3.6 :Politique d'ordonnancement FCFS.

2.3. Quelques travaux sur le multi-streaming

SCTP, par son mécanisme de multi-streaming permet de résoudre les problèmes de pertes et de livraison de données apparus avec le TCP via deux niveaux de numérotation. Une numérotation au niveau association avec le TSN et une autre au niveau du stream avec le SSN. Si une perte se produit au niveau d'un stream donné, elle n'affecte pas la livraison des chunks de données contenus dans les autres streams. Ce qui permet de réduire les retards de transmission causés par les problèmes de head-of-line Blocking [46].

Cependant, ce mécanisme peut être utilisé en termes de différenciation du type des données transmises. Des études antécédentes ont certes travaillé dans ce sens et ont abouti à de meilleures performances des transmissions.

Les auteurs du [47] proposent une extension pour SCTP basée sur le fait que dans les réseaux de transmission de données et multiservices sans fil tels que l'UMTS, les applications multimédias devraient bénéficier de plus de bande passante que les autres applications. Donc, l'extension modifie la structure de bloc de données SCTP dans le but d'introduire une gestion des priorités inter-flux qui attribue à chaque flux un niveau relatif d'importance qui favorise un certain type de données sur les autres. Cette extension contribue à améliorer les performances du protocole dans les réseaux multiservices.

Dans une autre étude présentée en [46], les auteurs proposent d'augmenter la fonctionnalité multi-streaming de SCTP avec la programmation enfichables à ordonnancement multi-flux de SCTP pour satisfaire leurs exigences spécifiques à l'application. Ils mettent en œuvre leur proposition dans le noyau Linux et montre que cette extension augmente grandement la flexibilité de SCTP et apporte des améliorations de performances visibles. Ceci est le premier travail qui démontre l'efficacité de SCTP enfichables multi-flux ordonnancement à travers les implémentations réelles [46].

3. Rétro-conception de SCTP à partir du code NS2

Il s'agit dans ce titre de comprendre la conception du multi-streaming à partir du code SCTP sous NS2.C'est une étape connu en génie logiciel, dite rétro-conception et qui consiste à comprendre une conception dans un domaine donnée en utilisant l'implémentation correspondante.

L'objectif n'étant pas de maîtriser le savoir-faire de cette étape, mais de justifier la contrainte de compréhension du code SCTP. En effet, l'intérêt de notre travail étant de proposer une extension à SCTP mais surtout, qui puisse être implémentée sous NS2 et donner des résultats en respectant les délais de notre étude.

Autrement dit, même si l'implémentation de SCTP sur NS2 est relative aux spécifications théoriques, il est important de bien maîtriser comment et à quel point elle les satisfait réellement.

D'ailleurs, en étudiant le protocole SCTP sous NS2 on a trouvé qu'il distingue entre différents types de données applicatifs: des données provenant d'application FTP et des données provenant d'application connaissant les principes de SCTP, référencées dans ce qui suit par SctpApp1ainsi que d'autres types de données applicatives non spécifiées. Les traitements de ces types comportent des parties similaires et d'autres plus spécifiques. Notons que cette distinction n'est absolument pas discutée théoriquement.

3.1. Traitement des données FTP

La couche application envoi les données à la couche transport SCTP sur un seul flux(Figure3.7).Au niveau SCTP, dès la phase d'établissement de l'association, l'agent SCTP sous NS répartie déjà les données sur les streams. Ensuite, une fois l'association établie, pour les envoyer vers la couche inférieure, SCTP fait un passage par tous les streams pour préparer les chunks en utilisant un algorithme de RR.

SCTP met ensuite ces chunks dans un buffer commun de façon à prendre un seul chunk d'un stream puis passer au suivant (Figure3.5). Cette opération est répétée jusqu'à la fin de la durée de vie de l'association.

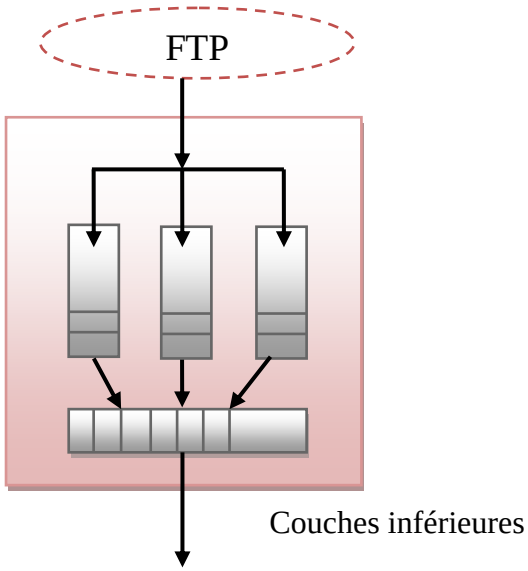


Figure 3.7 : Multi-streaming en RR pour transfert de données de type FTP.

Dans le sens inverse, en réception de paquet de la couche réseau à la couche transport en SCTP, les chunks sont remis dans les streams correspondants indiqués dans l'envoi : l'agent SCTP utilise un algorithme FCFS(Figure3.6).

3.2. Traitement des données SctpApp1

Pour le transfert des données provenant d'application SctpApp1, censé connaître le mécanisme de multi-streaming, leurs transferts vers la couche transport ne prend absolument pas en compte la multitude de streams, donc il utilise un seul stream (voir lafigure3.8).

Quand les données arrivent à l'agent SCTP, il les pose également dans un unique stream, ensuite les chunk de ce stream sont déplacés dans un buffer qui envoie les paquets SCTP à la couche inférieure.

Dans l'autre sens de transmission, la couche réseau envoi ces paquets à la couche transport et remet les chunks dans l'unique stream de l'association.

Pour envoyer les chunks à l'application SctpApp1, l'agent SCTP sous NS utilise le même algorithme de FCFS (Figure 3.6) qui dispose ces chunks dans un buffer commun.

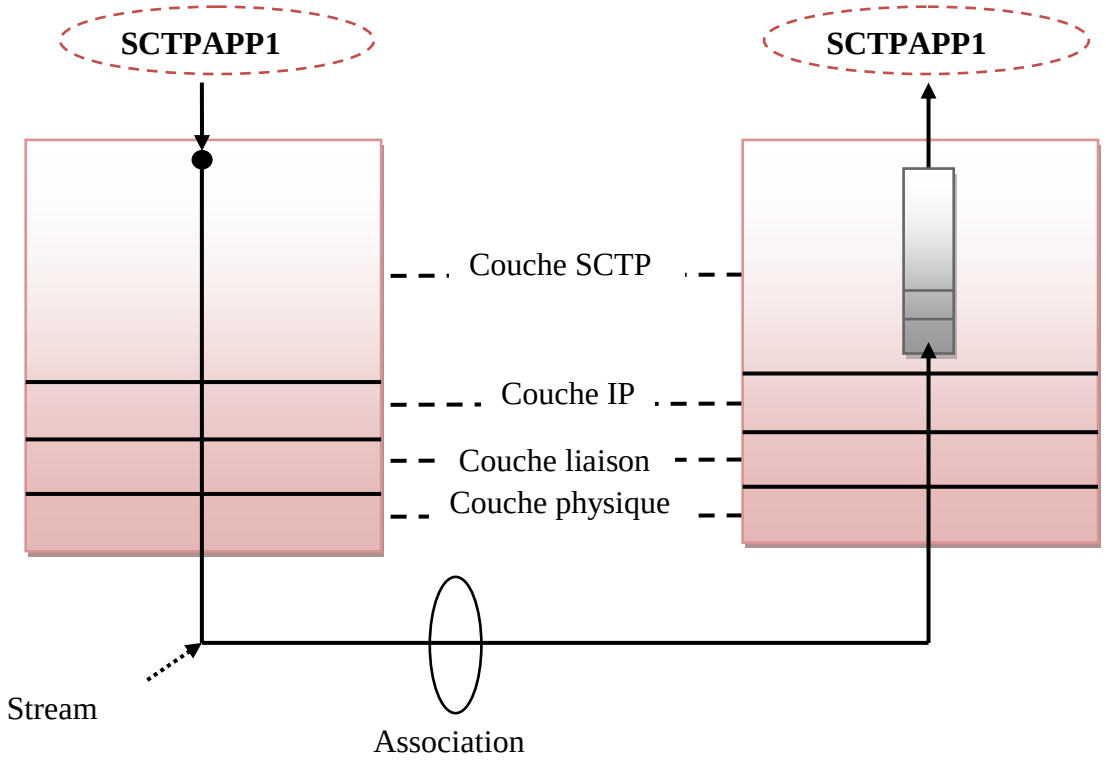


Figure 3.8: Transfert de données de type SctpApp1.

Notons que l'agent SCTP prépare une certaine conception du multi-streaming pour d'éventuels développements dans les recherches. En effet, l'agent SCTP a prévu l'utilisation de plusieurs streams.

Cependant, d'une part, le transfert des chunks à partir de ces streams est implémentée de manière très basique, à l'aide d'un algorithme d'ordonnancement similaire à l'algorithme utilisé lors du transfert des données de type FTP. D'autres parts, l'application SctpApp1 de NS, supposée connaitre le multi-streaming ne contient aucune spécification de son utilisation.

Dans ce qui suite nous essayons de mieux expliquer la conception et le fonctionnement de SCTP sous NS2 à travers des diagrammes qui expliquent les méthodes déployés dans le code.

3.3. Traitement des messages de la couche application à SCTP

Ce traitement se fait par la définition et invocation de la méthode Sendmsg () au niveau de la couche transport et de la couche application respectivement. C'est une procédure permet l'envoi des données vers la couche transport et prend deux paramètres : iNumBytes est la taille des données et le cpFlags c'est les données en entrée.

Le traitement des données se fait sur 3 étapes :

Dans la 1ère partie , SCTP va tester sur la source des données : est-ce des données applicatives relatives à l'utilisateur ou des données de contrôle et accorde un traitement particulier à chaque cas.

Dans la 2ème partie, si les données ne sont pas de contrôle, il va tester si ces données de l'application sont valides (différent de null).

En 3ème partie, le traitement se fait en deux cas selon l'état de l'association (fermée / établit) : si l'association est encore fermée, SCTP détermine la taille de données par la fonction GenChunk (INIT) et déclenche un temporisateur lui permettant de réémettre ce paquet (contenant le chunk INIT) en l'absence de réponse. Ensuite le client à se met en attente du paquet de réponse contenant le cookie.

```
iOutDataSize = GenChunk(SCTP_CHUNK_INIT, ucpOutData);
    opT1InitTimer->resched(spPrimaryDest->dRto);
eState = SCTP_STATE_COOKIE_WAIT;
SendPacket(ucpOutData, iOutDataSize, spPrimaryDest);
```

Alors que si l'association est établie, SCTP se prépare à l'envoi des données à la couche inférieure avec la procédure *SendMuch();*

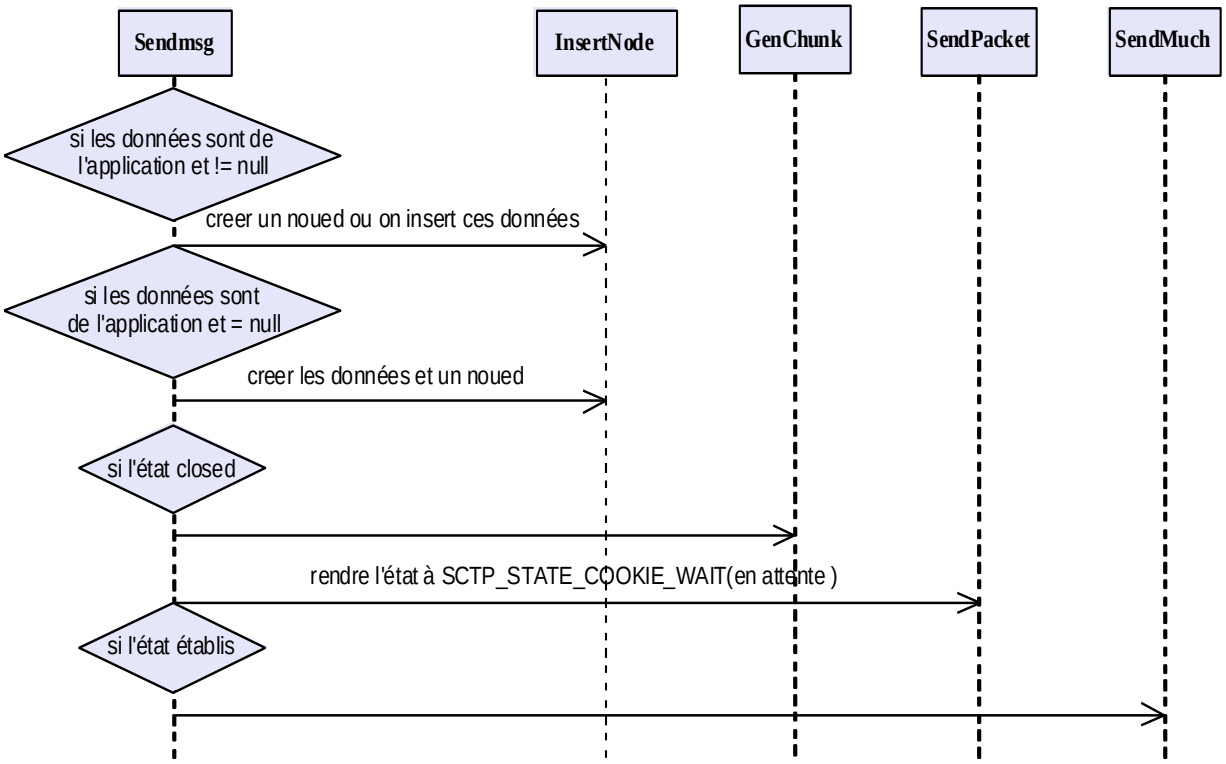


Figure 3.9:Digramme de procédure Sendmsg.

3.4. Transmission de SCTP aux couches inférieures

Comme nous avons vu précédemment, la procédure Sendmuch est déclenchée si l'association est en état établi, elle permet l'envoi des chunks à partir de buffer vers la couche basse.

Pour envoyer les données il faut tester que la taille de ces derniers est cohérente (données erronées) et inférieure à la taille de Cwnd.

while((spNewTxDest->iOutstandingBytes<spNewTxDest->iCwnd) && (eDataSource == DATA_SOURCE_INFINITE || sAppLayerBuffer.uiLength != 0))

puis la méthode calcule et contrôle la taille de données de chunks cumulés dans un paquet SCTP avant de les envoyer.

iOutDataSize = BundleControlChunks(ucpOutData);
iOutDataSize += GenMultipleDataChunks(ucpOutData+iOutDataSize, 0);

Le diagramme suivant apporte plus de précisions quant au traitement des données entre méthodes du code NS2.

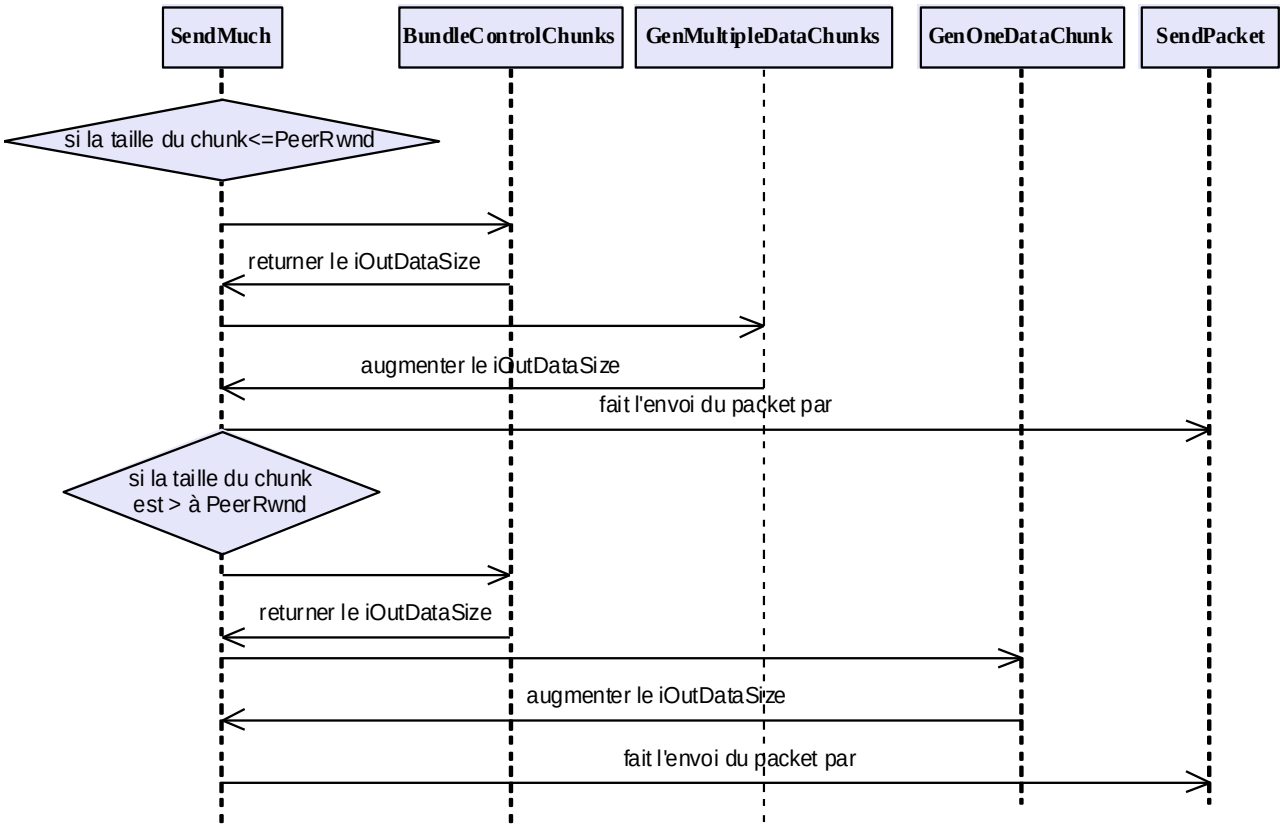


Figure 3.10:Digramme de procédure SendMuch.

3.5. Traitement des paquets de la couche inférieure à SCTP

En examinant le sens contraire des transmissions, i.e. de la couche inférieure à un stream de SCTP. La procédure Passtostream permet d'affecter un chunk qui arrive, elle prend un seul paramètres *(spChunk)* .

Dans cette procédure il y a 2 partie, :

La première partie, SCTP se pointe sur le stream identifié par le chunk reçu. Autrement dit, le chunk en réception indique dans son entête le numéro du stream d'où il provient. Ce numéro est pris en considération pour se pointer sur la queue du buffer du stream. Cette conception utilise une structure de nœud, et l'ensemble des streams sont représentés par un tableau dont chaque case pointe sur un stream

*(SctpInStream_S *spStream = &(spInStreams[spChunk->usStreamId]))*,

Dans la 2ème partie, SCTP teste si le chunk en entrée appartient à un stream ordonné : Dans un premier cas, si le steam n'exige pas l'order de ces chunks, alors le chuk en réception sera directement transmis à la couche supérieure et un acquittement de réception est généré :

if(spChunk->sHdr.ucFlags& SCTP_DATA_FLAG_UNORDERED)

{*PassToUpperLayer(spChunk);*

Dans le deuxième cas, si le stream en question est ordonné, l'agent SCTP insère le chunk dans le buffer du stream selon son ordre.

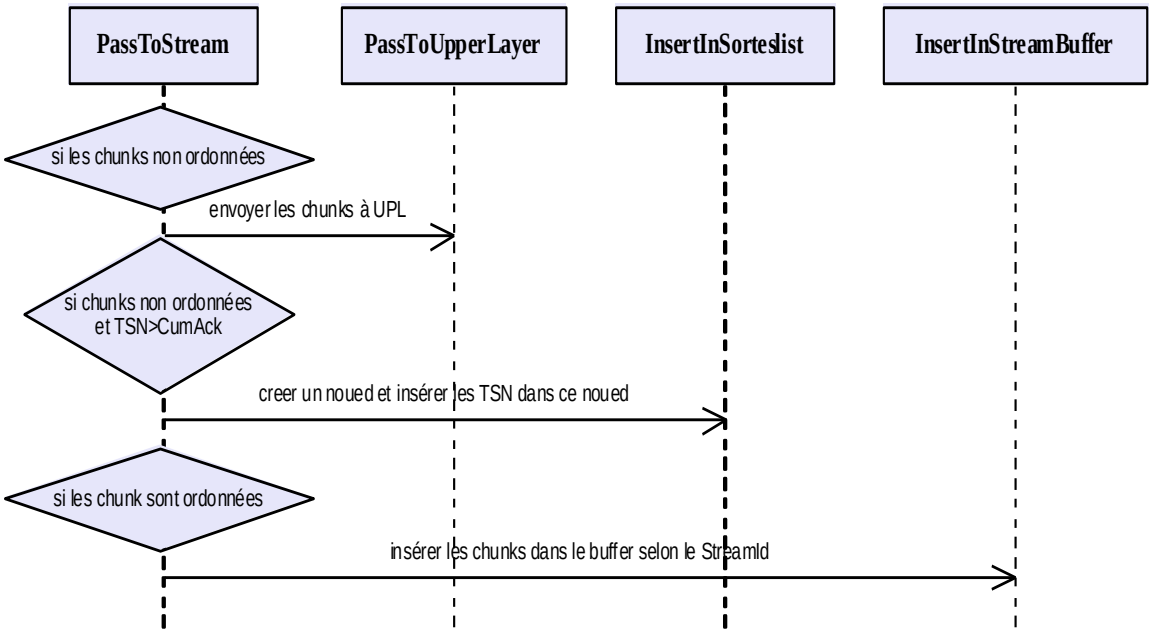


Figure 3.11:Digramme de procédure PassToStream.

3.6. Transmission des données de SCTP à l'application

La transmission des chunks vers la UPL (Upper layer) se fait par un RR modifié : la boucle assure de passer par chaque stream du 1ère jusqu'au dernier mais dans chaque passage il va dépiler le contenu du stream en entier. Pour ce faire, le code SCTP utilise un buffer des chunks pour y dépiler les chunks.

*for(i = 0; i<iNumInStreams; i++){*pour fait le passage par les streams.

spStream = &(spInStreams[i]);

spCurrNode = spStream->sBufferedChunkList.spHead;

while(spCurrNode != NULL)

Mettre le chunk dans *le BufferedChunk*.

Dans la méthode qui assure l'envoi des chunks vers la couche applicative, il existe des tests relative à la fiabilité des streams (si le stream exige ou pas les acquittements) mais on ne s'intéresse pas par cette partie puisque dans notre conception on considère que tous les streams sont fiables.

Si les streams sont fiables, il fait deux testes et selon ces deux testes soit il envoi les chunks vers UPL.

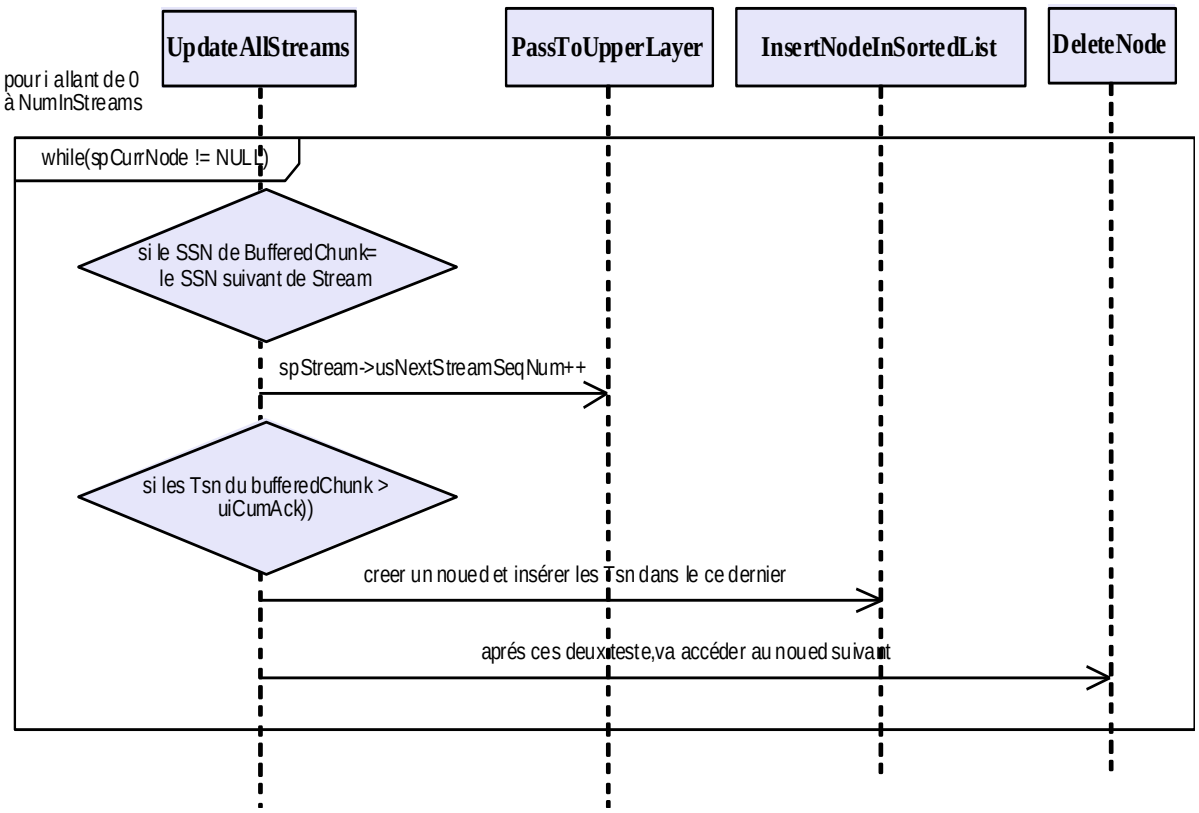


Figure 3.12:Digramme de procédure UpdateAllStreams.

Nous proposons dans ce qui suit d'intégrer de nouvelles mesures pour permettre la multitude des flux de transmission, en termes d'identification et de traitement.

4. Problématique

D'après les études que nous avons établies en sections précédentes, nous déduisons que :

Le multi-streaming peut être déclenché tant au niveau transport qu'au niveau applicatif, à condition de maintenir une utilisation cohérente de la bande passante entre les applications en cours et d'éviter des complications supplémentaires au niveau applicatif dues à la gestion cohérente de divers flux.

Sur SCTP de NS, les messages générés au niveau applicatif sont transmis à SCTP sur un unique flux. Le nombre de flux utilisés au niveau SCTP peut être précisé dès l'établissement de l'association. Il est compté à un seul flux, par défaut, relativement au flux applicatif qui ignore le mécanisme de multi-streaming et même pour les applications supposées connaître ce mécanisme. Cependant, ce nombre peut être étendu à plusieurs flux.

Pour les données de type FTP, lors de l'envoi de ces données de la couche application vers la couche réseaux, un algorithme de RR se met on œuvre, la défaillance de cet algorithme est qu'il passe par tous les streams de manière identique par la sélection d'un seul chunk par stream à chaque passage.

Pour le type de données SctpApp1 le transfert des données vers la couche transport ne prend pas en compte le multi-streaming même si l'agent SCTP a prévu de considérer l'utilisation de plusieurs, donc il utilise une seul stream et ne fait pas une distinction entre les flux.

Les messages provenant de la couche application à la couche SCTP varie de type et contraintes de transmissions. Ces différences peuvent être considérées et traitées via le mécanisme de multi-streaming dès le passage de la couche application vers la couche SCTP.

En d'autres termes, bien que SCTP soit un protocole dont les spécifications théoriques présentent des opportunités dans le traitement de la qualité de services des transmissions, particulièrement en termes de différenciation de services, les détails de sa conception et même son implémentation sous NS2 ne les assurent pas convenablement. Ce qui nécessite des extensions conceptuelles tant au niveau de l'application qui déploie SCTP qu'au niveau de SCTP même.

5. Solution proposée

Nous proposons dans ce qui suit des extensions qui permettent la différenciation des services transport selon différents types d'application par une distinction entre les flux transmis, à la fois en termes de traitement et d'identification.

5.1.En termes d'identification

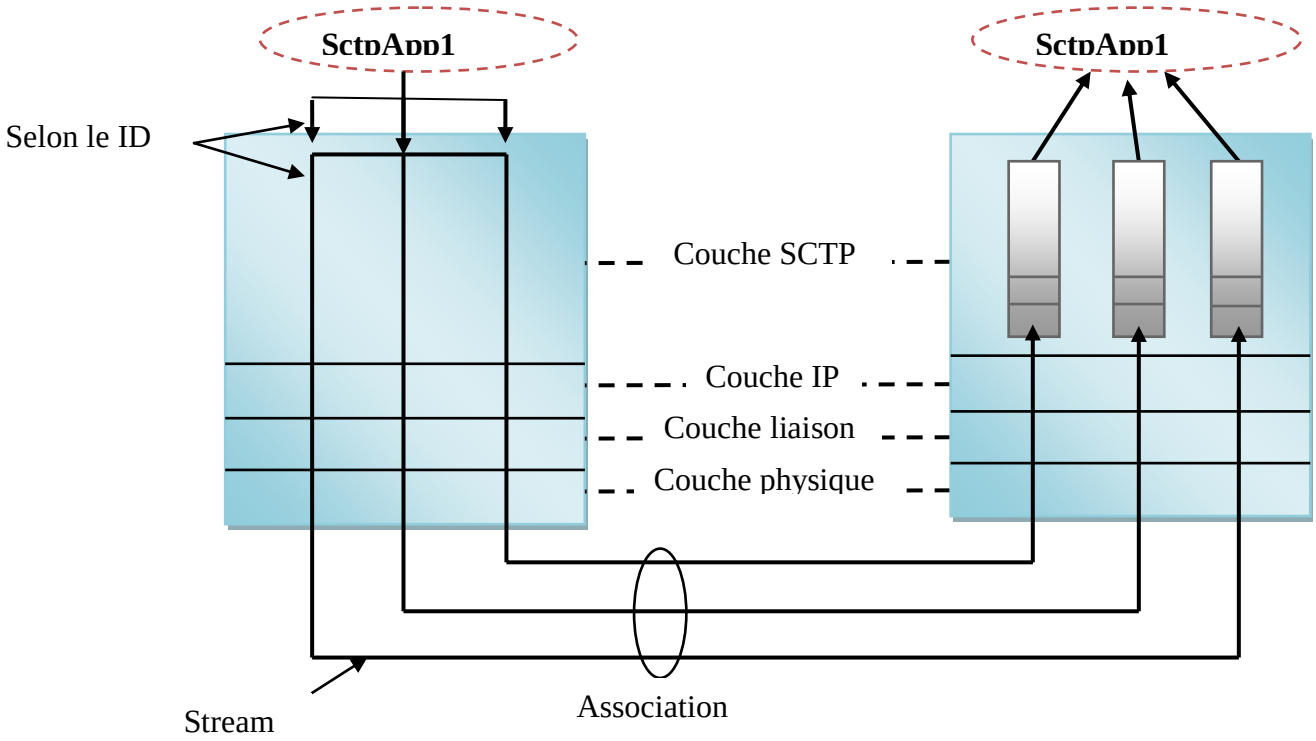


Figure 3.13:Notre extension dans le transfert de données de type SctpApp1.

Il s'agit là de déterminer comment les messages applicatifs ou chunks au niveau transport sont déposés sur les streams avant même de les traiter. Evidemment, si l'association SCTP n'utilise qu'un seul stream, tous les messages applicatifs ou les chunks de SCTP y sont déposés. Cependant, lorsque l'association comporte plusieurs streams, il faudra alors définir pour chaque message de données, sur quel stream il sera déposé.

Cette décision peut être prise par SCTP lui-même, dans ce cas le problème est réduit à un RR par tous les stream pour y déposer un unique chunk à chaque tour. D'autres algorithmes peuvent être considérés selon les objectifs d'utilisation du multi-streaming.

Cependant, il est plus intéressant que l'identification des streams soit faites au niveau application et ce pour assurer un traitement particulier pour chaque stream selon le type de données qu'il comporte et dont l'application en a connaissance.

En effet, l'application qui communique au niveau transport avec SCTP et connait ses spécifications et les utilise doit se limiter à ses caractéristiques. Entres autres, il s'agit de la taille des messages envoyés (pour éviter leur fragmentation), ainsi que particulièrement dans le contexte de notre travail, d'identifier pour les messages de données le numéro du stream (SID) qui recevra le message en question.

Nous avons opté dans notre travail pour cette deuxième éventualité : c'est la couche application qui doit déterminer une distinction entre les données et affecte chaque message à son stream. Notons que des travaux sur les flux Internet démontrent que les données en transmissions sont classées principalement en trois types de données : audio, vidéo et texte [42] D'autres études préfèrent étendre ce classement à cinq types afin de mieux distinguer les contraintes relatives.

Notons que notre travail dans ce cas ne concerne pas la réalisation d'application qui manipule différents types de données. Nous opérons plutôt au niveau du SAP, c'est-à-dire sur la frontière entre les couches application et transport.

En termes d'identification, on s'intéresse aux données applicatives SctpApp1, qui peuvent utiliser les mécanismes de l'agent SCTP par conception. On distingue ces données par un mécanisme exploitable de manière générale (par exemple pour implémenter la qualité de service, améliorer le handover, …etc). Il s'agit de déterminer à quelle stream appartient chacune de ces données avant d'être envoyée au niveau du transport. Ceci est fait en utilisant une identification du stream au niveau applicatif.

5.2. Traitement des chunks sur les streams

L'algorithme d'ordonnancement que nous proposons repose sur la pondération du nombre de chunk transmis de chaque stream par rapport au nombre total de chunk transmis dans une association pendant un certain intervalle de temps. Il assure un passage par tous les streams de l'association et pour chaque stream, il transmet un nombre différent de chunks. C'est un de Round Robin pondérée par le nombre de paquet à partir de chaque stream.

Évidemment, le stream pour lequel le nombre de chunks transmis sur un tour est supérieur est le stream dont les données présentent des contraintes plus importantes.

Selon notre algorithme d'ordonnancement proposé, le client choisit 3chunk de données à partir de la file d'attente d'émission du stream 0 puis il envoie 2chunk de données à partir de la file d'attente d'émission du stream 1. Un seul chunk de données sera envoyé par la suite, à partir de la file d'attente d'émission du stream 2.

Le choix de pondération plus importante dans l'ordre des numéros de streams (de 0 à 2) est conséquence directe de l'ordre des numéros de streams dans les transmissions via le code de SCTP sous NS2.

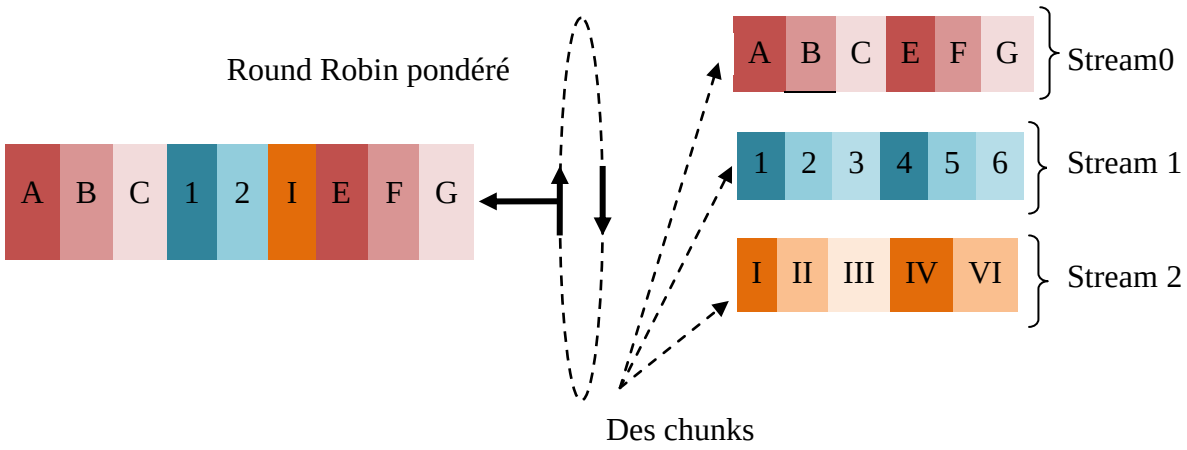


Figure 3.14 : Algorithme d'ordonnancement proposé.

Ce principe pour la sélection des chunks entre streams de données est appliqué aussi bien pour les applications FTP que pour les applications SctpApp1. A une différence près, est que l'application SctpApp1 identifie déjà les streams et donc les chunks sur les streams selon leurs caractéristiques alors que pour FTP, le nombre de streams et les chunks associés sont déterminés au niveau de SCTP uniquement.

6. Diagrammes récapitulatifs

Les diagrammes sur les figures suivantes constituent un récapitulatif qui explique l'interaction entre les couches application, transport et réseau et place notre extension dans ces interactions.

Le 1er Diagramme représente le passage d'un chunk de la couche application vers la couche réseau et le 2èmediagramme, il montre les séquences du passage d'un chunk de la couche réseau vers la couche application. Dans les deux diagrammes, l'encadré rouge présente nos modifications.

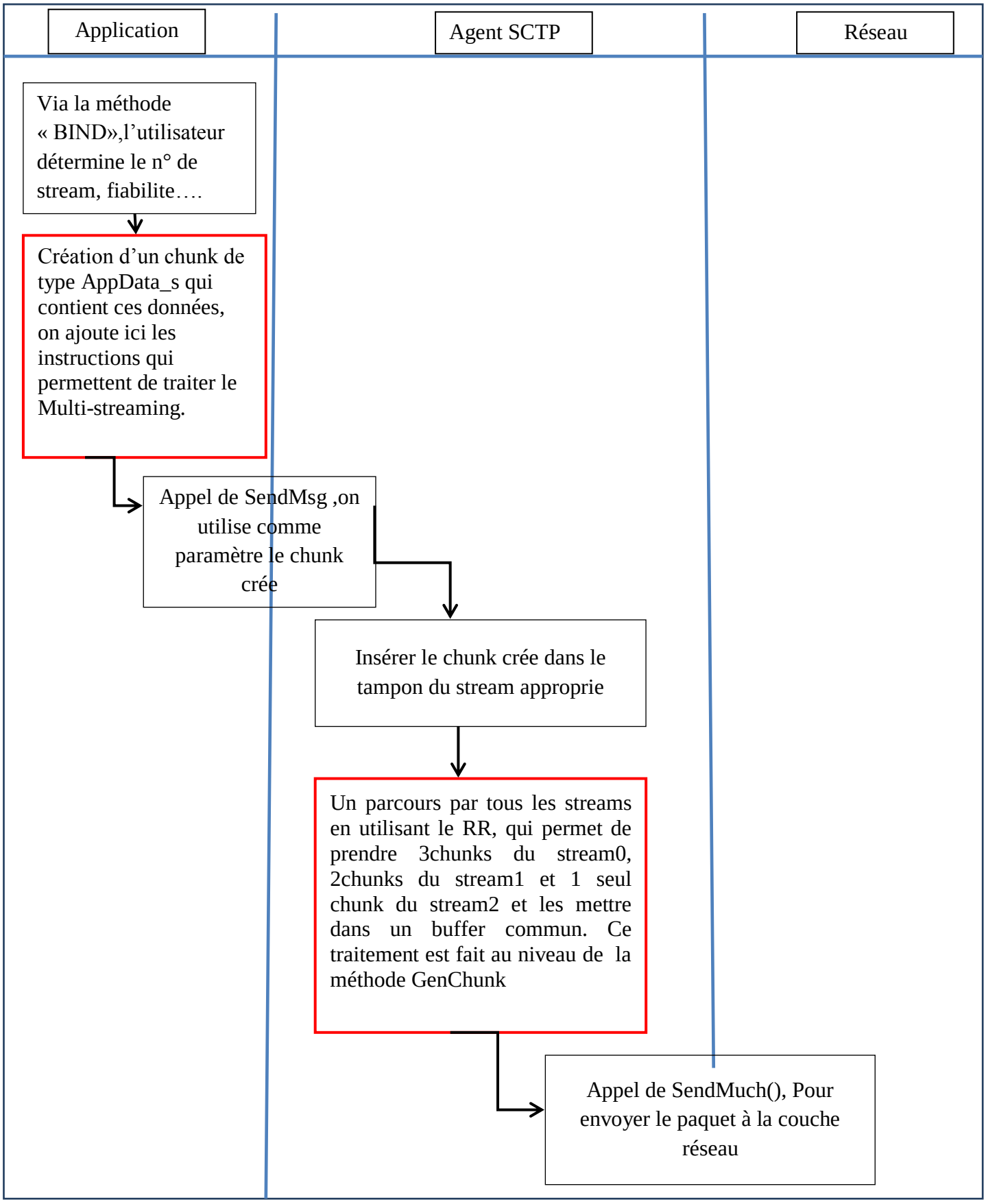


Figure 3.15:Diagramme de transmission d'un chunk de la couche application à la couche réseau.

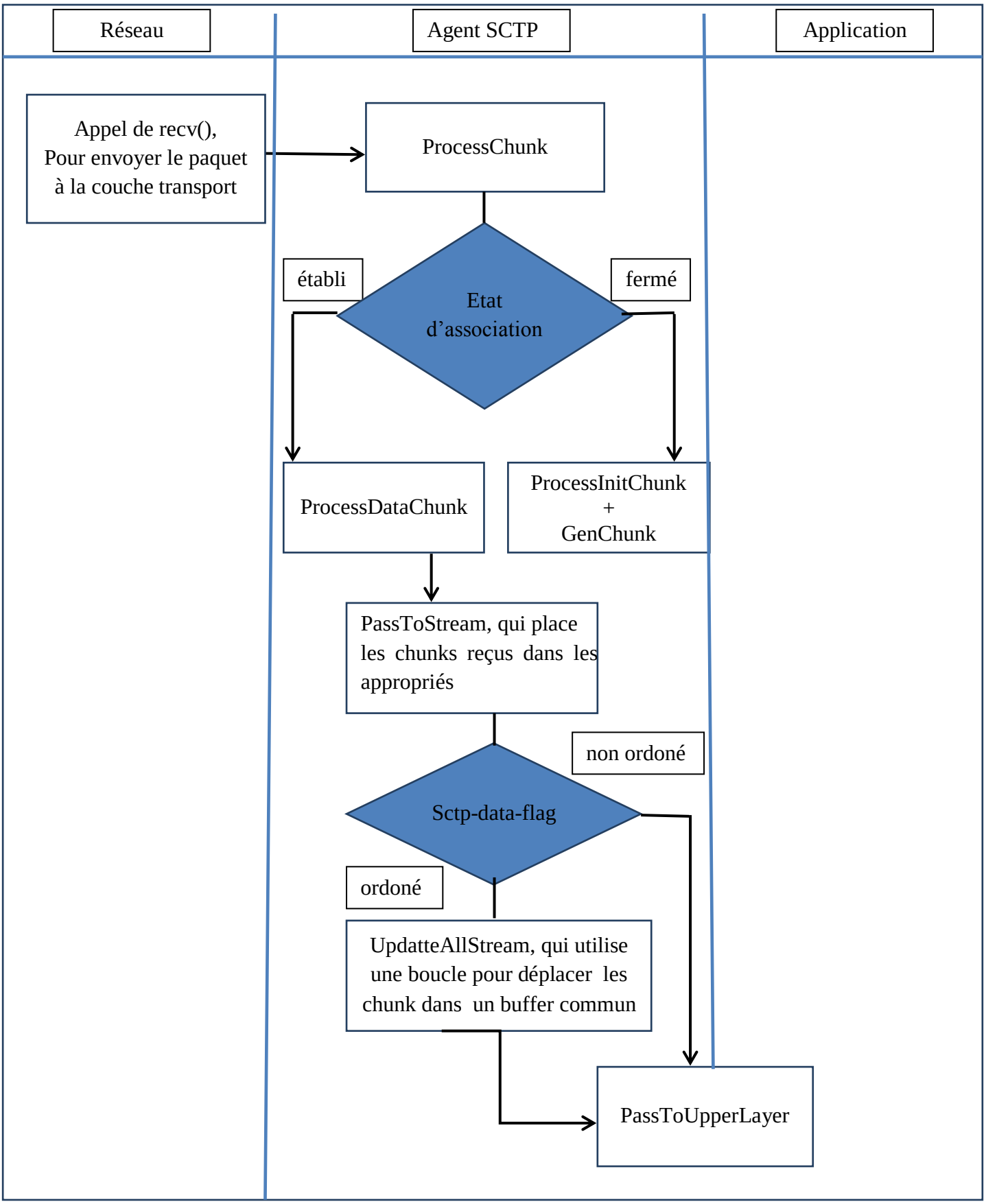


Figure 3.16:Diagramme de transmission d'un chunk de la couche réseau à la couche application.

7. Conclusion

Dans ce chapitre, nous avons proposé une amélioration des performances du protocole SCTP qui consiste de faire une extension au mécanisme de multi-streaming. Cette extension concerne le SAP entre application et SCTP et sur SCTP lui-même.

Dans le chapitre suivant nous allons passer à l'implémentation des algorithmes

« SCTP », et« SctpApp1 » que nous avons proposé, afin d'évaluer leurs performances et comparer les résultats, en utilisant le simulateur NS2.

Chapitre 4

Simulation et Analyse des Résultats

1. Introduction

Dans ce chapitre nous allons décrire l'implémentation sous NS2 du protocole SCTP. Nous présentons les modifications apportées au code de SCTP sous l'environnement de simulation utilisé et les scenarios de simulation.

2. Caractéristiques de l'environnement de simulation utilise

Dans notre travail nous avons utilisé une machine virtuelle sur laquelle nous avons installé le système linux Fedora14 (les étapes d'installation sont précisées dans l'Annexe C). Nos simulations ont été faites sous NS2.

- machine virtuelle Oracle VM Virtual Box.
- le système linux Fdora14.
- Le Simulateur NS version 2.35 et avec :
  - L'outil nam version 1.15.
  - L'outil xgraph version 12.2.

3. Implémentation

Pour mettre en œuvre notre conception de SCTP il faut premièrement ajouter nos modifications au code C++ de NS.

Il s'agit de la définition d'une application qui connait et qui est capable d'activer le multi-streaming ainsi que des modifications sur l'agent SCTP de NS pour implémenter notre extension à l'algorithme d'ordonnancement.

Nous soumettons les fichiers C++ à une topologie écrite en OTcl pour récupérer des résultats de simulation et les analyser.

3.1. Protocole SCTP sous NS2

Dans nos simulations nous utilisons l'agent SCTP développé sous NS2. Cet agent de base, supporte les caractéristiques suivantes :

– Établissement d'une association.

– Transmission des Data Chunks.

– Acquittement des Data Chunks reçus.

– Gestion du Timer de retransmission.

– Des points terminaux SCTP à multi-accès (multi-homed).

–gestion des SSN.

– Livraison en/hors séquence.

– Rapport des trous sur les TSNs des données reçues.

– SCTP slow start congestion Avoidance.

3.2. Traitement en multi-streaming des chunks

Pour traiter les données provenant d'applications dans « *sctp.cc* », un algorithme d'ordonnancement est responsable du transfert des données de la couche transport à la couche réseau. Cet algorithme traite les streams de façon différente selon que les données proviennent de FTP ou qu'ils soient des données d'application basée sur SCTP. Il est implémenté au niveau de la procédure GenChunk. Notons que cette procédure est responsable de la génération des chunks durant toutes les phases d'une association SCTP.

Lorsque l'association est établie, le passage entre stream est assuré au niveau de cette méthode comme l'indique le code suivant de sctp.cc

```
IntSctpAgent::GenChunk(SctpChunkType_EeType, u_char *ucpChunk)
{……….
AppData_S *spAppMessage = NULL;
intiSize = 0;
switch(eType)
   {
……….
case SCTP_CHUNK_DATA:
     if(eDataSource == DATA_SOURCE_APPLICATION)
          {
            ……….
          }
Else{
DBG_PL (GenChunk, "eDataSource=DATA_SOURCE_INFINITE") DBG_PR;
……
((SctpDataChunkHdr_S *) ucpChunk)->sHdr.usLength = iSize;
if( (uiDataChunkSize % 4) != 0)
          iSize += 4 - (uiDataChunkSize % 4);
            ((SctpDataChunkHdr_S *) ucpChunk)->uiTsn = ++uiNextTsn;

          ((SctpDataChunkHdr_S *) ucpChunk)->usStreamId
             = (usNextStreamId % uiNumOutStreams);
            if(eUnordered == TRUE)
              {
               ((SctpDataChunkHdr_S *) ucpChunk)->sHdr.ucFlags
                     |= SCTP_DATA_FLAG_UNORDERED;
                /* no stream seqnum on unordered chunks!
```

```
            */
          }
        else
          {
            ((SctpDataChunkHdr_S *) ucpChunk)->sHdr.ucFlags = 0;
            ((SctpDataChunkHdr_S *) ucpChunk)->usStreamSeqNum
                =
spOutStreams[usNextStreamId%uiNumOutStreams].usNextStreamSeqNum++;
          }
}
```

3.3. Application SCTP

Nous avons créer une *« sctp aware application »* dont l'implémentation application est identifiée la sous-classe « *Sctp_app1.cc* » de la classe « *Application* ». Nous avons utilisé des variables '*bindable*' à l'implémentation TCL qui existe dans le fichier d'entête '*Sctp_app1 .h'* pour identifier les paramètres d'une association SCTP.

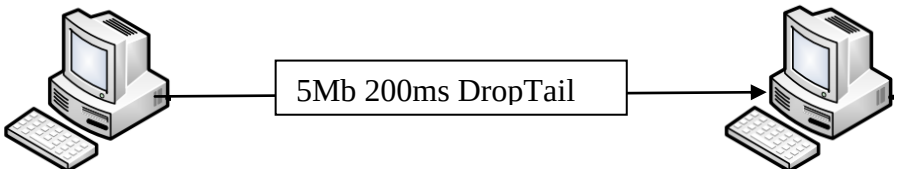
« *Sctp_app1.cc* » définit l'implémentation qui génère les messages à transmettre. Elle définit leur taille, la fiabilité choisie pour sa transmission, Il est implémenté au niveau de la procédure timeout mais cette implémentation ne distingue pas entre données. Nous étendons cette implémentation en spécifiant le numéro de stream de chaque chunk. La couche application transmet ces chunks aux agents SCTP via l'appel de la méthode « *sendmsg()* ».

```
timeout()
void SctpApp1::timeout()
{
      if (running_) {
                  data_to_transport = new AppData_S;
                  memset(data_to_transport, 0, sizeof(AppData_S));
                  data_to_transport->usNumStreams = numStreams;
                  data_to_transport->usNumUnreliable = numUnreliable;
                  data_to_transport->usReliability = reliability;
                  data_to_transport->uiNumBytes = 257;
                  agent_->sendmsg(agent_->size(), (char*)data_to_transport);
```

3.4. Script Tcl

La mise en œuvre de sctp.cc avec la notion de multi-streaming sous NS2 consiste à produire un scénario en TCL. Il s'agit de simuler une topologie de réseaux.

3.4.1. Modélisation du système



Emetteur Récepteur

Figure 4.15: Modèle de notre simulation.

Pour cela, nous avons créé deux nœuds NS2 (node), reliés par un lien full duplex (duplex-link) supportant un débit de 5Mb, un temps d'accès au medium de 200 ms, et qui utilise l'algorithme de file d'attente Drop tail pour la gestion des congestions.

Nous avons ensuite crée un agent SCTP (Agent/SCTP) attaché au nœud n0 (attach-agent) qui permet à n0 de transmettre des paquets SCTP sur le réseau. Puis, nous avons créé un envoi de paquets (Application/….) attaché à l'agent SCTP définit précédemment (attach-agent).

Nous avons enfin créé un agent vide, destiné à recevoir des paquets SCTP, (Agent/Null). On l'attache au nœud n1 (attach-agent) puis on connecte l'agent SCTP et l'agent vide (connect).

Enfin, on indique que les traces de simulation doivent être loguées, qu'on souhaite lancer NAM à la fin de la simulation, que celle-ci va durer 55 secondes et que les paquets ne sera émis qu'à partir de la $0.5^{ème}$ seconde de simulation, et jusqu'à la $20^{ème}$ seconde.

Notons que nous avons testé d'autres paramètres, mais ceux-là ne sont cités qu'à titre indicatif. Ci-après une partie d'un exemple élémentaire de script testé.

```
##################################creation des noued
set n0 [$ns node]
##########étiquter le noued 0 avec le nom émetteur et la coouleur rouge######
$n0 label "émetteur"
$n0 colorred
##########étiquter le noued 1 avec le nom recepteur et la couleur bleu ######
set n1 [$ns node]
$n1 label "recepteur"
$n1 color blue
################################Le lien entre ces deux nœuds est un duplex-link de 0.5 Mb de débit
################################et de retard égale à 200 ms
$ns duplex-link $n0 $n1 5Mb 200ms DropTail
$ns duplex-link-op $n0 $n1 orient right
```

3.5. Ordonnancement détaillé

3.5.1. Modifications apportées à « sctp.h et sctp.cc»

Notre implémentation se base sur le code sctp.cc et sctp_app1.cc de NS à la fois. Les modifications que nous apportons touchent aux fichiers .h et .cc et quelque procédures implémentées pour émission des chunks , dont la procédure « *sendmsg()*» est le déclencheur pour l'envoi des chunks à partir de la couche Application.

Dans nos modification nous avons ajouté des nouvelles variables pour permettre de parcourir les streams et sélectionner le nombre de chunks empilés à chaque stream.

Nous avons ajouté deux variables pour modifier l'algorithme de RR.

sctp.h

```
//*************** variables ajouter**********//
 int max;
 intcontMax
```

Parmi les méthodes du Sctp.cc nos modifications affectent la fonction Genchunk.

Selon le code suivant :

```
intSctpAgent::GenChunk(SctpChunkType_EeType, u_char *ucpChunk)
{
……
AppData_S *spAppMessage = NULL;
intiSize = 0;
```

```
switch(eType)
   {
……….……
case SCTP_CHUNK_DATA:
     if(eDataSource == DATA_SOURCE_APPLICATION)
          {
            ……..……
          }
Else
{
DBG_PL (GenChunk, "eDataSource=DATA_SOURCE_INFINITE") DBG_PR;
……
((SctpDataChunkHdr_S *) ucpChunk)->sHdr.usLength = iSize;
////////////////////////////////////determine le ndr de chunks dans chaque stream//////////////////////
switch(usNextStreamId % uiNumOutStreams)
{
case 0:
contMax=3;
fprintf(stderr, "%d t0 \n", usNextStreamId);
break;
case 1:
contMax=2;
fprintf(stderr, "%d t1 \n", usNextStreamId);
break;
case 2:
contMax=1;
fprintf(stderr, "%d t2 \n", usNextStreamId);
break;
}
//////////////////////////tester l'égaliter de max et contMax ,incrementernextStreamId
//////////////////////////
//////////////////////////et rénitialiser max à 0 pour passe à nextstream//////////////////////////////////////
if(max==contMax)
{
usNextStreamId++;
fprintf(stderr, "%d t \n", usNextStreamId);
max=0;
}
//////////////////////////incrémenter le max ////////////////////////////////////
max++;

if( (uiDataChunkSize % 4) != 0)
          iSize += 4 - (uiDataChunkSize % 4);
```

```
        ((SctpDataChunkHdr_S *) ucpChunk)->uiTsn = ++uiNextTsn;

       ((SctpDataChunkHdr_S *) ucpChunk)->usStreamId
          = (usNextStreamId % uiNumOutStreams);

        if(eUnordered == TRUE)
         {
          ((SctpDataChunkHdr_S *) ucpChunk)->sHdr.ucFlags
               |= SCTP_DATA_FLAG_UNORDERED;

           /* no stream seqnum on unordered chunks!
            */
         }

        else
         {
           ((SctpDataChunkHdr_S *) ucpChunk)->sHdr.ucFlags = 0;
           ((SctpDataChunkHdr_S *) ucpChunk)->usStreamSeqNum
                =
spOutStreams[usNextStreamId%uiNumOutStreams].usNextStreamSeqNum++;
         }
                                                  }
```

3.5.2. Modifications apportées à « sctp_app1.h et sctp_app1.cc»

Sctpapp1.h contient des déclarations des fonction et procédures utiliser par sctp_app1.cc comme les procédures *timeout(), start(), stop()* et la fonction *next()* et contient aussi la déclarations des variables double *interval_ , intnumUnreliable, intnumStreams, intreliability*.

Dans nos modification ne avons pas ajouté des nouveaux fonctions ou procédures, mais nous avons définie intégrés des nouvelles variables dans sctp_app1.cc.

Parmi les fonctions du sctp_app1.cc nos améliorations affecte le procédure *timeout()*.

1) premièrement nous avons ajouté des variables pour fait une notion de multi-streaming.

sctp_app1.h

```
//*************** variables ajouter**********//

int nextStreamId;
```

Parmi les fonctions du sctp_app1.cc nos améliorations affecte le procédure *timeout()*.

Le code que nous avons ajouté est comme suit :

sctp_app1.cc

```
 void SctpApp1::timeout()
{
      if (running_) {
                  data_to_transport = new AppData_S;
                  memset(data_to_transport, 0, sizeof(AppData_S));
                  data_to_transport->usNumStreams = numStreams;
                  data_to_transport->usNumUnreliable = numUnreliable;
                  data_to_transport->usReliability = reliability;
                  data_to_transport->uiNumBytes = 1450;
data_to_transport->usStreamId = (nextStreamId % numStreams);
nextStreamId++;
```

2) Deuxièmement nous envisageons de maitriser les types de données à transmettre par stream.

4. Étude des performances des extensions

Dans la suite on va faire la Simulation et l'analyse des performances, et ça nécessite la connaissance des différents champs du .tr générer par NS2 pour le protocole SCTP, Pour cette raison on va présenter d'avance un exemple d'un fichier trace .tr de SCTP avec la signification de ces différents champ

Les fichiers trace avec l'extension .nam constituent un premier type de fichier de trace. Ces fichiers contiennent des informations utiles pour la visualisation des nœuds et leurs déplacements ainsi que le parcours des paquets entre les nœuds .Leur usage requiert le logiciel **NAM** inclus dans NS2 et ce en exécutant la commande nam suivie du fichier à visualiser.

Les fichiers traces avec l'extension **.tr** constituent la partie utile à l'étude des performances du réseau.

Voilà un exemple de fichier trace .tr :

```
- 1.300294 0 1 sctp 56 -------H 1 0.0 1.0 1 -1 6 65535 65535
r 1.500384 0 1 sctp 56 -------H 1 0.0 1.0 1 -1 6 65535 65535
+ 1.500384 1 0 sctp 56 -------B 2 1.0 0.0 1 -1 7 65535 65535
- 1.500384 1 0 sctp 56 -------B 2 1.0 0.0 1 -1 7 65535 65535
r 1.700474 1 0 sctp 56 -------B 2 1.0 0.0 1 -1 7 65535 65535
+ 1.700474 0 1 sctp 1500 -------D 1 0.0 1.0 1 1 8 0 0
- 1.700474 0 1 sctp 1500 -------D 1 0.0 1.0 1 1 8 0 0
+ 1.700474 0 1 sctp 1500 -------D 1 0.0 1.0 1 2 9 1 0
+ 1.700474 0 1 sctp 1500 -------D 1 0.0 1.0 1 3 10 2 0
- 1.702874 0 1 sctp 1500 -------D 1 0.0 1.0 1 2 9 1 0
- 1.705274 0 1 sctp 1500 -------D 1 0.0 1.0 1 3 10 2 0
r 1.902874 0 1 sctp 1500 -------D 1 0.0 1.0 1 1 8 0 0
+ 1.902874 1 0 sctp 48 -------S 2 1.0 0.0 1 1 11 65535 65535
- 1.902874 1 0 sctp 48 -------S 2 1.0 0.0 1 1 11 65535 65535
r 1.905274 0 1 sctp 1500 -------D 1 0.0 1.0 1 2 9 1 0
+ 1.905274 1 0 sctp 48 -------S 2 1.0 0.0 1 2 12 65535 65535
- 1.905274 1 0 sctp 48 -------S 2 1.0 0.0 1 2 12 65535 65535
r 1.907674 0 1 sctp 1500 -------D 1 0.0 1.0 1 3 10 2 0
+ 1.907674 1 0 sctp 48 -------S 2 1.0 0.0 1 3 13 65535 65535
- 1.907674 1 0 sctp 48 -------S 2 1.0 0.0 1 3 13 65535 65535
r 2.10295 1 0 sctp 48 -------S 2 1.0 0.0 1 1 11 65535 65535
+ 2.10295 0 1 sctp 1500 -------D 1 0.0 1.0 1 4 14 0 1
- 2.10295 0 1 sctp 1500 -------D 1 0.0 1.0 1 4 14 0 1
```

Figure 4.2: Format d'un fichier trace avec l'extension .tr.

Lors du traçage SCTP, les paquets de type SCTP sont pertinentes. Leur type de paquet, le type de chunk, la taille des paquets, TSN (CumAck points pour les blocs SACK), flux ID, SSN, et à l'arrivée / départ / temps de perte sont données par les positions sur le champ 5, 7 (position de drapeau 8), 6, 12, 14 , 15 et 2, respectivement.

Si un paquet a plus d'un chunk, une ligne est imprimée pour chaque chunk. Quand un chunk de contrôle ne pas utiliser TSNs, ID de flux, ou SSNs, les lignes de trace de ces chunk utilisent des numéros non définis (-1 ou 65535) pour ces champs. Le + indique une arrivée des paquets, d une drop, et - un départ.

La position de drapeau 8 de champ 7 indiquer le type de chunk comme suit. I drapeau indique un chunk association de contrôle d'initiation (INIT, INIT-ACK, COOKIE-ECHO, COOKIE-ACK). Une prochaine version devrait utiliser I pour la INIT et chunk INIT-ACK, et C pour le COOKIE-ECHO et des chunk COOKIE-ACK. Les drapeaux D, S, H, et B indiquent un bloc DATA chunk, un SACK, un HEARTBEAT chunk, et un HEARTBEAT-ACK chunk. Un certain nombre de scripts traite ce fichier pour produire une sortie graphique ou des résumés statistiques (par exemple, voir la procédure d'arrivée en ~ ns / tcl / test / test-suite-sctp.tcl).[48]

Nous résumons, dans le tableau ci- dessous, la signification des différents champs d'un fichier trace.

Le champ	La signification du champ
1(+, -, r, d, h)	+indique une entrée en file d'attente. - indique une sortie de file. r une réception de paquet. **d** que le paquet est jeté. **h** qu'il s'agit d'un hop.
2	L'instant auquel l'évènement se produit .
3	Le nœud par lequel le paquet entre en file .
4	Le nœud qui se trouve à l'autre extrémité du lien pour lequel on entre ou on sort de file d'attente .
5	Le type de trafic.
6	La taille du paquet de données, en octets.
7	Les flags appliqués au paquet (I pour la INIT et chunk INIT-ACK, et C pour le COOKIE-ECHO et des chunk COOKIE-ACK. Les drapeaux D, E, H, et B indiquent un bloc DATA, un sac, un chunk de HEARTBEAT, et un chunk HEARTBEAT-ACK) .
8	L'identifiant du flux ou la class définie manuellement.
9	Le numéro du nœud source du paquet, un point, puis le numéro de l'agent émetteur.
10	Le numéro du nœud destinataire du paquet, un point, puis le numéro de l'agent récepteur.
11	Le numéro de séquence du paquet, relatif au flux auquel il appartient
12	TSN
14	l'identifiant de flux
15	SSN.

Tableau 4.1 : Signification des différents champs d'un fichier trace.

4.1. Simulation du protocole originel avec le type de données FTP

Pour mettre en œuvre le protocole SCTP sous NS2 il consiste à produire un scénario en TCL qu'il simule une topologie de réseaux.

Le code TCL de la simulation, en détaille est fournit sur l'Annexe (D).

- **Fichier de sortie NAM**

La visualisation du réseau est effectuée par l'outil NAM à partir du fichier de sortie (.nam) généré par NS2.

Voici le résultat de visualisation de réseau par l'outil NAM :

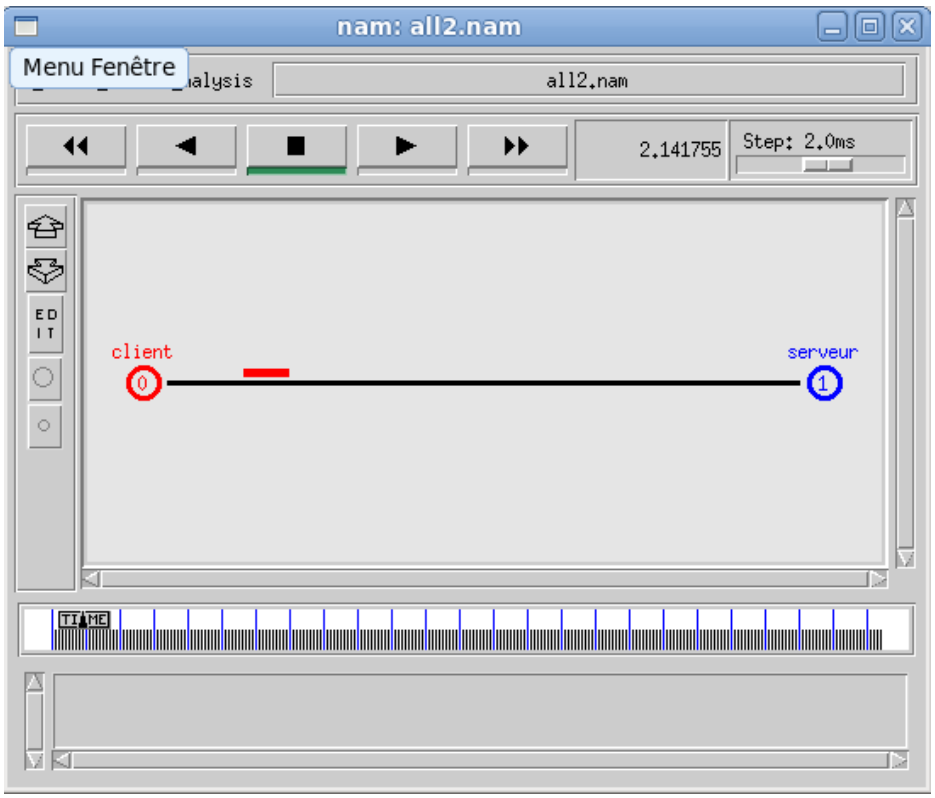


Figure 4.3: Visualisation du transfert des données FTP.

Pour extraire l'information sur le nombre de stream utiliser par l'application à partir du ficher de trace (.tr), on utilise des programmes écrits en langage AWK, et on utilise la commande AWK (présentée dans le chapitre 2) qui permet de sauvegarder le résultat dans un fichier.

Voici le code AWK(1) utilisé pour extraire les informations concernant les données transporté par chaque stream:

```
BEGIN {
streamId = 0;
  flag;
```

```
            }
      {
streamId = $14;
            time = $2;
            flag = $7;
evenement = $1;
TSN=$12;
if ($7=="-------D" && $1=="+" )
 {
printf("%d %s %d \n" ,TSN,"    ",streamId ) > "sctpapp1";
}
  }
END {
print("Done");
   }
```

Voilà le fichier qui contient les informations extraire à partir de fichier .tr :

TSN	streamId
1	0
2	1
3	2
4	0
5	1
6	2
7	0
8	1
9	2

En utilisant les résultats d'AWK appliqué et l'outil xgraph, nous avons obtenus les graphes suivants :

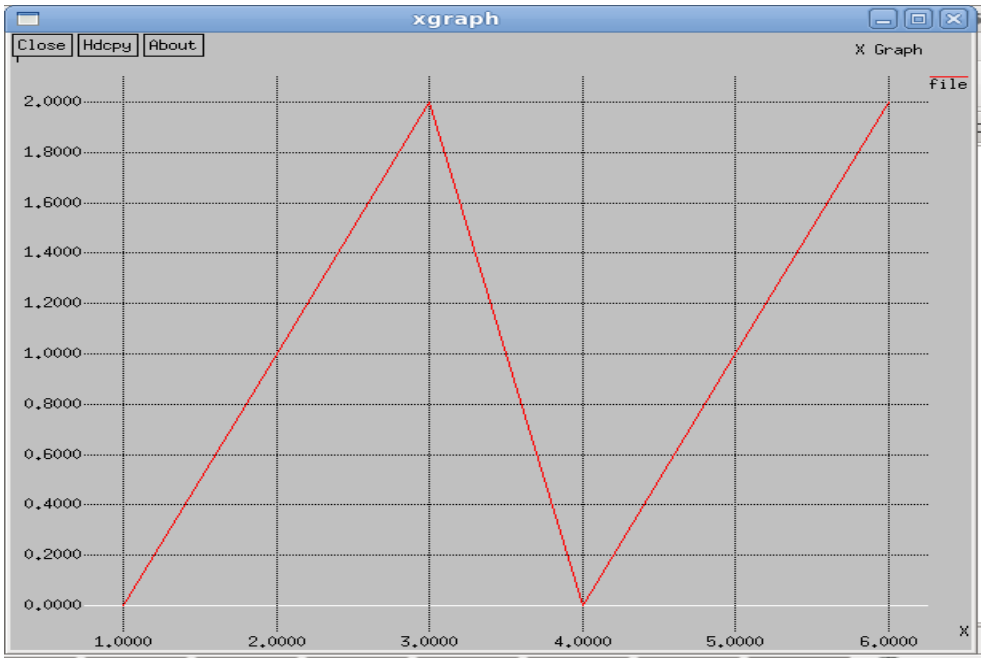


Figure 16.4: FTP originel avec 3 streams.

Voici le code AWK(2) utilisé pour calculer le pourcentage de la quantité de données dans chaque stream :

```
BEGIN {
streamId = 0;

            flag;
     }
     {
streamId = $14;
              time = $2;
              flag = $7;
taille = $6
 if ($7=="-------D" && $1=="+" )
    {
     if ($14== "0" )
        {

bytes_counter+=taille ;

        }

 if ($14== "1" )
        {

        bytes_counter1+=taille ;

        }
     if ($14== "2" )
```

```
        {
  bytes_counter2+=taille ;
  }  }
}
    s= bytes_counter+bytes_counter1+bytes_counter2;

END {
printf(" %s %s %d %s %d %s\n","SID 0","     ",bytes_counter,"
",100*bytes_counter/s,"%") > "quantold";
printf(" %s %s %d %s %d %s\n","SID 1","     ",bytes_counter1,"
",100*bytes_counter1/s,"%") > "quantold";
printf(" %s %s %d %s %d %s\n","SID 2","     ",bytes_counter2,"
",100*bytes_counter2/s,"%") > "quantold";
print("l'opération fait avec succès");
}
```

Voilà le pourcentage de la quantité de données les streams :

SID 0	882000	33 %
SID 1	882000	33 %
SID 2	882000	33 %

En utilisant les résultats d'AWK appliqué et l'Excel, nous avons obtenus les graphes suivants :

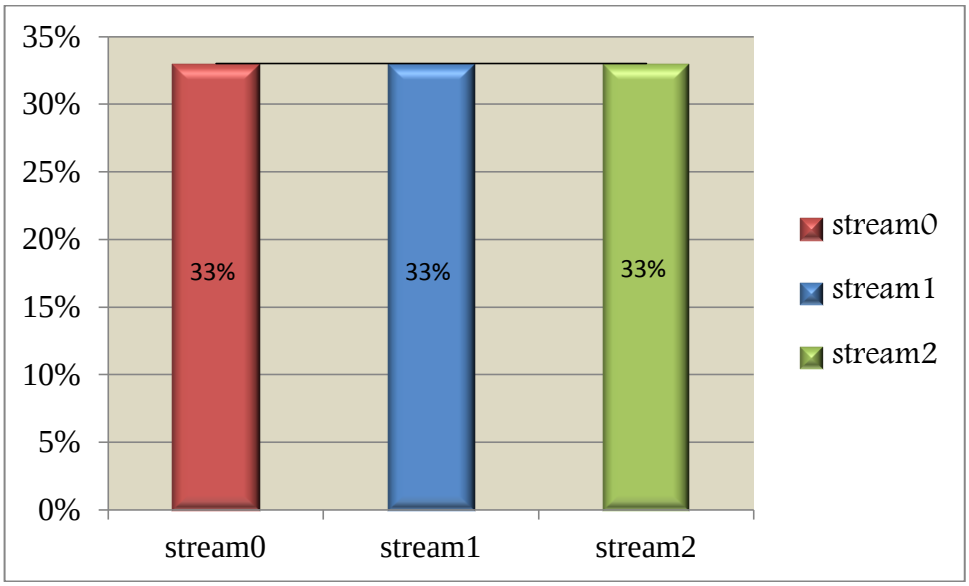


Figure 4.5: Pourcentage des données dans chaque stream avec le type FTP originel.

4.2. Simulation du protocole originel avec le type de données SctpApp1

Même pour le type SctpApp1 pour le mettre en œuvre sous NS2 il consiste à produire un scénario en TCL qu'il simule une topologie de réseaux.

Le code TCL de la simulation, en détaille est fournit sur l'Annexe (D).

- **Fichier de sortie NAM**

Voici le résultat de visualisation de réseau par l'outil NAM :

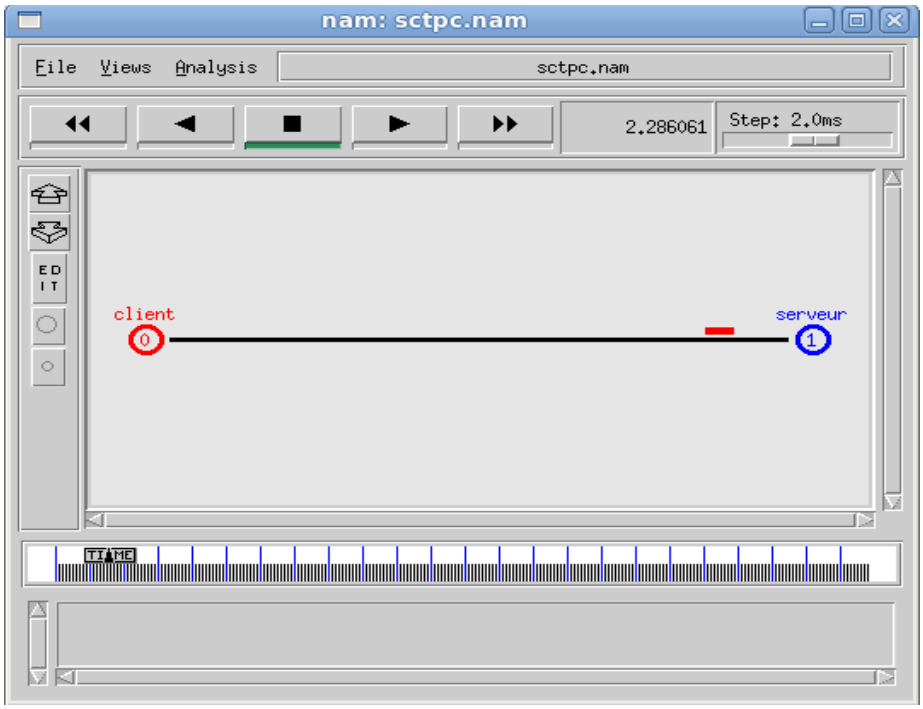


Figure 4.6: Visualisation du transfert des données SctpApp1.

Avec l'utilisation de même programme AWK(1) pour le fichier de sortie .tr de SctpApp1 on obtient les résultats suivant qui signifie qu'il est utilisé un seule stream :

TSN	streamId
1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	0
9	0

En utilisant les résultats d'AWK appliqué et l'outil xgraph, nous avons obtenus les graphes suivants :

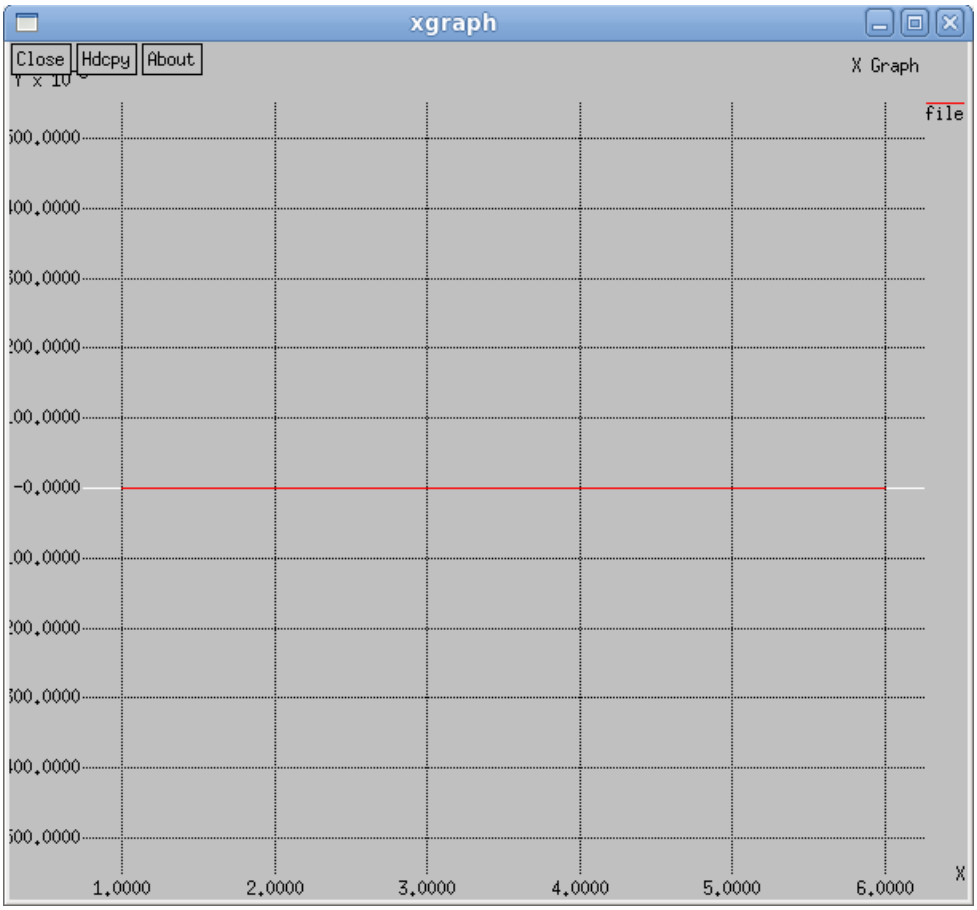


Figure 4.7 : SctpApp1 originel sans le multi-streaming.

Voilà aussi le pourcentage de la quantité de données les streams en utilisant le même programme AWK(2) :

SID 0	9396	100 %
SID 1	0	0 %
SID 2	0	0 %

En utilisant les résultats d'AWK et l'Excel, nous avons obtenus les graphes suivants :

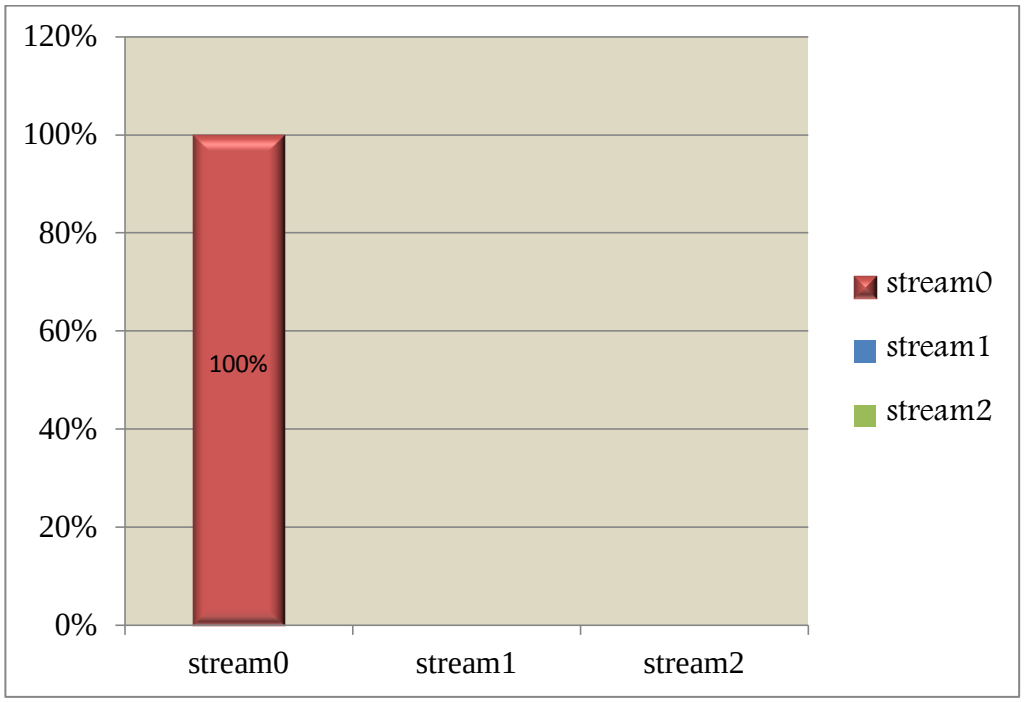


Figure 4.8 : Pourcentage des données dans chaque streams avec le type SctpApp1 originel.

4.3. Simulation du nouveau protocole

4.3.1. Simulation du nouveau protocole avec le type de données FTP

Voilà le fichier qui contient les informations extraire à partir de fichier .tr en utilisant notre modifications :

TSN	streamId
1	0
2	0
3	0
4	1
5	1
6	2
7	0
8	0
9	0

En utilisant les résultats d'AWK appliqué et l'outil xgraph, nous avons obtenus les graphes suivants :

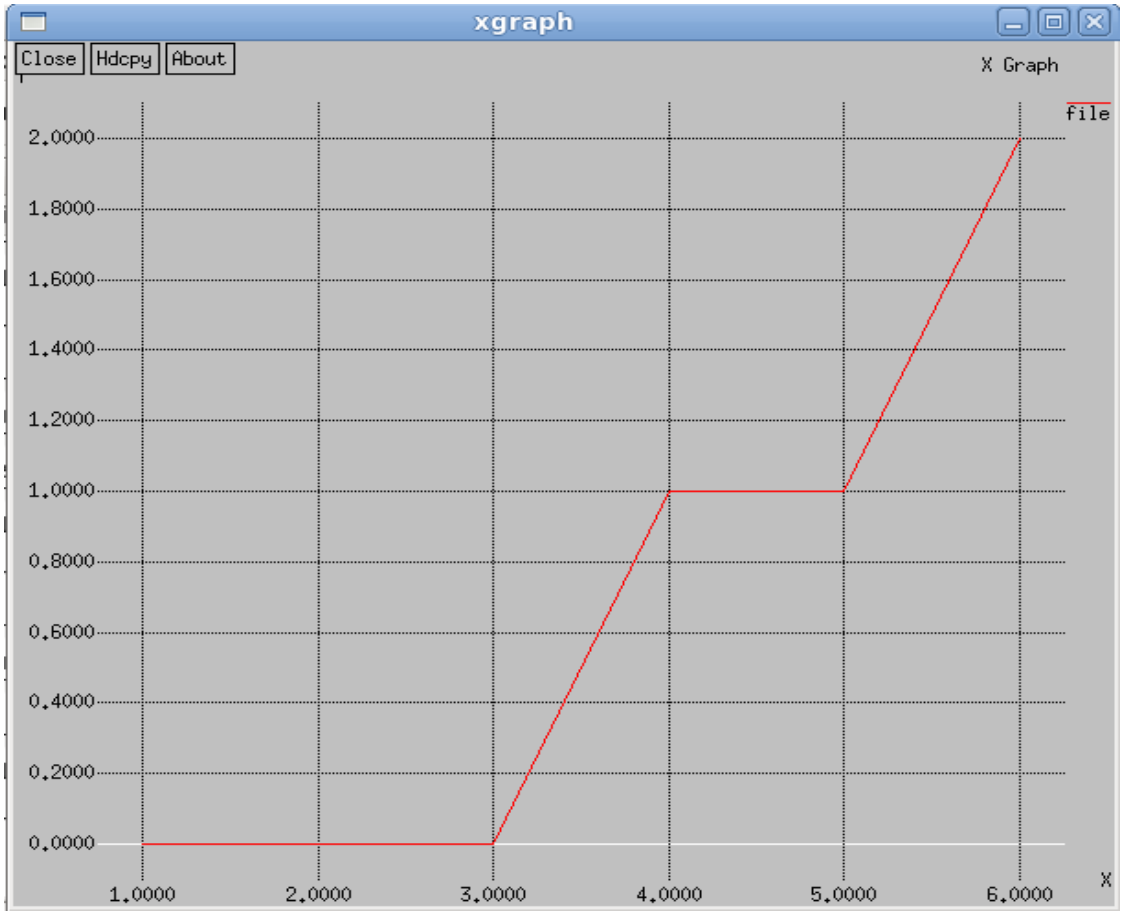


Figure 4.9 : Nouveau FTP avec 3 stream.

Résultat d'Awk(2) qui calcule la quantité des donnes et leurs % dans chaque stream :

SID 0	2164500	50 %
SID 1	1440000	33 %
SID 2	720000	16 %

En utilisant les résultats d'AWK et l'Excel, nous avons obtenus les graphes suivants :

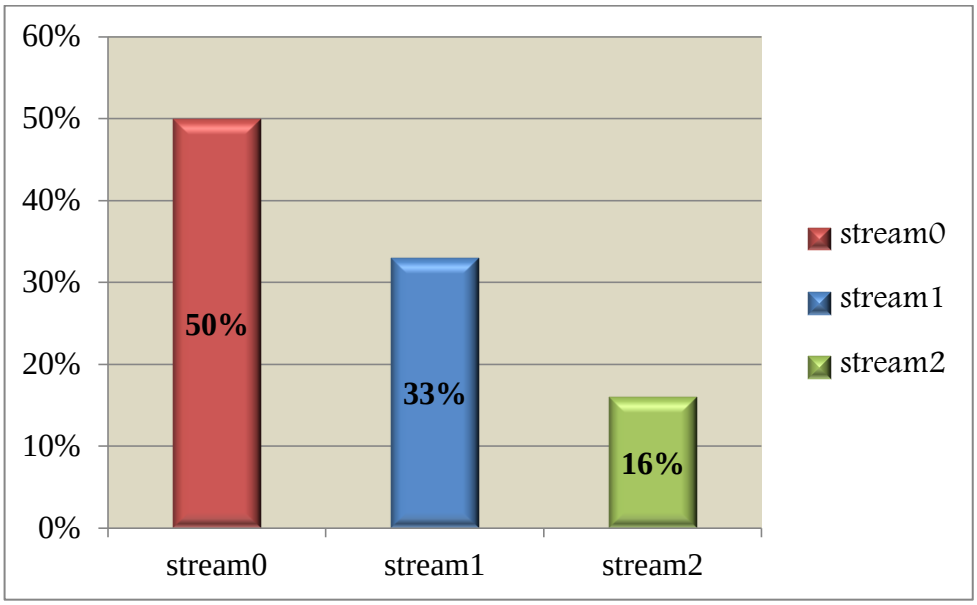


Figure 4.10:Pourcentage des données dans chaque stream avec le type FTP après la modification.

4.3.2. Simulation du nouveau protocole avec le type de données SctpApp1

Comme nous avons vu précédemment, nous avons fait deux modification dans Sctp_app1.cc, la première pour faire une application qui connaitre le multi-streaming et la deuxième pour définit une priorité entre le stream dans la quantité des données transporter.

- ***Résultats de la première modification :***

Le fichier qui contient les informations extraire à partir de fichier .tr en utilisant notre 1ère modifications proposé :

TSN	streamId
1	0
2	1
3	2
4	0
5	1
6	2
7	0
8	1
9	2

On obtient le même graphe originel de FTP (Figure 4.4)

Voilà les résultats d'Awk (2) qui calcule la quantité des donnes et leurs % dans chaque stream :

SID 0	30000	33 %
SID 1	30000	33 %
SID 2	30000	33 %

On obtient le même graphe de FTP (Figure 4.5)

- ***Résultats de la deuxième modification :***

Le fichier qui contient les informations extraire à partir de fichier .tr en utilisant notre 2ème modifications proposé :

TSN	streamId
1	0
2	0
3	0
4	1
5	1
6	2
7	0
8	0
9	0

On obtient le même graphe de FTP(Figure4.9)

Résultat d'Awk (2) qui calcule la quantité des donnes et leurs % dans chaque stream

SID 0	91500	50 %
SID 1	60000	33 %
SID 2	30000	16 %

Nous avons obtenus le même graphe de FTP(Figure4.10)

4.4. Comparaison entre l'ancien et le nouveau protocole

Le tableau ci-dessous présente le niveau de données dans chaque stream à la fin du temps de simulation pour le nouveau protocole et l'origine protocole dans les deux cas (où les données aware application ou non).

	Stream	sctp	nouvsctp
FTP	S0	33%	50%
	S1	33%	33%
	S2	33%	16%
SctpApp1	S0	100%	50%
	S1	0%	33%
	S2	0%	16%

Tableau 4.2 : Tableau comparatif entre les deux protocoles.

A partir des données illustrée dans le tableau, nous permet de remarquer que les quantités des données transporter sur les streams sont totalement différentes dans les protocoles SCTP et nouvSCTP, dont le protocole originel traiter les streams de la même manière par contre dans nos protocole proposer en donne des priorités au streams d'une façon que le stream 0 est plus prioritaire au stream 1, et le stream plus prioritaire au stream 2.

5. Conclusion

Basés sur les résultats de la simulation, nous avons montré que notre protocole proposé, nouvSCTP qui consiste à faire une extension de multi-streaming du protocole SCTP, définit un nouveau algorithme d'ordonnancement qui permet de faire une priorité entre les streams dans la quantité des données transférer et faire une application qui connais le mécanisme de multi-streaming implémenter par l'agent SCTP comparé au protocole « SCTP ».

Conclusion & perspectives

Le protocole SCTP, depuis son premier RFC en 2000 constitue l'un des sujets de recherche innovants pour diverses disciplines des sciences et techniques de l'information et de la communication mais avec toutefois des contraintes spécifiques qui permettent de relever certains défis.

Parmi les problèmes posés l'utilisation du mécanisme de multi-streaming de ce protocole qui traite tous les streams de manière identique et que ce mécanisme n'est pas connu par certains types de données. Notre travail propose une modification au protocole de transport SCTP en vu de faire un nouvel algorithme d'ordonnancement des transmissions des données et une application qui peut en tirer profit du multi-streaming.

Nous avons présenté notre travail en suivant un ordre d'idée qui commence par la présentation des caractéristiques des protocoles transport TCP et SCTP, ensuite nous avons examiné l'outil de simulation utilisé NS2. La conception, la réalisation et les résultats de notre contribution viennent en fin de ce rapport.

L'analyse des résultats obtenus des simulations de nouvSCTP proposé et SCTP originel, montre qu'un traitement des chunks sur les streams, adapté à la priorité entre ces streams, permet de modifier la quantité des données transporté par chaque stream.

Notre travail constitue une première brique dans l'implémentation de la qualité de service des transmissions sous SCTP. Les travaux ultérieurs peuvent se pencher encore mieux sur les paramètres considérés par la communauté scientifiques pour intégrer la QoS au code de NS2. Le multi-homing est également un mécanisme intéressant qui couplé aux spécifications du multi-streaming pour la QOS, constituent une bonne perspective de travail, notamment pour les réseaux mobiles.

ANNEXE A

1. Installations des packages :

Nous allons Installer ubuntu 14.04 sur notre machine de développement et avant sa nous allons d'abord installer des packages à l'aide de ces commandes :

$ sudo apt-get update

```
amel@ubuntu:~/Desktop$ cd ..
amel@ubuntu:~$ sudo apt-get update
```

$ sudo apt-get install gcc g++ python python-dev mercurial bzr gdb valgrind gsl-bin libgsl0-dev libgsl0ldbl flex bison tcpdump sqlite sqlite3 libsqlite3-dev libxml2 libxml2-dev libgtk2.0-0 libgtk2.0-dev uncrustify doxygen graphviz imagemagick python-pygraphviz python-kiwi python-pygoocanvas libgoocanvas-dev python-pygccxml

```
amel@ubuntu:~$ sudo apt-get install gcc g++ python python-dev mercurial bzr gdb
valgrind gsl-bin libgsl0-dev libgsl0ldbl flex bison tcpdump sqlite sqlite3 libsq
lite3-dev libxml2 libxml2-dev libgtk2.0-0 libgtk2.0-dev uncrustify doxygen graph
viz imagemagick python-pygraphviz python-kiwi python-pygoocanvas libgoocanvas-de
v python-pygccxml
Reading package lists... Done
```

2. Téléchargement et installations du NS

Télécharger **NS3** puis extraire ce dernier.

écrire les commandes suivant :

$ cd ns-allinone-3.21

$ chmod +x build.py

$ ./build.py --enable-examples --enable-tests

```
amel@ubuntu:~/Desktop$ cd ns-allinone-3.21
amel@ubuntu:~/Desktop/ns-allinone-3.21$ chmod +x build.py
amel@ubuntu:~/Desktop/ns-allinone-3.21$ ./build.py --enable-examples --enable-te
sts
```

Il faudra du temps pour exécuter cette commande. Sois patient. Si tout est ok, vous obtenez ce message : “Build finished successfully”.

```
or-debug
Waf: Leaving directory `/home/amel/Desktop/ns-allinone-3.21/ns-3.21/build'
'build' finished successfully (52m31.769s)
```

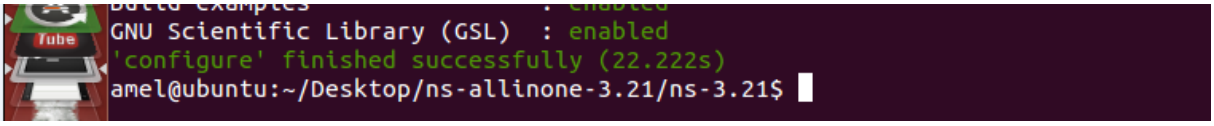
Après nous avons écrire les commandes

$ cd ns-3.21

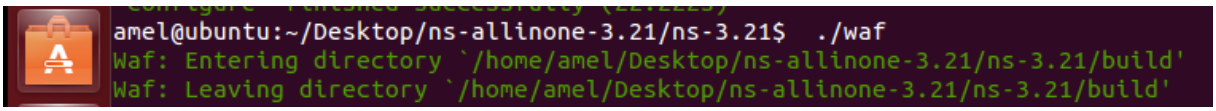
$ ./waf -d debug --enable-examples --enable-tests configure

```
amel@ubuntu:~/Desktop/ns-allinone-3.21/ns-3.21$ ./waf -d debug --enable-examples
--enable-tests configure
Setting top to                           : /home/amel/Desktop/ns-allinone-3.21/ns
```

Si tout est ok, vous obtenez ce message “configure finished successfully”.



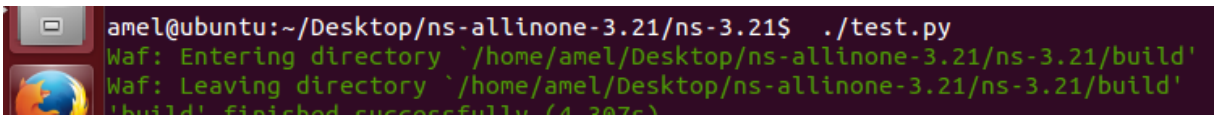
$ ./waf



1. Vérification de la sucées de l'installation

Pour vérifier que tout est installé avec succès ou non, il faut exécuter cette commande :

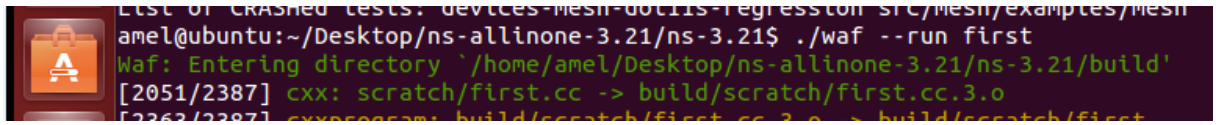
$ ./test.py



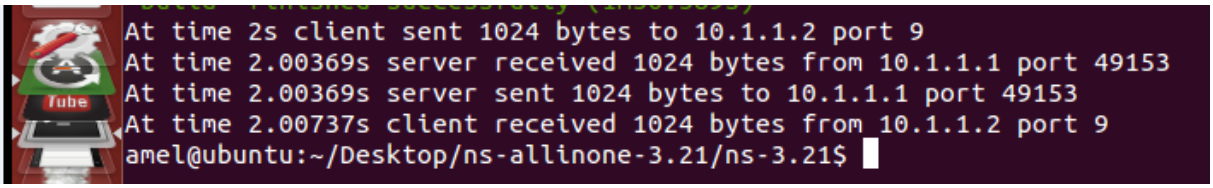
Maintenant, nous essayons d'exécuter l'un des programmes de dossier - ns-allinone-3.21 / ns-3.21 / exemple.

Nous copions first.cc du dossier exemple, puis le mettre sur ns-allinone-3.21 / ns-3.21 dossier / scratch, puis nous avons écrire la commande :

$ ./waf –run first



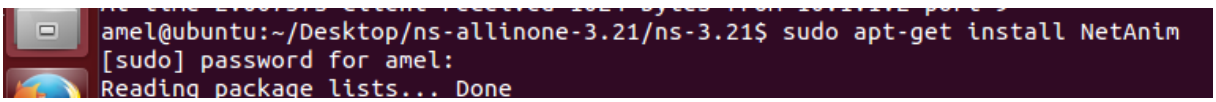
Il faut sortie ces résultats :



Parfois, nous devons montrer un graphe. Dans ce cas, nous allons compiler et extraire un fichier xml et l'analyse que nous en utilisant NetAnim.

Nous allons donc installer NetAnim par la commande :

$ sudo apt-get install NetAnim



ANNEXE B

Dans notre travail nous avons utilisé une machine virtuelle sur laquelle nous avons installé le système linux ubuntu. Nos simulations ont été faites sous NS2.

- machine virtuelle VMware Workstion 9 .
- le système linux ubuntu14.04.LTS

Télécharger le NS2.35

Décompresser avec la commande suivant :

```
amel@ubuntu:~$ cd ~/
amel@ubuntu:~$ tar -xvzf ns-allinone-2.35.tar.gz
```

utiliser la version GCC-4.4. dont l'installation se fait par la commande :

```
amel@ubuntu:~$  sudo apt-get install gcc-4.4
[sudo] password for amel:
```

dans le NS2.3 on ouvre le linkstate puis le ls.h est on modifie le mot "erase" par "this->erase" dans la ligne 137.

```
amel@ubuntu:~/Desktop/ns-allinone-2.35/ns-2.35/linkstate$ gedit ls.h
```

```
void eraseAll() { this->erase(baseMap::begin(), baseMap::end()); }
T* findPtr(Key key) {
        iterator it = baseMap::find(key);
        return (it == baseMap::end()) ? (T *)NULL : &((*it).second);
}
```

Dans cette étape on va dire au NS2 quelle est la version GCC qu'il va utiliser

```
amel@ubuntu:~/Desktop/ns-allinone-2.35/otcl-1.14$ gedit  Makefile.in
amel@ubuntu:~/Desktop/ns-allinone-2.35/otcl-1.14$
```

Dans le code on Change CC= @CC@ par CC=gcc-4.4.

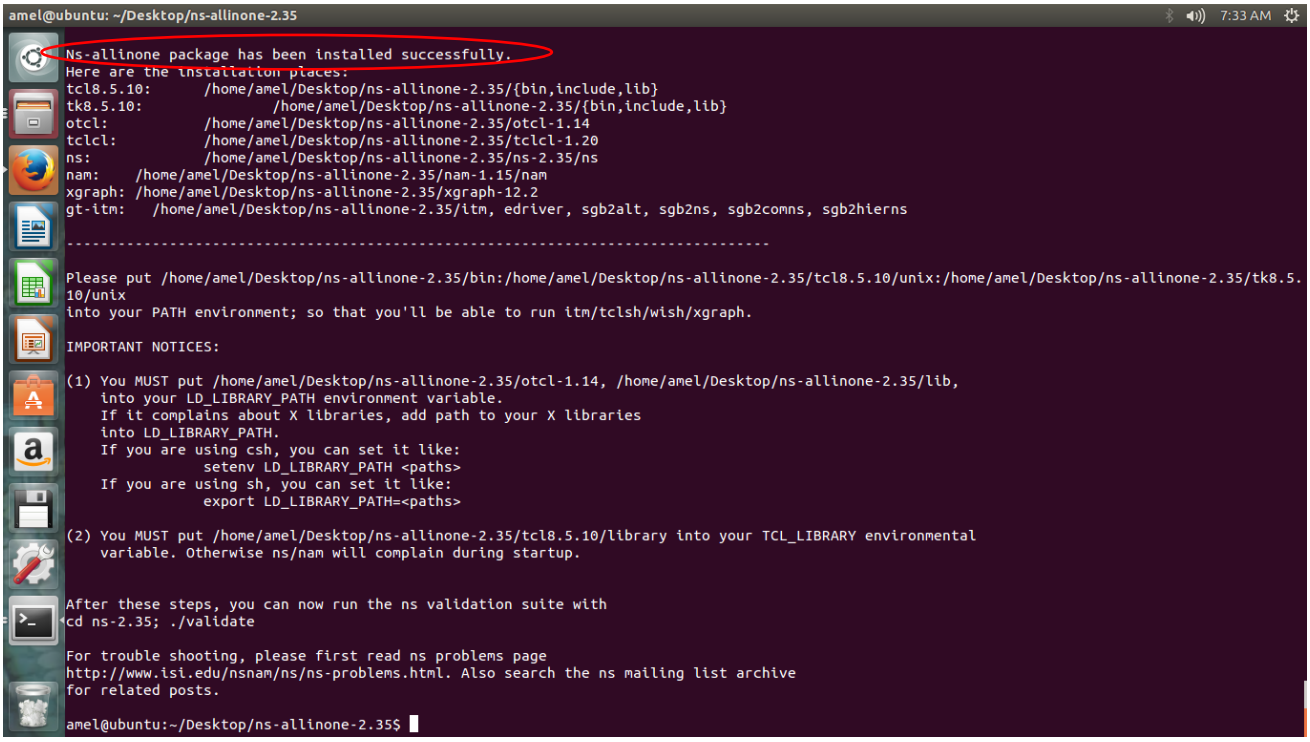
```
CC=             gcc-4.4
CFLAGS=         @CFLAGS@
RANLIB=         @RANLIB@
```

Le système étant prêt pour l'installation du NS, on doit premièrement entrer dans le root par la commande suivant :

```
amel@ubuntu:~$ sudo su
root@ubuntu:/home/amel#
```

Installer avec la commande : ./install

Si l'installation est termine avec succès on va voir la figure suivante qui indique les chemins des différents composants de l'architecture de NS.



Dans la dernière étape on intègre quelques changements aux chemins de l'environnement (ajouter quelque ligne) sur le fichier **.bashrc**

```
amel@ubuntu:~$ sudo gedit ~/.bashrc
[sudo] password for amel:
amel@ubuntu:~$
```

La figure suivante montre les lignes ajoutées **:**

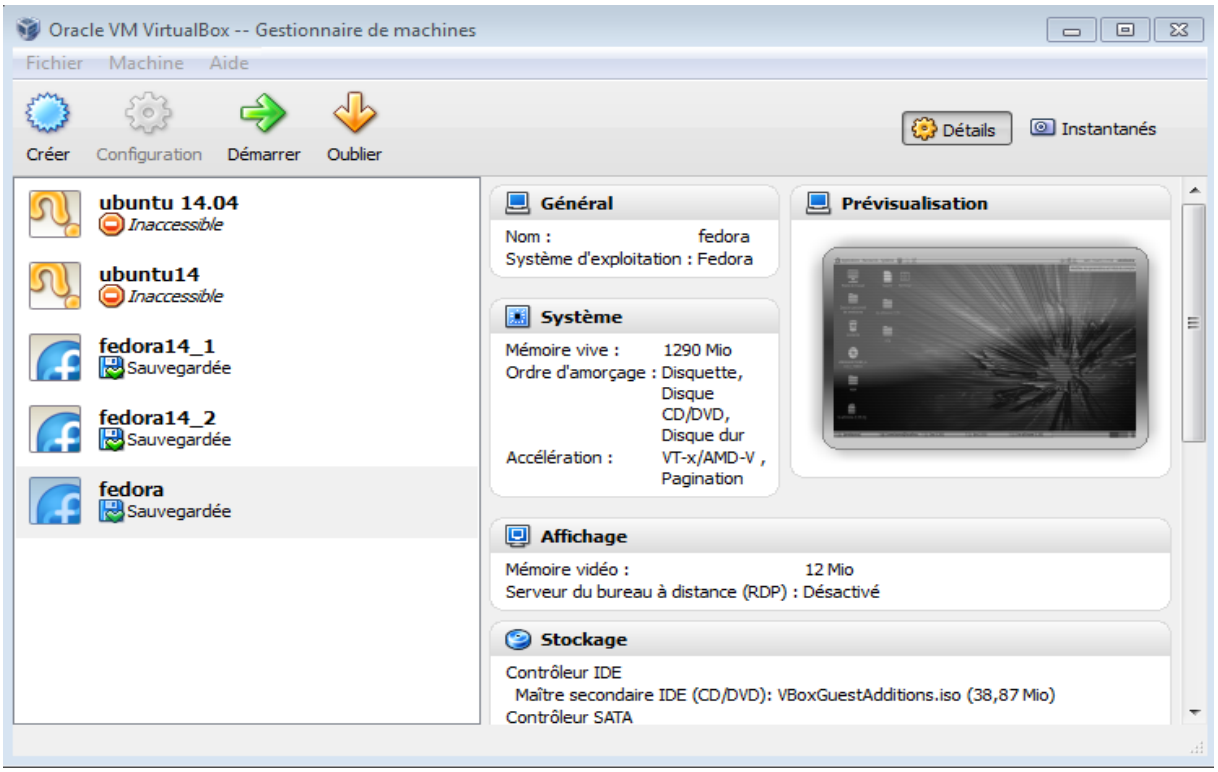
```
# LD_LIBRARY_PATH
OTCL_LIB=/home/amel/Desktop/ns-allinone-2.35/otcl-1.14
NS2_LIB=/home/amel/Desktop/ns-allinone-2.35/lib
X11_LIB=/usr/X11R6/lib
USR_LOCAL_LIB=/usr/local/lib
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$OTCL_LIB:$NS2_LIB:$X11_LIB:
$USR_LOCAL_LIB
# TCL_LIBRARY
TCL_LIB=/home/amel/Desktop/ns-allinone-2.35/tcl8.5.10/library
USR_LIB=/usr/lib
export TCL_LIBRARY=$TCL_LIB:$USR_LIB
# PATH
XGRAPH=/home/Desktop/ns-allinone-2.35/bin:/home/amel/Desktop/ns-allinone-2.35/
tcl8.5.10/unix:/home/amel/Desktop/ns-allinone-2.35/tk8.5.10/unix
#the above two lines beginning from xgraph and ending with unix should come on
the same line
NS=/home/amel/Desktop/ns-allinone-2.35/ns-2.35/
NAM=/home/amel/Desktop/ns-allinone-2.35/nam-1.15/nam
PATH=$PATH:$XGRAPH:$NS:$NAM
```

Pour vérifier que NS est accessible sur Linux, la commande « ns » doit afficher « % ».

A partir du répertoire ‘Home/Desktop/ns-allinone-2.35/ns2.35’ , lancer la dernière commande « ./validate » qui génère les fichiers « .o »(object) de NS et permet ainsi d’exécuter des simulations.

ANNEXE C

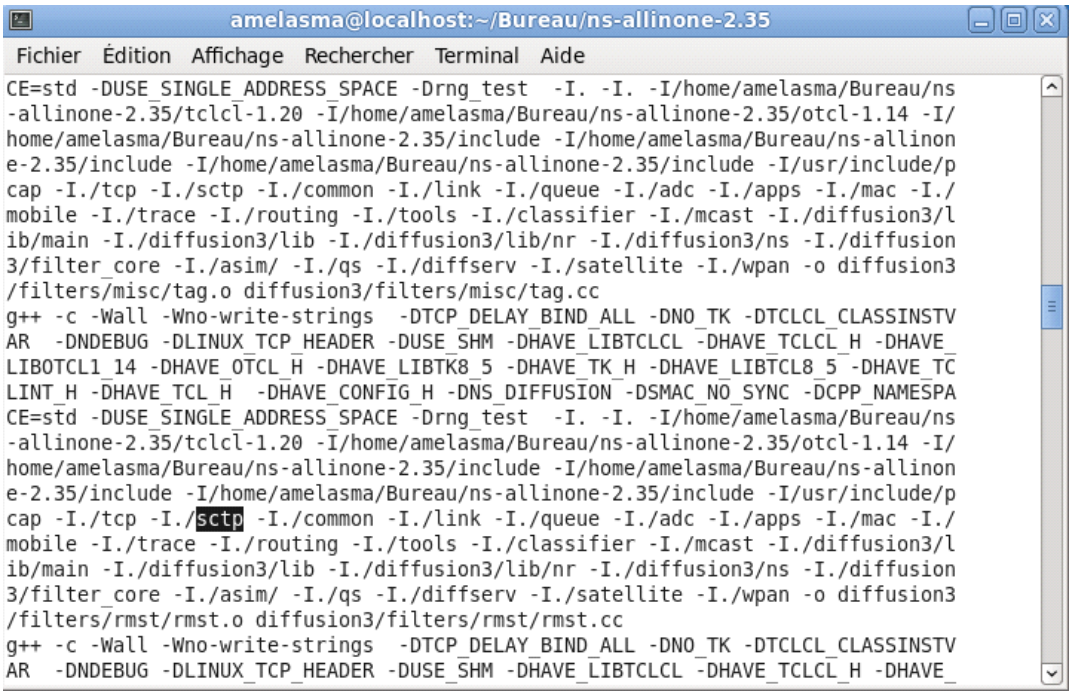
A cause d'une erreur lors l'utilisation de la machine virtuelle workstation nous avons changé notre travaille à la VM virtuale Box.



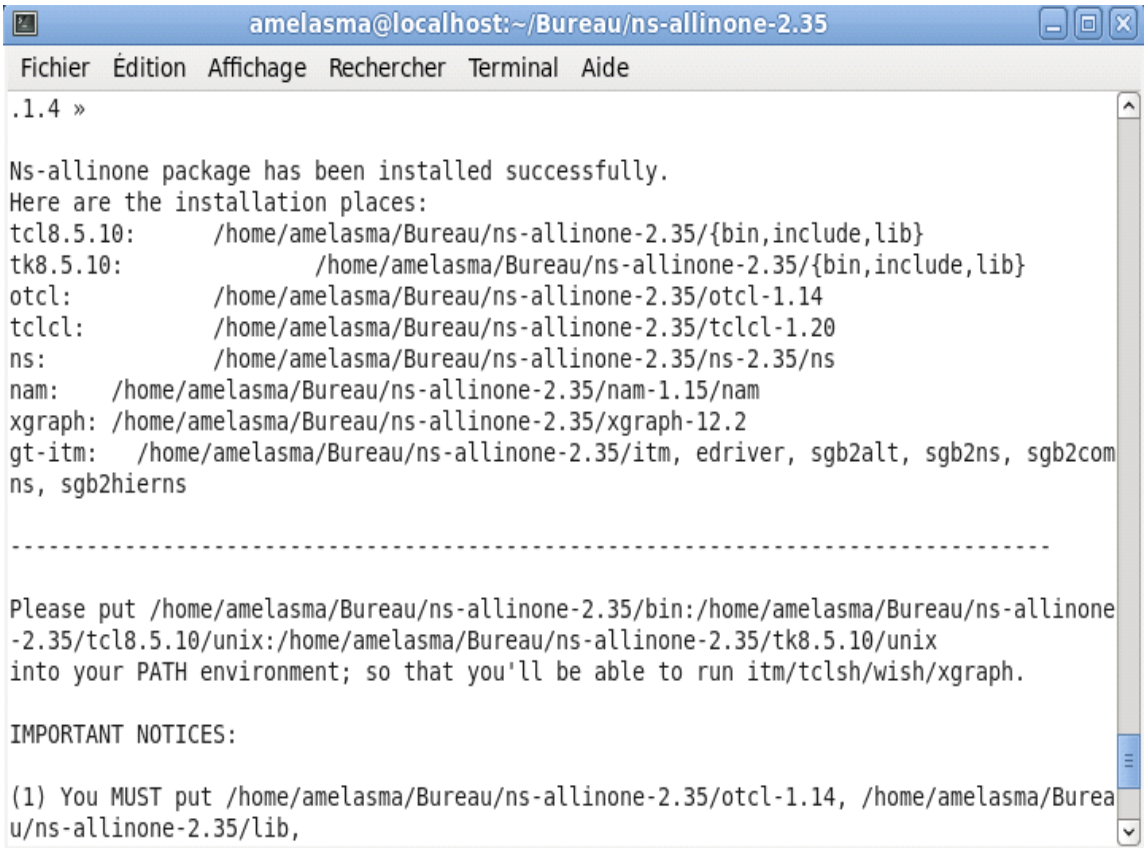
Télécharger le NS2.35 décompresser le ,installer directement avec la commande : ./install dans le terminal.

```
[amelasma@localhost Bureau]$ ls
bashrc  ns-allinone-2.35  ns-allinone-2.35.zip
[amelasma@localhost Bureau]$ cd ns-allinone-2.35
[amelasma@localhost ns-allinone-2.35]$ ./install
```

Une figure lors l'installation du NS2.35



Si l'installation est termine avec succès on va voir la figure suivante qui indique les chemins des différents composants de l'architecture de NS.



Dans la dernière étape on intègre quelques changements aux chemins de l'environnement (ajouter quelque ligne) sur le fichier **.bashrc**

Entrer au dossier personnel dans notre cas (amelasma), Affichage, afficher les fichiers cachés, ouvrir le fichier .bashrc et ajouter les ligne suivants :

.bashrc

```
# .bashrc

# ajouter les path de amel
export PATH=/usr/java/jdk1.7.0_03/bin:$PATH

# Source global definitions
if [ -f /etc/bashrc ]; then
        . /etc/bashrc
fi

# User specific aliases and functions
#LTD_LIBRARY_PATH

OTCL_LIB=/home/amelasma/Bureau/ns-allinone-2.35/otcl-1.14

NS2_LIB=/home/amelasma/Bureau/ns-allinone-2.35/lib

X11_LIB=/usr/X11R6/lib

USR_LOCAL_LIB=/usr/local/lib

export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$OTCL_LIB:$NS2_LIB:$USR_LOCAL_LIB


# TCL_LIBRARY
```

```
TCL_LIB=/home/amelasma/Bureau/ns-allinone-2.35/tcl8.5.10/library

USR_LIB=/usr/lib

export TCL_LIBRARY=$TCL_LIB:$USR_LIB


# PATH

XGRAPH=/home/amelasma/Bureau/ns-allinone-2.35/bin:/home/amelasma/Bureau/
ns-allinone-2.35/tcl8.5.10/unix:/home/amelasma/Bureau/ns-allinone-2.35/tk8.5.10/unix

NS=/home/amelasma/Bureau/ns-allinone-2.35/ns-2.35/

NAM=/home/amelasma/Bureau/ns-allinone-2.35/nam-1.15/
```

Pour vérifier que NS est accessible sur Linux, la commande « ns » doit afficher « % ». partir du répertoire ‘Home/Bureau /ns-allinone-2.35/ns-2.35’, lancer la dernière commande « ./validate » qui génère les fichiers « .o »(object) de NS et permet ainsi d’exécuter des simulations.

ANNEXE D

Pour le type de données FTP :

```
Trace set show_sctphdr_ 1
################################### création d'un simulateur
set ns [new Simulator]

################################ création du fichier de trace utilisé par le visualisateur
##### ########################et indication à ns de l'utiliser

      set nf [open all2.nam w]
      $ns namtrace-all $nf
       set allchan [open all2.tr w]
       $ns trace-all $allchan

################################# lorsque la simulation sera terminée, cette procédure
est appelée pour lancer automatiquement la visualisation #################
         proc finish {} {
         global ns allchan
         $ns flush-trace
         close $allchan
         exec nam all2.nam &
         exit 0
                  }

################################################création des nœuds############
set n0 [$ns node]
$n0 label "client"
$n0 color red
set n1 [$ns node]
$n1 label "serveur"
$n1 color blue
puts "utilise 3stream..."
################################Le lien entre ces deux nœuds est un duplex-linkde
5Mb de débit  et de retard égaleà200ms############################

         $ns duplex-link $n0 $n1 5Mb 200ms DropTail
          $ns duplex-link-op $n0 $n1 orient right


##############################
         set sctp0 [new Agent/SCTP]
         $ns attach-agent $n0 $sctp0
```

```
        $sctp0 set mtu_ 1500
        $sctp0 set dataChunkSize_ 1468
        $sctp0 set numOutStreams_ 3
        $sctp0 set initialCwnd_ 2
##########################
        set sctp1 [new Agent/SCTP]
        $ns attach-agent $n1 $sctp1
        $sctp1 set mtu_ 1500
        $sctp1 set initialRwnd_ 131072
#######################################
  ############################le trafic issus de l'agent sctp0 est envoyé vers sctp1
        $ns connect $sctp0 $sctp1
        $sctp1 listen

#Au niveau applicatif nous considérons comme application le transfert de fichier simple
FTP############

       set ftp0 [new Application/FTP]
       $ftp0 attach-agent $sctp0

###################### scénario de début et de fin de génération des paquets par sctp0

      $ns at 0.5 "$ftp0 start"
      $ns at 20.0 "$ftp0 stop"

########################## la simulation va durer 55 secondes de temps simulé

      $ns at 55.0 "finish"
#################################
###################################Definition de classes pour la coloration
$sctp0 set class_ 1
$sctp1 set class_ 2
###################################Coloration des classes : bleu pour sctp0 (classe 1) et
rouge pour sctp1
$ns color 1 Red
$ns color 2 Blue
################################ début de la simulation
puts "Starting Simulation..."
         $ns run
```

Pour le type de données SctpApp1 :

```
Trace set show_sctphdr_ 1
set ns [new Simulator]
set nf [open sctpc.nam w]
$ns namtrace-all $nf
set allchan [open sctpc.tr w]
$ns trace-all $allchan

proc finish {} {
        global ns allchan
        $ns flush-trace
        close $allchan
        exec nam sctpc.nam &
        exit 0
                }

set n0 [$ns node]
$n0 label "client"
$n0 color red
set n1 [$ns node]
$n1 label "serveur"
$n1 color blue
puts "utilise 3 stream..."
###################
$ns duplex-link $n0 $n1 5Mb 200ms DropTail
$ns duplex-link-op $n0 $n1 orient right
puts "Loading scenario file..."
####################
set sctp0 [new Agent/SCTP]
$ns attach-agent $n0 $sctp0
$sctp0 set mtu_ 1500
$sctp0 set initialCwnd_ 2
#################
set sctp1 [new Agent/SCTP]
$ns attach-agent $n1 $sctp1
$sctp1 set mtu_ 1500
$sctp1 set initialRwnd_ 131072
################
$ns connect $sctp0 $sctp1
$sctp1 listen
####################
set SctpApp0 [new Application/SctpApp1]
$SctpApp0 set numStreams_ 3
```

```
$SctpApp0 set interval_ 0
$SctpApp0 attach-agent $sctp0
############################
$ns at 0.5 "$SctpApp0 start"
$ns at 20.0 "$SctpApp0 stop"
$ns at 55.0 "finish"
$sctp0 set class_ 1
$sctp1 set class_ 2
############################
$ns color 1 Red
$ns color 2 Blue
puts "Starting Simulation..."
$ns run
```

Bibliographie et Webographie

[1] François Buntschu, Aron Bernasconi, «Stream Control Transmission Protocol (SCTP)», Projet Ra&DHES-SO Objet No : TI01-02, Août 2003.

[2] Youssouf M'madiDjoumbe, «Audit et analyse de la qualité et de la performance du réseau de signalisation SS7/SIGTRAN de la sonatel», Mémoire ingéniorat, École supérieure multinationale des télécommunications, 2011.

[3] Randall Stewart, Paul D. Amer, « Why is SCTP needed given TCP and UDP are widely available? » , Internet Society (www.isoc.org), September 2007.

[4]Talbi Hamza , «Architecture et politique de robustesse par le multihoming dans les réseaux ad hoc et les réseaux satellitaires» ,Thèse de Magistère ,Université Constantine1.

[5] Scott Ruffin, « Réseaux (I) : Modèle de référence OSI» , Note technique 4D-200005-16-FR, Version 1,1 Mai 2000.

[6]Dupessey Xavier, Glossi Etienne, Latrille-Debat Simon Et Viougeas Eric « Etude des simulateurs de routage pour réseaux sans fil », Institut universitaire de technologie de Valence IUT, Juin 2008.

[7]« Définition des sept couches du modèle OSI et explication de leurs fonctions», (https://support.microsoft.com/fr-fr/kb/103884), Révision 21, 2015.

[8]Ahmed Mehaoua, «Couche Transport Tcp Et Udp», support de cours (http://www.mi.parisdescartes.fr/~mea/cours.htm), Université Paris Descartes, Décembre 2009.

[9] «Chapitre 4 : Couche Transport OSI», support de cours (http://cttcb.fr/cardoni_ancien_site/exploration1/c4.htm).

[10] « TCP /IP» (http://home.scarlet.be/vinderick.pascal/reseaux_wan/protocoles.htm).

[11]Jean François Pillou «Le protocole TCP » (http://www.commentcamarche.net/contents/538-le-protocole-tcp.htm) , Mai 2016.

[12] Laurent Baysse, «Tcp/Ip», support de cours (http://csricted.univ-setif.dz/files/cours.htm), Setif, Mars 2005.

[13]Rami Langar, «Couche transport», support de cours (https://wwwphare.lip6.fr/~langar/

Polytech_courses.htm).

[14] Abdellatif Ezzouhairi, « Intégration et gestion de mobilité de bout en bout dans les réseaux mobiles de prochaine génération », Université de Montréal, Décembre 2009.

[15] «Sctp: Stream Control Transmission Protocol Protocole et services», (http://www.efort.

com/r_tutoriels/SCTP_EFORT.htm), 2009.

[16] Meriem Afif, « Interaction des mécanismes RLC/MAC et de sctp dans les réseaux mobiles B3G », Thèse de doctorat, École Nationale Supérieure des Télécommunications de Paris, Novembre 2007.

[17] Antoine.Delley, support de cours, http://antoine.delley.home.hefr.ch/didacticiel_tif-fr/8.5.html.

[18] Pawel Hadam, « Transports nouvelle génération dans les réseaux a très haut débit », Thèse de Doctorat, Institut National Polytechnique de Grenoble - Inpg, 2005.

[19] Olga Antonova, «Introduction and comparison of SCTP, TCP-Mh, DCCP Protocols», Université de Helsinki ,-Hut T-110.551 seminar on internetworking, Avril 2004.

[20] Ertuğrul Yılmaz,«NR-SACKs (Non-Renegable Selective Acknowledgments)»,Université de Delaware , November 2007.

[21] Sakuna Charoenpanyasak, « Optimisation inter-couches du protocole SCTP en réseaux Ad Hoc», Thèse de doctorat , Université De Toulouse , Juin2008 .

[22] Mohamed Rabie Naimi, «Evaluation des performances de mobile SCTP dans l'equilibrage de charge basée sur le type de flux dans les réseaux Nemo », Mémoire de Magister, Université Abou Bekr Belkaid Tlemcen , Février 2012.

[23] Guillaume Auriol, « Spécification et implémentation d'une architecture de signalisation a gestion automatique de la QDS dans un environnement IP multidomaines», Thèse de Doctorat, Institut National des Sciences Appliquées de Toulouse, 2004.

[24] Elyès Ben Hamida « Modélisation stochastique et simulation des réseaux sans fil multi-sauts »,Thèse Doctorat, Institut National des Sciences Appliquées de Lyon ,2009.

[25]Mounir-ssada, « Chapitre VI :Simulation et discussion des résultats» (http://docslide.fr/documents/ 6-simulation-et-discussion-des-resultats.html), juillet 2015.

[26] Kherbache Zeyneb et LARIBI Amina «Étude de la Qualité de Service (QoS) dans les réseaux WIFI » Mémoire de Master, Université Abou Bakr Belkaid Tlemcen 2011.

[27] Louati Chaimae, Elhoussaini Smail, Wann Ibrahima, Hazaoud Alae «Simulateur interactif de la qualité de service dans un routeur », Rapport de Master, Centre d'Enseignement et de Recherche en Informatique, 2013.

[28] Saadane Houda « La qualité de service d'un streaming vidéo dans un réseau ad hoc (égale à égal) », mémoire de Magister, Université Badji Mokhtar Annaba,2012.

[29] Aina Andriamampianina, Randrianarisaina « Modélisation de la consommation d'énergie en vue de la conception conjointe (matériel/logiciel) Des Applications Embarquées. Application Aux Réseaux De Capteurs Sans Fil (Wsn). », Thèse de doctorat, Université de Nantes, février2015.

[30]Jarmo Prokkola , « OPNET - Network Simulator» Rapport de projet , Centre de Recherche Technique VTT de Finland , University de Oulu .

[31] Sidi Ykhlef Asma Et Kebir Khadidja « Modélisation et simulation d'un réseau en utilisant OPNET modeler » Mémoire de Licence, Université Abou Bakr Belkaid– Tlemcen, Juin 2011.

[32] Adrien Van Den Bossche« Proposition d'une nouvelle méthode d'accès déterministe pour un réseau personnel sans fil à fortes contraintes temporelles », Thèse de Doctorat , Université de Toulouse II, juillet 2007.

[33] Youssef Baddi, «Introduction au simulateur réseau NS2», (http://ybaddi.developpez.com / tutoriels/ns2/),2011.
[34] Vivien Boistuaud Et Julien Herr, «Modélisation de réseaux avec le logiciel NS2 », Rapport de TP, l'Université de Marne-la-vallée, 2000.

[35] F.Boumghar, W.Bacha « Simulateurs de réseaux Ns-3 et Ns-2 »,(http://slidegur.com /doc/40917/simulateurs -de-r%C3%A9seaux-ns-3-et-ns-2.htm),Université des Sciences et de la Technologie Houari Boumediene, 2014.

[36] Sedjelmaci Amina Nadjet «Extension de la QoS du Wifi vers le WiMAX », Mémoire de Magister en Systèmes et Réseaux de Télécommunication, Université Abou Bakr Belkaid– Tlemcen,2010-2011.

[37] Babaameur Dalila, Boumghar Fériel Célia, Bacha Wissem Et Chaiani Mounira «Les Simulateurs NS3&NS2 », rapport de projet , Université Houari Boumediene, 2014.

[38] Mohamed Djihad Ben Salem Et Oussama Bougoffa « Etude comparative de deux simulateurs pour les réseaux ad-hoc sans fil », mémoire de Master, Université Kasdi Merbah Ouargla, 2013.

[39] Tran Alexandre Et Layouni Majid, «Simulator Network 2 NS-2 », Rapport de projet de master, Université d'avignon, 2009.

[40] Béchu Jérôme «OBS et NS-2 », Rapport de stage, UQAM Montréal, eXia 2006-2007.

[41] Labeni Yacine Et Boulkour Abdelouahid «Outil pour l'analyse des traces de simulation des protocoles de routage ad hoc avec NS-2 »Mémoire de Master en Informatique, Université Mohamed Sadik Benyahia-Jijel,2014.

[42] Nadeen Salameh,«Conception d'un système d'alerte embarqué basé sur les communications entre véhicules » Thèse de doctorat, Institut National des Sciences Appliquées de Rouen ,2011.

[43] Isabelle Vollant, «la commande awk **»,** **(**http://www.shellunix.com/awk.html), juillet 2012.

[44] Trace files and description (http://nile.wpi.edu/NS/analysis.html).

[45] Gerard J. Heinz II «Priorities in stream transmission», Thèse de Master en Informatique et sciences de l'information, Université de Delaware, 2003.

[46]Yaogong Wang, Injong Rhee, «Augment SCTP multi-Streaming with pluggable scheduling », IEEE Computer Communications Workshops (INFOCOMWKSHPS), 2011.

[47] Meriem laafif, Sami Tabbane, «Inter-stream priority management in SCTP multi-streaming for HTTP traffic in UMTS», ICTTA.3rd International Conference, 2008.

[48] http://www.isi.edu/nsnam/ns/doc.