N° d'ordre : 06/TCO/STI Année Universitaire : 2007 / 2008

UNIVERSITE D'ANTANANARIVO

ECOLE SUPERIEURE POLYTECHNIQUE

DEPARTEMENT TELECOMMUNICATION

MEMOIRE DE FIN D'ETUDES

en vue de l'obtention

du DIPLOME D'INGENIORAT

Spécialité : Télécommunication

Option : Système de Traitement de l'Information

par : **RANDRIAMITANTSOA Andry Auguste**

PLANIFICATION, MODELISATION ET SIMULATION DES RESEAUX PAR LE PROTOCOLE MPLS.

Soutenu le 16 Janvier 2009 devant la Commission d'Examen composée de :

Président : Monsieur ANDRIAMIASY Zidora

Examinateurs :

Monsieur RAZAKARIVONY Jules

Monsieur RANDRIARIJAONA Lucien Elino

Monsieur RASAMOELINA Jacques Nirina

Directeur de mémoire : Monsieur RAKOTOMALALA Mamy Alain

REMERCIEMENTS

Tout d'abord, j'adresse ma profonde gratitude à Dieu tout puissant pour la Grâce qu'Il m'a accordée ainsi que pour la connaissance, le courage et la santé qu'Il m'a octroyés pour la réalisation de ce mémoire et à l'avenir.

Je témoigne toute ma reconnaissance :

À Monsieur RAMANANTSIZEHENA Pascal, Professeur Titulaire, Directeur de L'Ecole Supérieure Polytechnique d'Antananarivo,

À Monsieur RANDRIAMITANTSOA Paul Auguste, Professeur, Chef de Département Télécommunication au sein de l'Ecole Supérieure Polytechnique d'Antananarivo, qui m'ont accueilli à l'Ecole Supérieure Polytechnique d'Antananarivo, sans quoi nous n'aurions pas pu arriver à cette finalité ;

À Monsieur RAKOTOMALALA Mamy Alain, Enseignant Chercheur, mon encadreur pédagogique qui m'a énormément aidé au cours de cette étude malgré les fonctions diverses auxquelles il est rattaché. Je le remercie vivement pour son dévouement et les conseils qu'il m'a prodigués au cours de cette investigation.

Ma reconnaissance va également :

À Monsieur ANDRIAMIASY Zidora, Maître de conférences à l'ESPA, pour sa disposition de bien vouloir juger ce travail en dépit de ces occupations multiples.

À Monsieur RAZAKARIVONY Jules, Maître de conférences l'ESPA, qui a consacré son temps précieux pour juger ce travail.

À Monsieur RANDRIARIJAONA Lucien Elino, Enseignant Chercheur, qui a accordé son temps précieux pour juger le présent travail.

À Monsieur RASAMOELINA Jacques Nirina, Enseignant Chercheur, qui a bien voulu juger ce mémoire malgré ses diverses occupations.

Je suis profondément reconnaissant envers mes parents qui m'ont soutenu moralement et financièrement durant toutes mes études et surtout lors de l'accomplissement de cette recherche. Je leur dédie ce mémoire. Aussi je remercie vivement toute ma famille qui m'a aidée.

Enfin, je ne saurai oublier toutes les personnes qui m'ont amplement aidée de près ou de loin dans l'élaboration du présent mémoire.

Table des matières

NOTATIONS

C_{kd}	Capacité du lien individuel k
C_{kv}	La capacité du lien individuel pour le trafic voix
D_{down_i}	Débit des liens descendant au niveau de chaque site
D_{up_i}	Débit du lien montant au niveau de chaque site
$Débit_{apl}$	Débit par appel en Kbit/s
$N_{apl_{i\to j}}$	Nombre d'appel entre le nœud i et le nœud j
$P_k(t)$	Probabilité d'observation de k à l'instant t
P_n	Probabilité de blocage
$S_{i\to j}$	Signalisation entre deux sites
S_{ij}	Signalisation full-duplex
T_{ijv}	Le trafic entre les sites i et j
T_{tra}	Matrice de trafic
λ	Taux d'arrivée

ABREVIATIONS

ADSL	Asymetric Digital Subscriber Line
AF-PHB	Assured Forwarding PHB
ATM	Asynchronous Transmission Mode
BGP	Border Gateway Protocol
BIN	Backbone IP National
BMAP	Batch Markovian Arrival Process
BS	Bottom Stacking
CE	Client Edge
CL	Controlled Load
CoS	Class of Services
CR-LDP	Constraint-based Routing LDP
DiffServ	Differentiated Services
DSF	Differentiated Services Field
EF-PHB	Expedite Forwarding PHB
EIGRP	Enhanced Interior Gateway muting Protocol
FEC	Forwarding Equivalent Classes
FIB	Forwarding Information Base
FR	Frame Relay
GMPLS	Generalized MultiProtocol Label Switching
GS	Guaranted Service

HDLC	High level Data Link Control
HTTP	HyperText Transfer Protocol
IETF	Internet Engineering Task Force
IGP	Interior Gateway Protocol
IGRP	Interior Gateway Routing Protocol
IntServ	Integrated Services
IP	Internet Protocol
IS-IS	Intermediate System to Intermediate System
LDP	Label Distribution Protocol
LER	Label Edge Router
LFIB	Label Forwarding Information Base
LIB	Label Information Base
LLC/SNAP	Logical Link Control/SubNetwork Access Protocol
LSP	Label Switch Path
LSR	Label Switching Router
MAM	Maximum Allocation Multiplier
MIC	Modulation par Impulsion et Codage
MP-BGP	MultiProtocol BGP
MPLS	MultiProtocol Label Switching
OSPF	Open Shortest Path First
PE	Provider Edge
PHB	Per Hop Behavior

PIM	Protocol Independent Multicast
PPP	Point to Point Protocol
PVC	Permanent Virtual Channel
Qos	Quality of Service
RIP	Routing Information Protocol
RSVP	Resource reSerVation Protocol
RTP	Real Time Transport Protocol
SLA	Service Level Agreement
SNMP	Simple Mail Transfer Protocol
SPF	Short Path First
TDP	Tag Distribution Protocol
TE	Traffic Engineering
TE-LSA	Traffic Engineering Link-State Advertissement
TTL	Time To Live
UDP	User Datagram Protocol
VLSM	Variable Length Subnet Mask
VPI/VCI	Virtual Path Identifier/Virtual Channel Identifier
WDM	Wavelength Division Multiplexing

INTRODUCTION

Actuellement, les réseaux de communication connaissent un essor considérable. L'augmentation du nombre d'utilisateurs et l'accroissement conséquent du trafic de réseau, contribuent au développement des nouvelles technologies et au déploiement des réseaux haut-débit. Par ailleurs, les besoins en matière de service évoluent incessamment et les concurrences entre les divers opérateurs commencent à apparaître.

La technologie MPLS, initié par l'IETF en 1997, compte parmi ces nouvelles technologies. Elle se rapporte principalement sur l'amélioration de la qualité du réseau. D'où l'intitulé du présent devoir : « La planification, la modélisation et la simulation des réseaux par le protocole MPLS ».

Dans cette introduction, nous allons présenter brièvement la technologie MPLS car nous allons l'étudier en détail dans ce mémoire :

- MPLS est un standard de l'IETF associant les avantages de routage et ceux de la commutation. MPLS traduit la volonté de l'IETF d'améliorer les débits grâce à l'optimisation de la complexité du traitement des paquets dans le réseau.
- MPLS se fonde sur le concept de commutation de labels. Il s'agit d'insérer une étiquette entre les entêtes des couches 2 et 3. Les décisions d'acheminement des paquets se font alors sur la base de cette étiquette au lieu de remonter à de couche 3 et effectuer une consultation de la table de routage.

Grâce à sa puissance de commutation et à l'intégration de plusieurs fonctionnalités telles que la qualité de service, l'ingénierie du trafic ou les VPN, MPLS a pu s'imposer dans les réseaux IP actuels comme étant une technologie capable de satisfaire les nouvelles applications dans le domaine de la télécommunication.

L'étude de ce protocole MPLS s'étendra sur 5 chapitres. La Planification, la Modélisation et le Dimensionnement des réseaux seront exposés en premier lieu dans le premier chapitre. Ensuite, le second chapitre parlera de la MPLS et l'ingénierie de trafic. Le chapitre 3 offre un Dimensionnement des artères du backbone MPLS. Le quatrième chapitre présentera ensuite, les Outils de simulation pour l'étude des performances des réseaux. Enfin, le dernier chapitre donne une simulation sous NS-2.

CHAPITRE 1: PLANIFICATION, MODELISATION ET DIMENSIONNEMENT DES RESEAUX.

1.1. La modélisation des réseaux : [1] [2]

Un réseau et un commutateur sont des objets physiques complexes dont l'analyse totale est difficile. Cependant, la plupart de leurs caractéristiques sont inutiles pour traiter un problème de trafic. La modélisation se trouve alors l'approche adaptée à ces questions.

Modéliser signifie construire une abstraction simplifiée qui conserve uniquement les particularités des phénomènes à étudier, en partant d'une description complète du système réel. La modélisation est une technique délicate et empirique : un artisanat. Quelques exemples, pris dans la pratique de développement et de l'exploitation des réseaux de télécommunications modernes, peuvent en montrer la puissance et l'intérêt.

Dans ce chapitre le réseau sera étudié indépendamment des solutions techniques adoptées pour son implémentation (fibres optiques, liaisons radio, IP ou ATM …). Le réseau apparaît simplement comme un graphe, reliant des nœuds (commutateurs/routeurs) et desservant des terminaux. Sa fonction consiste à relier les terminaux et leur permettre des échanges d'informations appelé trafic. La première tâche définit en quoi consiste le trafic. Afin d'expliciter cette notion, il sera nécessaire d'examiner plus en détail le type de réseau et surtout le service d'interconnexion envisagé.

En général, les éléments de la théorie demeurent insuffisants pour traiter entièrement les problèmes de dimensionnement. Nous aurons alors recours à la simulation, considérée comme un complément à l'approche mathématique, les deux méthodes se confortant mutuellement. L'usage conjoint de ces deux approches aboutira à un bon niveau de sécurité dans les prévisions. Toutefois, il ne suffit pas d'estimer le niveau de ressources nécessaire ; cette estimation s'intègre dans une démarche plus générale de planification.

Le responsable du réseau commence d'abord par estimer la demande (son volume) et prévoir son évolution, pour dimensionner ensuite les ressources, et enfin organiser leur mise en service. La structure du réseau, entre autres sa taille, sont des sujets d'étude pour la planification.

1.2. La qualité de service dans la conception des réseaux : [1] [2]

La notion de qualité de service veut rendre compte, de façon chiffrée, le niveau de performances que l'utilisateur attend du réseau. La notion de QoS, sa mesure, sa vérification sont des tendances modernes ; c'est-à-dire que l'apparition des mécanismes de la dérégulation leur donne une importance croissante. La signification précise de la notion de qualité de service dépend évidemment du service envisagé. Ainsi le service a-t-il des exigences de temps de réponse ? Quelle est sa sensibilité aux erreurs de transmission ? Etc. Une définition complète se réfèrera souvent au mode de transport de l'information circuit ou paquet, bien que la solution adoptée par le réseau, pour rendre le service, doit rester transparente à l'utilisateur.

1.2.1. Réseau à commutation de circuits :

Le réseau téléphonique est un réseau de type à commutation de circuits. Une demande acceptée se voit offrir à son usage exclusif, un circuit (circuit électrique continu, ou intervalle de temps d'une trame MIC) pour toute la durée de la connexion. La QoS se définit en premier lieu par la possibilité de ne pas obtenir de circuit lors de la demande. Elle est mesurée par la probabilité d'échec. D'autres critères, liés aux technologies, interviendront dans la QoS. Ils ne concernent pas notre propos, et sont les critères liés à la qualité électrique de la liaison (atténuation du signal, distorsions).

1.2.2. Réseau à commutation par paquets:

Dans un réseau offrant le service de commutation de paquets, l'information des sources est fragmentée en blocs élémentaires qui voyagent dans le réseau indépendamment les uns des autres. Ces blocs ne possèdent aucune ressource en propre comme dans le cas du circuit. Les paquets d'une connexion se retrouvent alors en compétition avec d'autres, pour accéder aux mémoires ou aux lignes de transmission. Il en résulte un critère supplémentaire de qualité de service, lié au sort individuel de chaque paquet, qui subira un retard variable et pourra même être perdu. Le cas où l'une des mémoires qu'il doit traverser est saturée peut illustrer ce fait. La QoS est mesurée par la probabilité de perte de paquets et par le délai spécifiée par sa moyenne, ou plus précisément, par sa fonction de distribution.

A ses débuts, Internet avait pour seul objectif de transmettre les paquets à leurs destinations. Conçu pour le transport des données, IP (Internet Protocol) n'a pas été prévu pour les applications en temps réel. Le besoin en équipements fiables, d'un bout à l'autre du réseau devient alors

incontournable. Cependant, les inconvénients rencontrés dans les réseaux (perte de paquets, congestion, etc.) ne peuvent pas être surmontés sans une rénovation profonde de l'architecture.

La QoS d'une application se définit comme étant l'ensemble des exigences requises par cette application en termes de bande passante, délai, gigue et taux de perte. Ces exigences sont intrinsèques à la nature des applications. Ces dernières peuvent être classées ainsi en deux catégories :

- *les applications temps réel* : qui ont un besoin ferme en termes de délai et de faible variation de délai (gigue) ; elles peuvent tolérer quelques pertes de paquets ;

- *les applications non temps réel ou Best-Effort* : sont plus sensibles aux pertes de paquets (par rejet) qu'à des délais élevés. Dans ce cas, la fiabilité de la communication est plus importante que la garantie d'un délai borné.

Ainsi, la qualité de service est la méthode de garantie à un trafic de données, les meilleures conditions d'acheminement répondant à des exigences prédéfinies, quelle que soit sa nature. Elle fixe notamment des règles de priorité entre les différents flux.

Actuellement, la plupart des réseaux possède une ou plusieurs liaisons critiques qui dégradent les performances des applications. Ce problème vient du fait qu'il n'y a pas de distinction entre le trafic prioritaire et le trafic non prioritaire. La différenciation des services propose alors une solution qui consiste à séparer les deux types de trafic sur les liens critiques. Au cas où le réseau serait saturé, une alternative à l'augmentation de la bande passante constitue donc la différenciation du trafic. Maintenant, l'IETF propose deux modèles pour cela tels qu'IntServ et DiffServ. Les modèles IntServ, Diffserv et MPLS définissent une configuration des mécanismes suite à l'identification des besoins généraux de QoS.

1.3. Les solutions possibles pour dimensionner un réseau multiservice: [3]

Plusieurs mécanismes peuvent être envisageables pour satisfaire les besoins de certaines applications en qualité de services, tels que :

- le surdimensionnement du réseau ;
- la réservation de ressources et;
- le traitement différencié.

Les deux derniers mécanismes peuvent être intégrés pour fournir une meilleure solution.

1.3.1. Le surdimensionnement du réseau :

Le surdimensionnement du réseau signifie la conception du réseau de manière à ce qu'il soit capable de transmettre à tout moment les volumes de trafic et de satisfaire leurs besoins en qualité de service. En général, le choix du mécanisme dépend du pourcentage du type de trafic. Si tout le trafic est sensible au délai et nécessite un traitement prioritaire, alors le surdimensionnement reste la seule alternative.

1.3.1.1. Pourquoi surdimensionner le réseau ?

D’une part, la dégradation des performances due au fait qu'à tout moment une liaison physique peut tomber en panne rend parfois nécessaire le surdimensionnement du réseau. Dans les réseaux IP transportant aussi bien le trafic data que le trafic multimédia, l'infrastructure IP nécessite un surdimensionnement pour permettre le trafic temps réel.

D’autre part, le surdimensionnement peut être une solution au problème de congestion. En effet, laisser une marge de ressources disponibles dans le réseau permet, en cas de transmissions en rafales, d'éviter la congestion.

1.3.1.2. Les inconvénients du surdimensionnement :

Prenons le cas d’un réseau qui transporte à la fois la VoIP, un trafic HTTP (HyperText Transfer Protocol) et SNMP (Simple Mail Transfer Protocol). Dans ce cas, il n’est pas économique de dimensionner le réseau de façon à ce qu'il fournisse la plus large bande passante. Effectivement, les équipements à mettre en place sont couteux. En d'autres termes, chaque augmentation dans la bande passante entraîne un investissement supplémentaire du point de vue équipements matériels.

1.3.2. La réservation de ressources :

La réservation de ressource est un autre moyen de dimensionner un réseau. Elle signifie qu'une proportion de la capacité du réseau est allouée à une classe de service. Cette ressource est réservée le long de la route empruntée par un trafic donné. La réservation de capacité est prédéfinie.

La réservation de capacité peut être :

- *de type physique* :

Dans ce cas une partie de la capacité au niveau de la couche physique est allouée à un service donné. Par exemple, dans les réseaux à lien optique utilisant WDM (Wavelength Division Multiplexing), une longueur d'onde peut être réservée pour les appels téléphoniques dans les réseaux à commutation de circuit entre deux centres de commutation ;

- *signalée semi-permanente :*

La réservation permanente pour une classe de service peut être établie avec des protocoles de signalisation appropriés. C'est le cas par exemple des circuits virtuels permanent PVC de l'ATM ;

- *signalée sur demande :*

La réservation est établie sur demande. C'est le cas de la signalisation RSVP le long du chemin appliqué pour les flux individuels ;

- *réservation statistique* :

La réservation est interprétée en utilisant un terme statistique prédéfini tel que la disponibilité durant 95 % du temps.

1.3.2.1. Le modèle IntServ :

Définition:

Le modèle IntServ (Services Intégrés) tend au respect de l'intégrité des flux, c'est une transposition des techniques de CoS (Class of Services) des réseaux à état dit dur (hard-state) tel qu'ATM et Frame Relay. A l'instar de ces protocoles, IntServ définit un état dit mou (soft-state) via un protocole de signalisation (RSVP). Un réseau à état mou est un réseau qui maintient un contexte durant un certain temps. En l'absence de renouvellement périodique cet état est détruit.

L'exploitation commerciale de nouvelles technologies, telle que la voix sur IP, ne pourra pas débuté avec un réseau internet peu performant que si des mécanismes lui sont implémentés, élevant et garantissant ainsi le niveau de performance. Les trafics temps réel, tels que la voix et la vidéo, peuvent utiliser une réservation de ressources pour être sûr de bénéficier de la qualité nécessaire. Le modèle *IntServ* se base sur l'allocation de ressources selon la demande de l'utilisateur. Les ressources restent alors allouées pendant toute la durée de transmission sans être affectées par un trafic IP « Best Effort ».

1.3.2.2. Le protocole RSVP :

Définition :

Il autorise pour les flux multimédias une réservation de ressources réseau de bout en bout, ce protocole permet la cohabitation de flux multimédia et de flux sporadiques non prioritaires.

Le protocole RSVP est le protocole de réservation qui a été conçu au départ pour fonctionner avec le modèle IntServ. Le protocole RSVP permet, par exemple dans le cas de la téléphonie sur IP, de réserver des ressources pour chaque communication. Mais d'une façon générale, le protocole RSVP reste un protocole de signalisation permettant l'émission d'une demande d'allocation de bande passante à tous les nœuds sur le chemin d'un flot donné.

1.3.3. La différenciation de services :

Définition :

L'approche DiffServ est conforme à l'approche IP. Diffserv implémente un mécanisme de partage de bande passante en introduisant la notion de politique d'acheminement en fonction d'une classe de service. Les flux ne sont plus traités individuellement comme dans IntServ, mais seront attribués à une classe de service identifiée par un champ spécifique *Differentiated Services Field (DSF).* Tous les flux d'une même classe sont traités de la même façon dans le réseau.

Le modèle de différenciation de services semble être plus adéquat pour les réseaux multiservices tels que l'internet. Autrement, ce modèle signifie donner la priorité à une classe de service au dépend d'une autre classe dans le cas d'une congestion. Le modèle DiffServ (Differentiated Services) définit une approche totalement différente par rapport au modèle IntServ (Integrated Services). Il ne nécessite ni une réservation de bout en bout ni signalisation et permet d'affecter chaque paquet à une classe de service. La complexité est reléguée dans les extrémités du réseau. En outre, les services différenciés de l'architecture DiffServ diminuent substantiellement les informations d'état que chaque nœud du réseau doit mémoriser.

Dans l'architecture DiffServ, le traitement différencié des paquets s'appuie sur trois opérations fondamentales :

- la classification des flux en classes de services ;
- l'introduction de priorités au sein des classes (Scheduling) ;
- la gestion du trafic dans une classe donnée (Queue management).

De ce fait, les paquets sont classés grâce à un mécanisme de marquage d'octets dans l'en-tête des paquets IP. Les services qui sont octroyés par les routeurs à ces paquets dépendent des classes définies.

1.3.4. Le protocole MPLS : réservation de capacité et traitement différencié :

Définition :

MPLS permet un acheminement commuté de datagrammes. A cet effet, un protocole de distribution d'identifiants de routes ou de labels, prédétermine des routes en établissant une correspondance entre une destination IP et un label. En fonction de son adresse de destination, chaque datagramme en entrée du réseau se voit affecter par le routeur de périphérie d'entrée (eLSR) qui est un identifiant de route ou de label. Il est ensuite acheminé dans le réseau par rapport à cet identifiant, mais plus en fonction de l'adresse de destination. Comme dans le réseau en mode connecté, l'identifiant n'a qu'une valeur locale. Le routeur de sortie supprime le routeur de label et achemine le datagramme vers sa destination. L'ensemble forme un réseau MPLS.

Le protocole MPLS est utilisé dans les réseaux IP pour permettre de réserver des ressources et de prédéterminer des chemins. Il fournit des chemins virtuels ou tunnels à travers le réseau pour connecter les nœuds se trouvant à la périphérie du domaine MPLS. Comme pour le concept d'architecture DiffServ, MPLS réduit le coût des traitements associés au relayage des paquets en les reportant à la périphérie du réseau et en réduisant la fréquence. Ainsi, il réduit le délai de transmission.

Le cœur du domaine MPLS est constitué de routeurs appelés LSR (Label Switching Router) ayant pour rôle de commuter les flux suivant la valeur du label. Les routeurs de périphérie, appelés LERs (Label Edge Router) ont pour fonction d'attribuer ou de retirer les labels. A l'entrée du réseau MPLS, les paquets IP sont classés dans des FEC (Forwarding Equivalent Classes). Des paquets appartenant à une même FEC suivront le même chemin et auront la même priorité et la même méthode de « forwarding ». L'ensemble des LSR utilisés pour une FEC constituant un chemin à travers le réseau, est appelé Label Switch Path (LSP). Il existe un LSP pour chaque FEC et les LSP sont unidirectionnels.

CHAPITRE 2 : MPLS ET INGENIERIE DE TRAFIC

2.1. Introduction :

Initialement MPLS avait pour but de commuter rapidement le trafic IP en ajoutant des labels aux paquets, pour accélérer les traitements de ce dernier dans les routeurs. Avec le développement de routeurs capables de traiter les en-têtes IP au niveau matériel, l'utilité de MPLS ne réside plus dans sa rapidité d'effectuer les traitements. En réalité, MPLS est désormais utilisé comme un outil d'ingénierie de trafic dans les réseaux dorsaux IP.

Dans la première partie de ce chapitre, nous allons voir globalement la technologie MPLS, son fonctionnement, ses applications, son interaction avec le modèle DiffServ et les perspectives qu'elle apporte. La deuxième partie traitera les objectifs de l'ingénierie de trafic et les mécanismes mis en œuvre dans l'architecture MPLS-TE (MPLS Trafic Engineering). Enfin, la dernière partie présente des attributs pouvant être affectés à un agrégat de trafic et exploités dans l'ingénierie de trafic.

2.2. Etude de MPLS: [4] [5] [6] [7]

Dans les réseaux IP traditionnels, le routage des paquets se base sur l'adresse de destination contenue dans l'entête de niveau 3. Chaque routeur doit alors accéder à l'entête réseau puis consulter sa table de routage, pour déterminer finalement le saut prochain et l'interface de sortie vers laquelle envoyer ce paquet.

Avec la croissance continue de la taille des réseaux et l'explosion des tailles des tables de routage, ce mécanisme est devenu de plus en plus consommateur en temps et en mémoire. Une méthode plus efficace pour l'acheminement des paquets est donc nécessaire à trouver préalablement. C'est dans cette méthode que s'insère la technologie MPLS. En effet, l'objectif de MPLS est de combiner en une seule entité l'efficacité des protocoles de routage et la rapidité de commutation de niveau 2, en basant la transmission des paquets sur la commutation de « labels ».

2.2.1. La technologie MPLS:

Dans cette partie, nous allons essayer de survoler les principales notions se rapportant au fonctionnement de MPLS.

2.2.1.1. Principe de base :

Initié par l'IETF en 1997, MPLS se concrétise par l'ajout d'une étiquette sur chaque paquet IP, appelée *label* ou *tag*. Cette opération identifie ainsi le chemin à suivre, appelé LSP, dans le réseau indépendamment du protocole de couche 3 utilisé ; d'où le terme «MultiProtocol ». Ces labels sont affectés par des routeurs de périphérie, appelés LER, dès l'entrée du paquet dans le réseau MPLS, et retirés à la sortie de ce même réseau par un autre LER. A l'intérieur du réseau, les routeurs LSR font la commutation de labels.

Le schéma suivant montre le rôle des différents routeurs en fonction de leur emplacement dans le réseau MPLS.

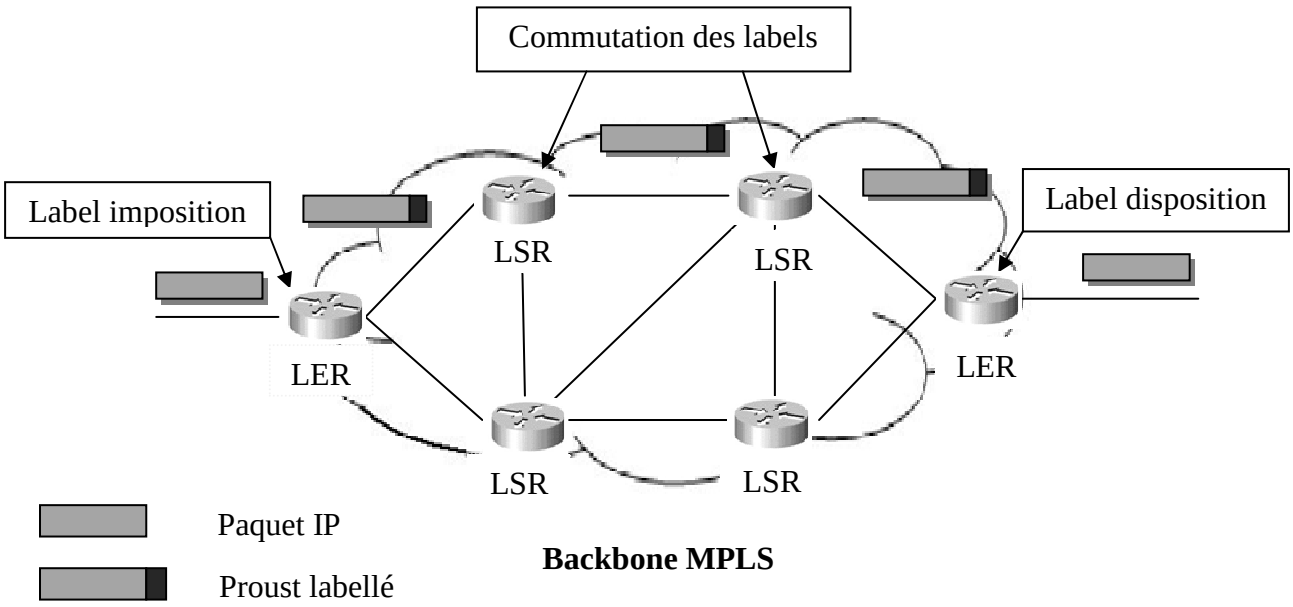

Figure 2.01 : Principe de fonctionnement de MPLS

Le routage s'effectue selon des classes d'équivalence (Forward Equivalence Class), associées à un ensemble de paquets qui ont les mêmes propriétés entre autres la classe de service et l'adresse de destination. Ainsi tous les paquets ayant une même FEC empruntent le même chemin, LSR et LER de sortie, et bénéficient du même traitement.

L'affectation d'un paquet particulier à une FEC est effectuée une seule fois par le LER d'entrée lorsque le paquet franchit le réseau. A l'intérieur du réseau, aucune reclassification n'a lieu.

2.2.1.2. Architecture MPLS :

L'architecture MPLS est basée sur deux composantes essentielles :

- *plan de contrôle* :

Il se charge de l'échange des informations de routage et des labels entre les nœuds adjacents. Une très large panoplie de protocoles de routage IGP peut être utilisée au niveau de ce plan, à savoir OSPF, IGRP, EIGRP, IS-IS, RIP et BGP. L'échange de labels nécessite la mise en œuvre de protocoles tels que TDP, LDP et MP-BGP ; utilisés pour l'échange des préfixes VPN ou encore RSVP-TE ou CR-LDP pour MPLS Traffic Engineering.

- *plan de données* :

Il transmet les paquets en utilisant les labels. Il se base sur un mécanisme simple indépendant des types de protocoles de routage et de distribution de labels mis en œuvre et utilise une table dite table LFIB remplie par le protocole de distribution de labels.

2.2.1.3. Labels :

Un label est un identifiant de petite taille fixe qui signifie locale. Il est utilisé pour faire référence à la FEC à laquelle est assigné un paquet particulier. L'affectation de la FEC peut s'effectuer complètement ou partiellement en se basant sur les informations de la couche réseau, essentiellement l'adresse de destination. De plus, elle est réalisée selon le fait que les paquets suivront le même chemin ou bénéficieront du même traitement.

Exemple : Si deux LSR A et B se mettent d'accord pour que A passe des paquets à travers B, alors il utilise le label L si et seulement si ce paquet appartient à la FEC F. Pour A, L est un «label sortant» (outgoing label) et A est dit « LSR upstream ». Tandis que pour B, L est un « label entrant » (incoming label) et B est dit « LSR downstream ». La signification de L est locale aux deux LSR A et B. Chaque LSR doit être capable d'interpréter d'une façon unique les labels entrants.

2.2.1.4. Mode trame et mode cellule :

Il existe deux modes de fonctionnement de MPLS : le mode trame (*frame mode*) et le mode cellule (*cell mode*).

2.2.1.4.1. Mode trame :

Le label est un champ de 32 bits inséré entre les entêtes de couches 2 et 3. Le format du champ « label » est le suivant.

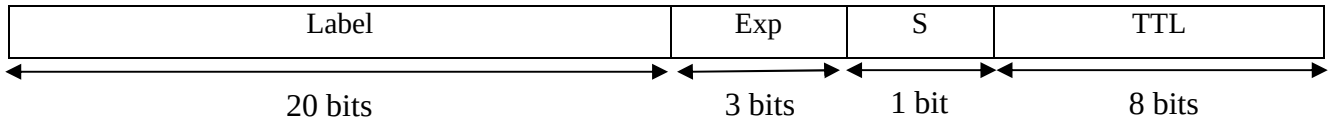

Figure 2.02 : Format du champ LABEL

Chaque label est composé des champs suivants :

- *label* : contient la valeur du label ;

- *exp* : utilisé pour définir la classe de service (CoS) ou la valeur du champ « précédence » de l'entête IP ;

- *S* : MPLS autorise l'insertion de multiples labels. Ce bit initialisé à 1, indique que ce label est le dernier dans la pile. S'il est à zéro, alors d'autres labels existent dans la pile

- *TTL* : même fonctionnement que le champ TTL dans l'entête IP.

La figure suivante illustre l'implémentation de MPLS en mode de trame sur différentes technologies :

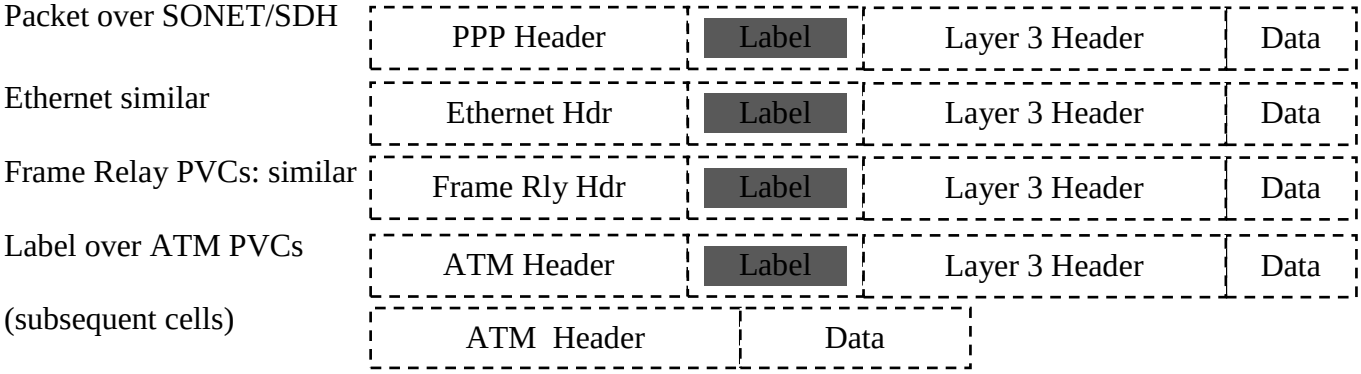

Figure 2.03 : Implémentation de MPLS en mode trame sur différentes technologies

Comme le montre la figure 2.03, dans le cas d'ATM, le label n'est ajouté qu'au niveau de la première cellule. En fait, dans le cas de la mise en œuvre des commutateurs ATM simples qui n'ont aucune connaissance de MPLS et du routage IP, des circuits virtuels sont simplement établis entre les routeurs MPLS. Les labels sont alors encapsulés entre l'entête LLC/SNAP et l'entête IP.

2.2.1.4.2. Mode cellule :

Le mode cellule constitue la deuxième solution prévue pour ATM. Il s'agit de mettre en œuvre MPLS sur des commutateurs dits « IP aware », qui possède une connaissance dans la topologie IP

grâce à un protocole de routage. Ces commutateurs sont appelés « ATM LSR ». Dans ce mode, le label est encodé dans le champ VPI/VCI de la cellule ATM.

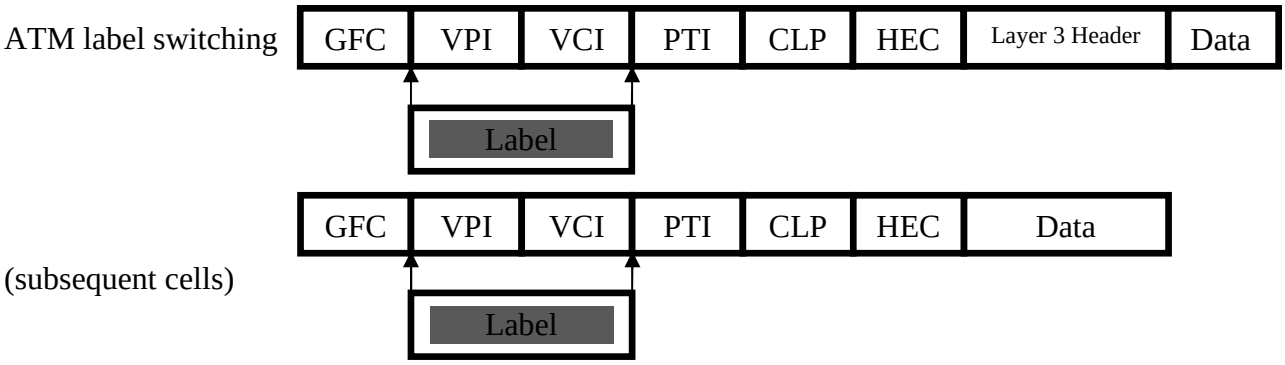

Figure 2.04 : Implémentation de MPLS en mode cellule.

2.2.1.5. LSP (Label Switched Path):

Un LSP est une séquence de LSR chargée de router les paquets labellés d'une certaine FEC. Il existe deux modes d'activation des LSP : le *control-driven LSP setup* et le *datadriven LSP setup*.

- *Control-driven LSP setup* :

La mise en place d'un LSP est dirigée par un protocole (OSPF, RSVP-TE, etc.) ou un fichier de configuration. A l'arrivée des paquets, les différents chemins sont prêts et il suffit de *mapper* le trafic par-dessus.

- *Data-driven LSP setup* :

L'établissement du LSP n'est effectué qu'au moment de l'arrivée des premiers paquets.

2.2.1.6. Pile de labels (Label stack) :

MPLS utilise généralement un seul label dans un paquet mais les applications suivantes peuvent ajouter d'autres labels :

- *MPLS Virtual Private Network* (MPLS VPN) : le premier label pointe sur le routeur périphérique de sortie et le second identifie le VPN;

- *MPLS Traffic Engineering* (MPLS-TE) : le premier label pointe sur la fin du tunnel TE et le deuxième fait référence à la destination.

Les deux applications combinées peuvent générer trois labels pour un même paquet.

2.2.1.7. Assignation de labels :

Ce paragraphe expose les différentes manières d'attribution et de contrôle d'attribution de labels.

2.2.1.7.1. Modes d'allocation de labels :

Il existe deux façons d'affecter les labels aux paquets de données :

- *allocation des labels par plateforme* :

Un label assigné à un réseau X de destination est annoncé à tous les voisins. Le label doit être localement unique et valide dans toutes les interfaces d'entrée. C'est l'opération par défaut dans le mode trame. Cette solution a l'avantage de manipuler des tables de labels de tailles faibles et de permettre un échange de label plus rapide ;

- *allocation des labels par interface* :

Les labels sont alloués par interface. Cependant, un même label peut être réutilisé avec une signification différente sur d'autres interfaces. Ce mode risque d'engendrer des tables de labels de tailles importantes.

2.2.1.7.2. Contrôle d'allocation de labels :

Il existe deux modes pour le contrôle de l'allocation des labels: *indépendant label allocation control* et *ordered label allocation control.*

- *Contrôle d'allocation indépendante* :

Il est utilisé avec le mode trame. Tous les routeurs peuvent commencer à propager les labels les uns indépendamment des autres.

- *Contrôle d'allocation ordonnée* :

Le mode cellule de MPLS exige que les LSR ATM aient reçus les labels de leurs sauts prochains (*next hop*) avant de pouvoir générer leurs propres labels locaux.

2.2.1.8. Distribution de labels :

MPLS a introduit un nouveau champ qui doit être utilisé pour les décisions d'acheminement des paquets. Bien que ces labels aient une signification locale, ils doivent être communiqués aux nœuds directement joignables. La première proposition consiste à inclure ce paramètre dans les

protocoles de routage IP existants. Seulement cette solution est rejetée vu le nombre important des protocoles à modifier.

La solution retenue consiste à créer de nouveaux protocoles pour l'échange de labels. Et c'est dans ce cadre que les protocoles LDP, TDP, RSVP-TE (modification du protocole RSVP), CR-LDP ou MP-BGP étaient créés.

Dans l'architecture MPLS, la décision de lier un label L à une FEC particulière est prise par le *LSR downstream*. Celui-ci informe alors le *LSR upstream* de cette correspondance.

La méthode de distribution des labels est dite « downstream », car les liens *label/FEC* sont distribués dans la direction *downstream* vers *upstream*. Il en existe deux variantes :

- *distribution* « *Unsollicited downstream* » :

Tous les routeurs peuvent générer les labels locaux et les faire propager aux routeurs adjacents d'une façon asynchrone et indépendante du fait que les routeurs soient des LSR *downstream* ou *upstream* pour la destination. Ce type de distribution est utilisé par le mode trame de MPLS ;

LIB on B :

	LSR	Label
X	local	25

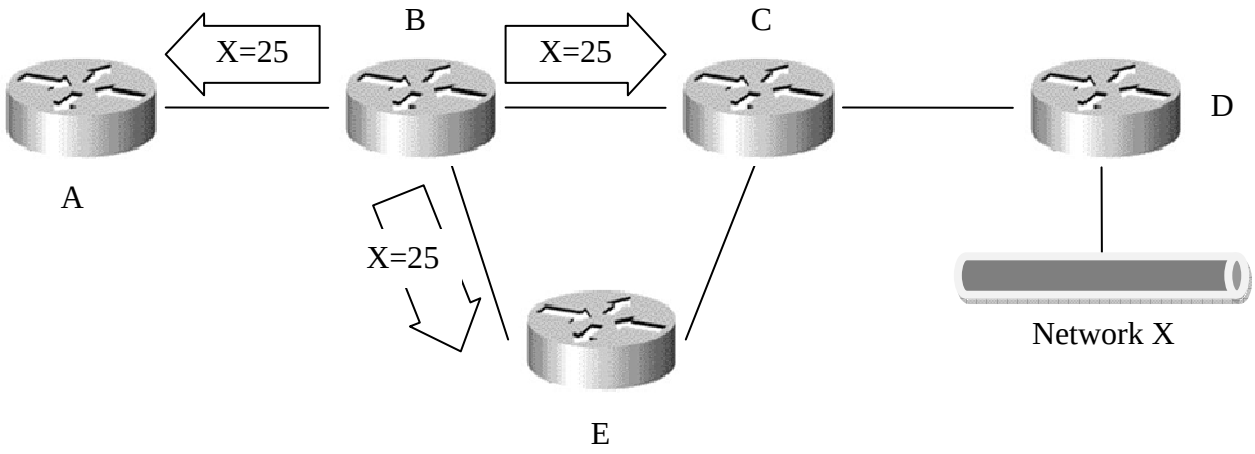

Figure 2.05 : Unsolicited downstream

- *distribution «Downstream-on-demand »*:

Elle est utilisée avec le mode cellule de MPLS. Un LSR n'affecte de label que si une demande lui a été adressée par un LSR *upstream*

Routing Table of A		Routing Table of C		Routing Table of D		Routing Table of E	
Network	Next hop	Network	Next hop	Network	Next hop	Network	Next hop
X	C	X	D	X	E	X	conn

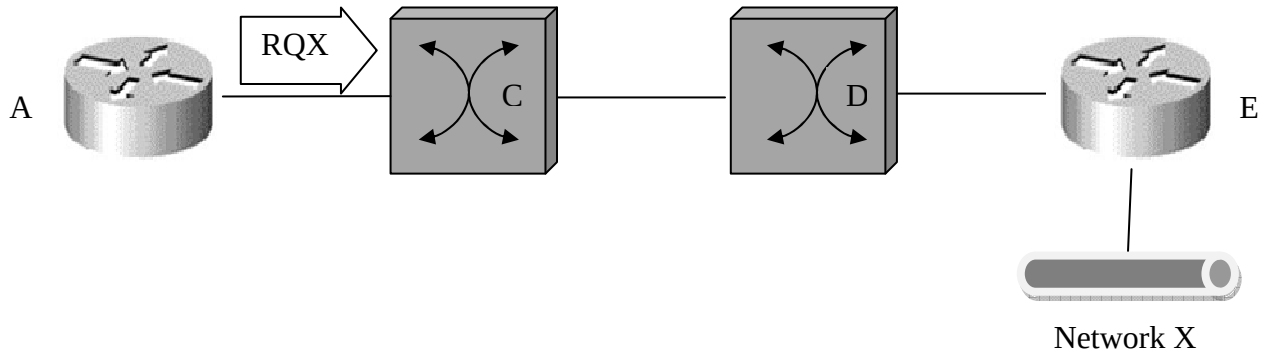

Figure 2.06 : downstream-on-demand

Des implémentations MPLS fournissent seulement la distribution de labels *downstream on demand*, certains fournissent la distribution de labels unsolicited downstream, et d'autres peuvent combiner les deux.

2.2.1.9. Tables MPLS :

A partir des informations apprises par le protocole de distribution des labels, les LSR construisent deux tables, la LIB et la LFIB. De manière générale, la LIB contient tous les labels appris des LSR voisins. Tandis que la LFIB, utilisée pour la commutation proprement dite des paquets, est un sous-ensemble de la LIB. MPLS utilise aussi une table dite FIB qui contient toutes les informations contenues dans la table de routage.

2.2.1.9.1. Table FIB (Forwarding IP Base):

C'est une table de commutation IP complète qui contient les mêmes informations que la table de routage et qui permet au routeur de prendre ses décisions d'acheminement. La génération d'entrées est déclenchée par la production de changements au niveau de la table de routage. Si pour une destination donnée, il n'existe pas d'entrée correspondante, alors le paquet est supprimé.

2.2.1.9.2. Table LIB (Label Information Base) :

C'est la première table construite au niveau d'un LSR, elle contient pour chaque sous-réseau IP la liste des labels reçus par les voisins. Ses entrées sont de la forme réseau de destination, LSR, label ; où LSR est le nœud qui a généré le label.

2.2.1.9.3. Table LFIB (Label Forwarding Information Base):

Elle est construite à partir de la table LIB et la table FIB ; et est utilisée pour la commutation de labels.

Pour chaque réseau IP appris par l'IGP, un prochain saut (next-hop) pour atteindre ce réseau est déterminé. Le LSR choisit ainsi l'entrée de la table LIB qui correspond au réseau IP et sélectionne comme label de sortie, le label annoncé par le voisin déterminé par l'IGP. Ses entrées sont de la forme : label, action, prochain saut ; le champ « action » contiendra soit la valeur du nouveau label qui remplacera l'ancien, soit le mot «pop» qui veut dire que le LSR doit se contenter d'enlever le label avec quoi il reçoit le paquet.

2.2.1.10. Penultimate Hop Popping :

A la réception d'un paquet labellé, le routeur périphérique de sortie aura à effectuer deux consultations :

- *table LFIB* : pour enlever le label;
- *table FIB* : pour faire router les paquets en se basant sur l'adresse IP du prochain saut.

Le *Penultimate Hop Popping* (PHP) constitue une solution qui permet d'optimiser le nombre de consultations au niveau du routeur périphérique de sortie. Il s'agit d'enlever le label au niveau de la sortie du routeur qui précède le dernier nœud du domaine MPLS. En fait, le routeur périphérique de sortie annonce ces réseaux IP avec le label « implicit-null » à ses voisins. Un LSR ayant

comme label de sortie « implicit-null » doit faire suivre le paquet sur l'interface de sortie spécifiée. Le routeur périphérique de sortie n'aura alors plus besoin de consulter la table LFIB.

2.2.1.11. Rétention de labels :

MPLS considère deux modes de rétention de labels : mode libéral (libéral mode) et le mode conservatif (conservatif mode).

- *Mode de rétention libéral* :

Chaque LSR stocke le label reçu dans sa LIB même si ce label n'a pas été reçu par le LSR du prochain saut (*next hop*). Ce mode accélère la convergence dans le cas de défaillance du chemin initial.

- *Mode de rétention conservative* :

Un LSR ne stocke que les labels reçus des sauts prochains et ignore les autres labels.

2.2.2. Applications MPLS :

Au-delà des préoccupations premières ayant provoqué sa genèse, MPLS peut être utilisé pour différentes applications.

2.2.2.1. Le routage IP unicast :

C'est l'application la plus commune de MPLS. Deux mécanismes sont mis en œuvre au niveau du plan contrôle: le protocole de routage IP et le protocole de distribution de labels. Le protocole de routage se charge de l'accessibilité aux différents réseaux. Tandis que le protocole de distribution de labels fait correspondre les labels aux réseaux fournis par le protocole de routage.

Pour cette application, la FEC correspond à un réseau de destination se trouvant dans la table de routage.

2.2.2.2. Le routage IP multicast :

Il est traité séparément parce qu'il exige un routage différent. Le protocole PIM avec des extensions pour MPLS, est utilisé pour propager les informations de routage ainsi que les labels. La FEC correspond ici à une adresse de destination multicast stockée dans la table de routage.

2.2.2.3. MPLS Traffic Engineering :

Il s'agit de rediriger, par l'intermédiaire de tunnels LSP, le trafic sur des liaisons peu ou pas utilisées afin de décharger certaines liaisons saturées, fortement sollicitées par l'IGP. Mais aussi pour répartir le trafic sur d'autres liaisons moins chargées qui ne sont pas sélectionnées par l'IGP.

MPLS-TE requiert OSPF ou IS-IS avec extensions au MPLS-TE comme IGP. Ces deux protocoles détiennent la totalité de la topologie dans leurs bases de données et doivent avoir des informations additionnelles sur les ressources et les contraintes du réseau.

Pour la propagation des labels et l'établissement des tunnels, MPLS-TE utilise soit le protocole RSVP-TE ou CR-LDP.

2.2.2.4. QoS :

La qualité de service différenciée est une extension au routage IP unicast. Elle se base sur le champ EXP de l'entête en définissant un traitement pour chaque valeur de ce champ ou sur la création de tunnels séparés pour les différentes classes.

La FEC correspond à une combinaison d'un réseau de destination et d'une classe de service.

2.2.2.5. MPLS VPN :

Les VPN sont implémentés en utilisant un label additionnel pour déterminer le VPN et le réseau VPN de destination correspondant. MP-BGP est utilisé pour faire propager les informations de routage VPN et les labels à travers le domaine MPLS. La FEC correspond à un réseau VPN de destination.

Comme le montre la figure suivante, ces applications diffèrent uniquement au niveau du plan de contrôle mais le fonctionnement du plan de données reste le même.

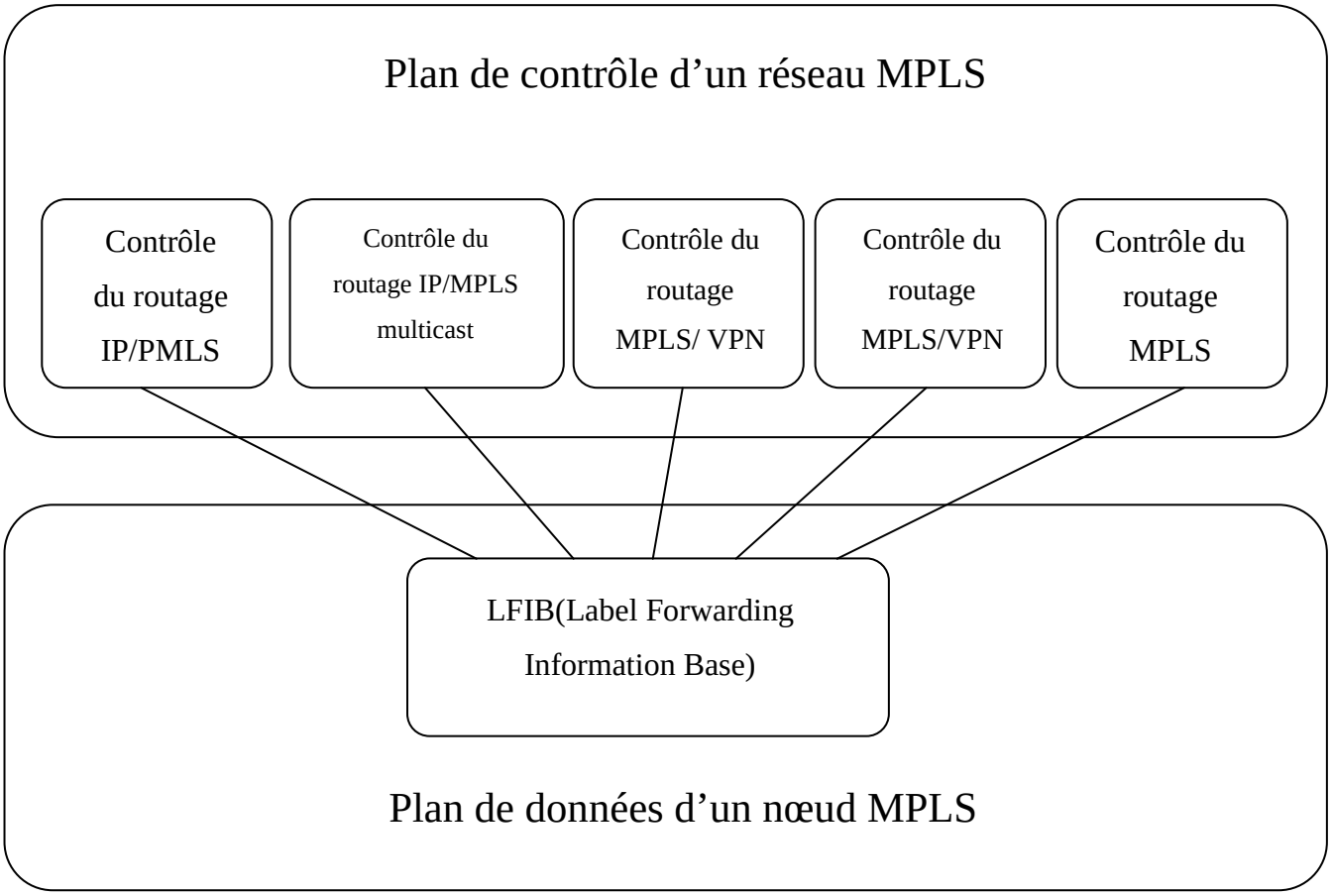

Figure 2.07 : Interactions entre les différentes applications MPLS

2.2.3. Intégration MPLS/DiffServ :

MPLS simplifie l'administration du cœur du réseau en ajoutant de nouvelles fonctionnalités particulièrement intéressantes pour la gestion de la qualité de service. Dans le même esprit que l'architecture MPLS, DiffServ procède à une agrégation des demandes de QoS par l'utilisation de valeurs discrètes de DSCP pour des flux possédants des exigences voisines de QoS. Pour cela, deux approches sont possibles.

2.2.3.1. Approche E-LSP :

L'information de classification est mappée dans le champ EXP de MPLS. Ce champ autorise 8 marquages différents contre 64 pour le DSCP. L'ordonnancement des paquets (PHB) est défini au niveau de chaque nœud dans le nuage MPLS en se basant sur la valeur du champ EXP.

2.2.3.2. Approche L-LSP :

Le label associé à chaque paquet MPLS contient la partie du marquage DiffServ qui spécifie la manière avec laquelle les paquets seront servis dans les files d'attente.

La partie « Drop Precedence » du marquage est contenue dans le champ EXP.

2.2.4. GMPLS : Generalized MultiProtocol Label Switching :

Le Generalized MultiProtocol Label Switching fait référence aux extensions portées à MPLS pour les différents types de réseaux. Il a pour objet de définir un plan de contrôle commun aux différentes techniques de transport. En effet, GMPLS ne se limite pas à la commutation de paquets ou cellules (cas de MPLS) mais s'étend aussi aux circuits où le label constitue un slot dans le multiplexage temporel, ou encore aux fibres optiques où les labels représentent des longueurs d'onde.

Cette brève introduction à MPLS offre une vision globale du fonctionnement de cette technologie utilisant des mécanismes de commutation de labels destinés à l'origine à réduire les coûts de routage de couche réseau.

Toutefois, ce sont surtout les retombées connexes de MPLS telles que la QoS, les VPN ou encore l'ingénierie de trafic qui ont fait que cette technologie soit bien appréciée par la majorité des fournisseurs d'accès à internet ainsi que par certaines grandes entreprises.

2.3. Etude de l'ingénierie de trafic avec MPLS : [8] [9] [10] [11]

2.3.1. Objectifs de l'ingénierie de trafic :

L'ingénierie de trafic (Traffic Engineering) correspond à l'assignation des flux de trafics sur une topologie physique selon différents critères. Les procédures de contrôle offertes par les protocoles de routage mis en œuvre pour internet ne sont plus adéquates pour les nouveaux réseaux émergents.

En effet, les IGP basés sur le plus court chemin, par exemple, contribuent significativement aux problèmes de congestion dans les systèmes autonomes dans internet.

Les algorithmes SPF optimisent généralement les chemins en se basant sur la topologie et non sur la disponibilité de ressources ou sur les caractéristiques de trafic.

En conséquence, des problèmes de congestion apparaissent fréquemment lorsque :

- les chemins les plus courts pour de multiples flux de trafics convergent vers des liens ou interfaces de routeurs spécifiques ;

- un flux particulier de trafic est routé à travers un lien dont la bande passante est insuffisante.

L'ingénierie de trafic se penche sur ce problème en essayant d'optimiser l'utilisation des ressources du réseau par l'intermédiaire de tunnels LSP établis à travers le backbone MPLS. Ceci en vue de partager la charge entre les liens (load balancing), de rediriger le trafic vers des liaisons peu ou pas utilisées ou encore de prévoir un contrôle précis du re-routage de trafic en cas d'incident sur le chemin primaire.

Le *traffic engineering* améliore ainsi la QoS des flux des trafics par la minimisation des pertes de paquets et des délais de transmission, la maximisation des débits utiles (*throughput*) et l'introduction de la notion de priorité. En outre, il permet aux opérateurs de mieux répondre aux attentes de leurs clients en leur offrant plus d'options de raccordement.

La notion de qualité de service recouvre un ensemble de techniques destinées à offrir de bout en bout aux applications le service dont elles ont besoin. Elle couvre généralement deux aspects : l'aspect temporel (délai d'acheminement) et l'aspect sémantique (perte de données), qui s'expriment selon quatre paramètres : bande passante (débit), délai de transfert, variation de ce délai (gigue) et enfin la fiabilité.

La qualité de service constitue un axe de recherche majeur dans les réseaux. Deux approches ont déjà été étudiées : la réservation de ressource (réseau à états) et la priorisation des flux (réseaux sans états) [10], [11].

2.3.2. Architecture MPLS-TE :

La mise en œuvre des solutions de l'ingénierie de trafic sur une architecture MPLS peut être réalisée grâce à la souplesse du mode de création et de définition des LSP.

La figure 2.08 qui suit présente l'architecture MPLS-TE.

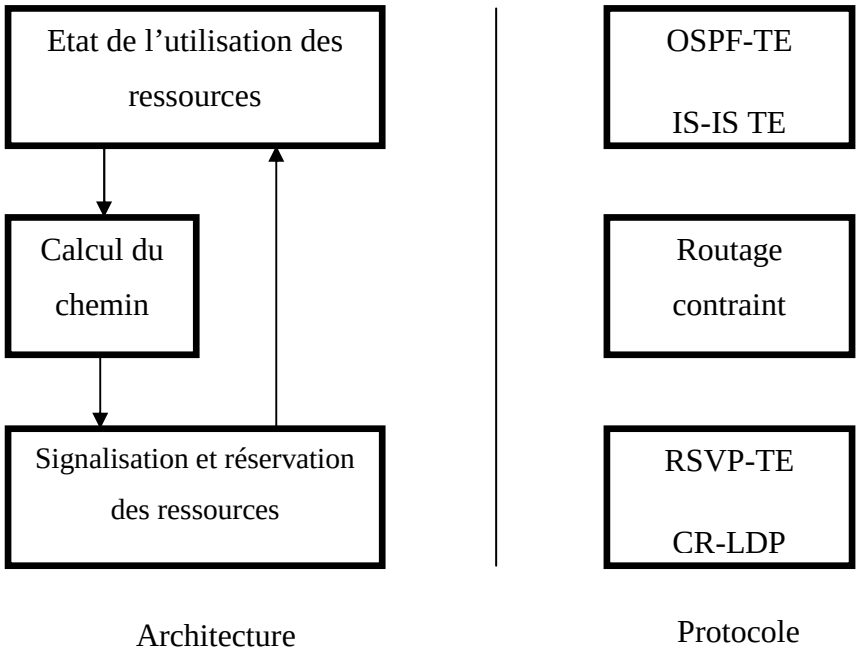

Figure 2.08 : Architecture MPLS-TE

2.3.2.1. Etat de l'utilisation des ressources :

La conception des tables de routage fait que tout le trafic allant d'un nœud du réseau à un autre doit suivre le même chemin. Elle ne distribue pas le trafic avec une granularité plus fine. Le routage explicité avec MPLS dévie le trafic des chemins les plus courts calculés par l'algorithme Dijkstra vers des chemins plus longs moins chargés.

Pour pouvoir le faire, les nœuds doivent avoir une connaissance de la capacité et de la charge des différents liens du réseau. Ces informations seront distribuées en utilisant les protocoles de routage à états de liens.

Les seuls protocoles à état de liens pouvant supporter l'ingénierie de trafic sont OSPF et IS-IS auxquels sont rajoutées quelques extensions pour l'ingénierie de trafic.

2.3.2.2. Calcul du chemin :

Afin de pouvoir gérer les critères de bande passante et de priorité introduits par l'ingénierie de trafic, des modifications ont été apportées au niveau de l'algorithme de calcul du plus court chemin (*Shortest Path First*) sur lequel se base le choix des routes dans OSPF et IS-IS. Le nouvel

algorithme a été baptisé PCALC et permet le routage basé sur les contraintes *Constraint Based Routing.*

L'algorithme PCALC est défini comme suit :

- ignorer les liens qui offrent une bande passante inférieure à celle requise ;
- ignorer les liens qui ne correspondent pas à l'affinité signalée ;
- exécuter l'algorithme Dijkstra sur la topologie restante en tenant compte des métriques de l'IGP (OSPF ou IS-IS) ;
- si plusieurs chemins subsistent, choisir celui avec le minimum de bande passante totale disponible sur tous les liens constituant le chemin ;
- s'il y a encore plusieurs chemins, sélectionner celui avec le minimum de sauts ;
- s'il existe toujours plusieurs routes, en choisir une aléatoirement.

2.3.2.3. Signalisation et réservation des ressources :

Le choix du chemin étant effectué, on a besoin de procédures qui puissent établir les tunnels en faisant propager les labels et en réservant les ressources. A ces fins, MPLS-TE déploye soit une extension du protocole RSVP, nommé RSVP-TE pour distribuer les labels, soit le protocole CR-LDP.

2.3.3. Attributs TE :

Un tunnel TE est généralement caractérisé par :

- ses deux extrémités (LER d'entrée et de sortie) ;
- un ensemble d'attributs qui détermine ses caractéristiques ;

2.3.3.1. Notion d'affinité :

L'affinité est simplement une valeur de 32 bits spécifiée sur les interfaces des routeurs MPLS et qui répertorie les ressources, généralement sur la base de la bande passante de l'interface dans des « classes de ressources ». Cet attribut, affecté à un flux de trafic, indique si les membres d'une certaine classe de ressources doivent être inclus ou exclus du chemin à travers lequel circulera le trafic. Ainsi, avant de déclencher le mécanisme de calcul de routes, toutes les classes de ressources dont l'affinité ne répond pas à l'affinité attribuée au flux de trafic ne seront pas tenues en considération.

2.3.3.2. Notion d'adaptation :

Cet attribut fait partie des paramètres de maintenance de routes. Associé à un flux de trafic, il indique si ce flux est sujet à une optimisation. En effet, l'ingénierie de trafic offre la possibilité d'optimiser les chemins utilisés par les tunnels LSP. Pour éviter la rupture du routage, le premier tunnel reste fonctionnel jusqu'à ce que le nouveau tunnel optimal soit totalement établi.

2.3.3.3. Notion de priorité :

Ce paramètre définit l'importance relative des trafics les uns par rapport aux autres et constitue un attribut important dans les environnements dont le routage est basé sur les contraintes (*constraint-based routing*) ou les environnements qui permettent la préemption.

2.3.3.4. Notion de préemption :

La préemption est utilisée pour s'assurer qu'un flux de trafic avec une priorité élevée soit toujours routé à travers les chemins relativement favorables dans un environnement de différenciation de services. Le réseau peut préempter un trafic A au profit d'un trafic B si B a une priorité supérieure à celle de A, si la ressource sollicitée ne peut pas accommoder A et B et si la préemption est activée.

2.3.3.5. Maximum Allocation Multiplier (MAM) :

Il s'agit d'un attribut configurable administrativement, qui détermine la proportion de ressources pouvant être allouée au trafic. Il est généralement applicable à la bande passante du lien, toutefois il peut également être appliqué aux buffers dans les LSR. La valeur du MAM peut être choisie de façon à ce que la ressource soit sur ou sous utilisée.

2.4. Conclusion :

Dans ce chapitre, nous nous sommes consacrés, dans une première partie, à la présentation de la technologie MPLS, son fonctionnement, les fonctionnalités qu'elle offre et son interaction avec le modèle *DiffServ* dans le but d'offrir une QoS différenciée.

Nous avons défini, dans une deuxième partie, l'ingénierie de trafic ainsi que ses objectifs. Nous avons également présenté la nouvelle architecture de l'ingénierie de trafic avec MPLS. Dans ce contexte, nous avons détaillé les modules de routage contraint et de signalisation. Nous avons conclu avec les attributs affectés aux agrégats de flux et qui peuvent être utilisés pour l'ingénierie de trafic. Dans le chapitre suivant, nous présenterons les étapes de la réalisation de la plateforme

IP/MPLS, l'implémentation de l'ingénierie de trafic et l'évaluation de cette solution en termes de qualité de service.

Dans l’architecture *DiffServ*, le traitement différencié despaquets s’appuie sur trois opérations fondamentales :

- la classification des flux en classes de services ;
- l’introduction de prioritéss au sein des classes (Scheduling) ;
- la gestion du trafic dans une classe donnée.

Ainsi il devient question de supporter un schéma de classification en attribuant des priorités à des agrégats de trafic.

CHAPITRE 3 : DIMENSIONNEMENT DES ARTERES DU BACKBONE MPLS.

3.1. Introduction :

L'amélioration des performances des réseaux se voit comme un défi d'optimisation composé de deux parties : *la gestion de capacité* et *la gestion du trafic*. Logiquement ces deux actions opèrent à différentes portées et à différentes échelles de temps. La gestion de la capacité consiste à faire sa planification, à contrôler le routage et à gérer les ressources comme la capacité des liens et la capacité des buffers. De plus, la planification de la capacité implique la planification des ressources nécessaires dans le réseau à savoir la capacité des liens. Tandis que, la gestion du trafic consiste à conditionner le trafic et à gérer les files d'attente.

Quand le réseau est construit pour la première fois, il tient en compte certaines données physiques et commerciales ainsi que des besoins, à court et à moyen terme, des utilisateurs. Le processus de planification réserve alors des ressources pour chaque type de service en tenant compte du trafic généré.

Ce troisième chapitre détermine la capacité des liens dans un réseau MPLS, tout en sachant que ce réseau supporte un trafic temps réel comme la voix et un trafic élastique DATA. Dans ce contexte, le dimensionnement d'un réseau offrant différentes classes de services s'avère difficile. Initialement, l'étude d'un réseau se fait en suivant des procédures successives qui peuvent être révisées par la suite. Il y a toujours certains éléments auxquels il faut accorder plus d'importance dans la conception d'un réseau :

- *la planification du trafic*: cette procédure consiste à déterminer le nombre de clients auxquels le service est destiné ainsi qu'aux différents types de services offerts par le réseau ;

- *le choix de la topologie du réseau* : l'emplacement des nœuds et leur classification. A l'exemple du choix du nombre de nœuds d'entrée et du nombre de nœuds du cœur dans un réseau dorsal MPLS ;

- *le calcul de la capacité des liens.*

Prenons particulièrement les hypothèses initiales suivantes : 5 sites : SI, S2, S3, S4 et S5.

Chaque site est formé d'un certain nombre d'utilisateurs respectivement N1, N2, N3, N4, N5 (figure 3.01).

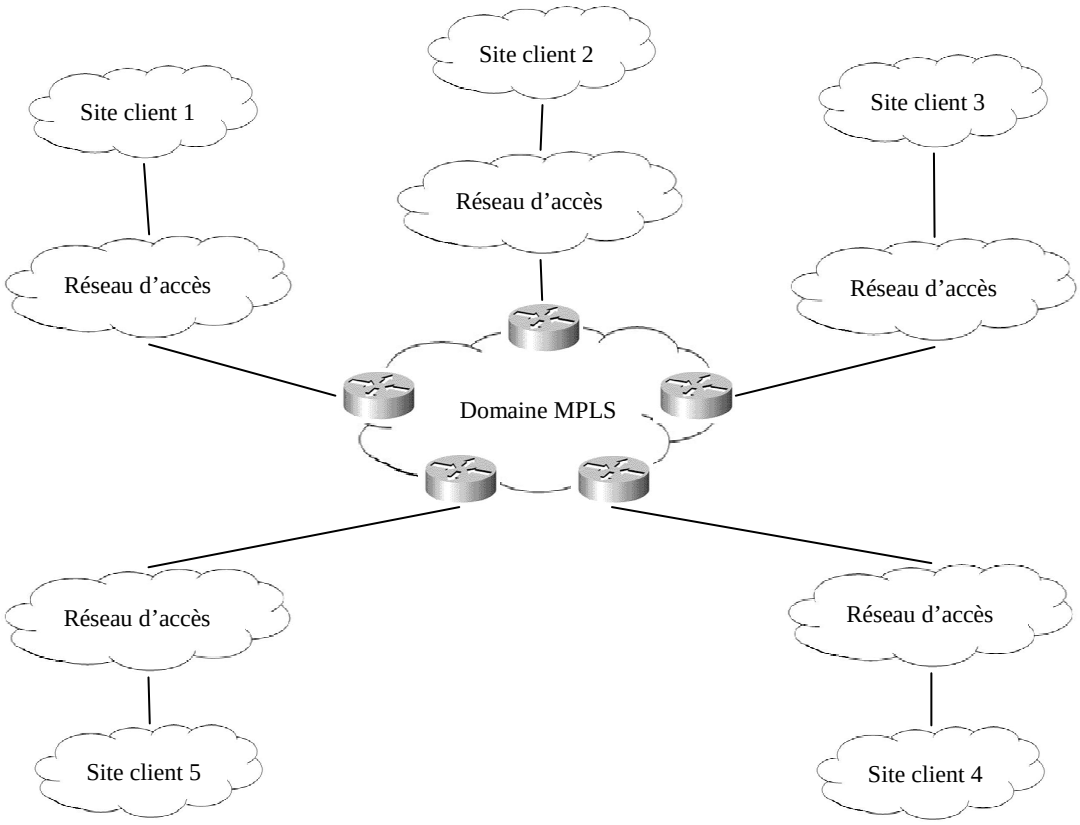

Figure 3.01: Topologie choisie pour un service de voix

3.2. Dimensionnement du réseau d'accès : [12] [13]

3.2.1. Modèle de trafic :

Considérons que la loi du trafic voix obéit à un processus de Poisson. Alors la probabilité d'observer k arrivées pendant un intervalle de temps t est donnée par :

$$P_k(t) = \frac{(\lambda t)^k}{k!} e^{-\lambda t} \qquad (3.01)$$

λ *:* étant le taux d'arrivée.

La propriété la plus importante du processus de Poisson, peut s'interpréter de la façon suivante: une arrivée poissonienne signifie que la probabilité pour qu'un client arrive pendant dt est à peu près égale λdt.

La deuxième propriété essentielle du processus de Poisson relie le processus d'arrivée poissonien aux variables aléatoires, mesurant ainsi le temps d'inter arrivée (exponentielles). Cette propriété s'interprète de la façon suivante : des arrivées poissoniennes signifient que les inters arrivés sont de types exponentiels. Les lois d'Erlang qui sont utilisées pour le dimensionnement des réseaux offrant le service de la voix, se basent sur les hypothèses de services exponentiels et des arrivées de type poissonien.

3.2.2. Lois d'Erlang :

Le choix des lois d'Erlang dans plusieurs projets se justifie par le fait qu'elles donnent généralement des résultats raisonnables. Les formules sont utilisées dans les conditions suivantes :

- le nombre de clients est supérieur au nombre de ressources disponibles pour les servir ;

- les demandes des clients sont indépendantes les unes des autres.

Dans les réseaux téléphoniques classiques, la loi la plus utilisée est celle d'Erlang B. Pour la voix sur IP, les opérateurs conservent toujours cette formule dans leur procédure de dimensionnement du réseau d'accès. La loi d'Erlang B calcule la probabilité de rejet d'une demande de ressource à raison de ressources non disponibles. La probabilité de blocage sera alors égale à :

$$P_n = \frac{\frac{A^n}{n!}}{\sum_0^n \frac{A^i}{i!}} = E_1(A, n) \qquad (3.02)$$

Où :

- n : le nombre de ressources disponibles,
- A : le trafic offert,
- P_n: la probabilité que les n ressources soient occupés,
- E_1 : la première formule d'Erlang qui est fonction de A et n.

3.2.3. Débit d'accès :

Pour chaque site, le débit d'accès se calcule en suivant les étapes suivantes :

- calcul du débit par appel ;

- calcul du nombre de circuits.

3.2.3.1. Débit par appel :

Le débit d'accès peut être calculé en tenant compte des éléments suivant :

- les codecs audio utilisés au niveau de la couche application ;
- les différentes encapsulations aux niveaux des différentes couches (transport, réseau) ;
- les protocoles au niveau de la couche liaison.

3.2.3.1.1. Les codecs audio :

Les codecs les plus utilisés pour la compression/décompression de la voix sur IP sont :

- G.711 offrant un débit de 64 Kbit/s ;
- G.723 offrant un débit de 6.3 et 5.3 Kbit/s ;
- G.729 offrant un débit de 8 Kbit/s.

Selon le débit généré par le codec et tenant compte des différentes possibilités des cycles de transmission, nous obtenons la taille des données audio (*data voice).* Ces données audio passeront à des encapsulations au niveau des différentes couches ; commençant par la couche transport jusqu'à la couche liaison de données.

3.2.3.1.2. Les encapsulations au niveau transport et réseau :

Les données audio de la couche application sont affectées au niveau de la couche transport, d'un entête RTP ayant une taille minimale de 12 octets, puis d'un entête UDP avec 8 octets enfin la mise en paquet au niveau de la couche réseau ajoute 20 octets pour l'entête IP. La figure suivante illustre le principe de la mise en paquet.

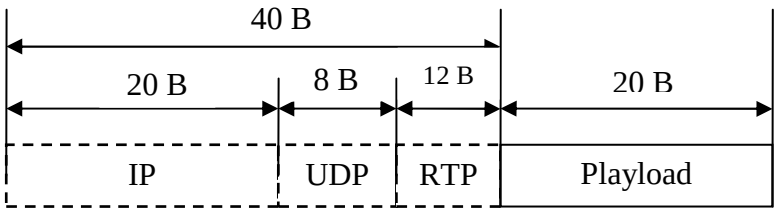

Figure 3.02: Encapsulation RTP/UDP/IP

Les 20 octets du protocole IP considérés ne tiennent pas compte des champs *Options* et *Padding.*

3.2.3.1.3. Les protocoles utilisés au niveau de la couche liaison :
L'encapsulation doit tenir compte des différents protocoles au niveau de la couche liaison.

- *Ethernet :*

La technologie Ethernet est la technologie la plus répandue dans les réseaux d'entreprises (LAN).

- *High level Data Link Control (HDLC):*

Ce protocole permet une liaison point à point ou point multipoint et une transmission synchrone. Il se base également sur le principe de fenêtre coulissante. La taille des données dans la trame HDLC est variable.

- *Frame Relay (FR) :*

Ce protocole est utilisé dans les réseaux maillés. Il assure l'établissement de circuit virtuel entre deux usagers.

- *Point to Point Protocol (PPP):*

Ce protocole est utilisé pour transporter des paquets entre deux usagers à travers des liens simples. Ces liens fournissent des transmissions simultanées *full-duplex*.

- *Asynchronous Transmission Mode (ATM):*

Ce protocole se base sur la transmission de cellules à l'intérieur d'un circuit virtuel. Son principal intérêt est de permettre la réservation de ressources pour un Circuit Virtuel.

La signification des différents champs n'est pas aussi importante que la taille des données qu'ajoute chaque protocole. Le débit généré sur le support physique change suivant la variation des différents paramètres cités ci-dessus.

Pour chaque site, nous supposons qu'il y a un choix uniforme entre les différents utilisateurs des différents paramètres :

- codec,
- cycle de transmission,
- protocole de couche liaison.

La formule de calcul du débit par appel est la suivante :

$$Débit_{apl} = [(Débit_{codec} \times cycle_{trans}) + entête_{RTP.UDP.IP} + entête_{protocoleliaison} + enqueue_{protocoleliaison}] \times 8/cycle_{trans} \quad (4.03)$$

- $Débit_{apl}$: Débit par appel en Kbit/s ;
- $Débit_{codec}$: Débit généré par le codec en Kbit/s ;
- $cycle_{trans}$: Le cycle de transmission de paquet en ms ;
- $entête_{RTP.UDP.IP}$: La taille de l'entête RTP/UDP/IP à ajouter en octets ;
- $entête_{protocoleliaison}$: La taille de l'entête du protocole de couche liaison en octets ;
- $enqueue_{protocoleliaison}$: La taille de l'enqueue du protocole de couche liaison en octets.

3.2.3.2. Calcul du nombre de circuits :

L'algorithme de la formule d'Erlang inverse permet de déterminer le nombre de circuits à mettre en œuvre pour supporter un trafic donné avec une probabilité de blocage fixe.

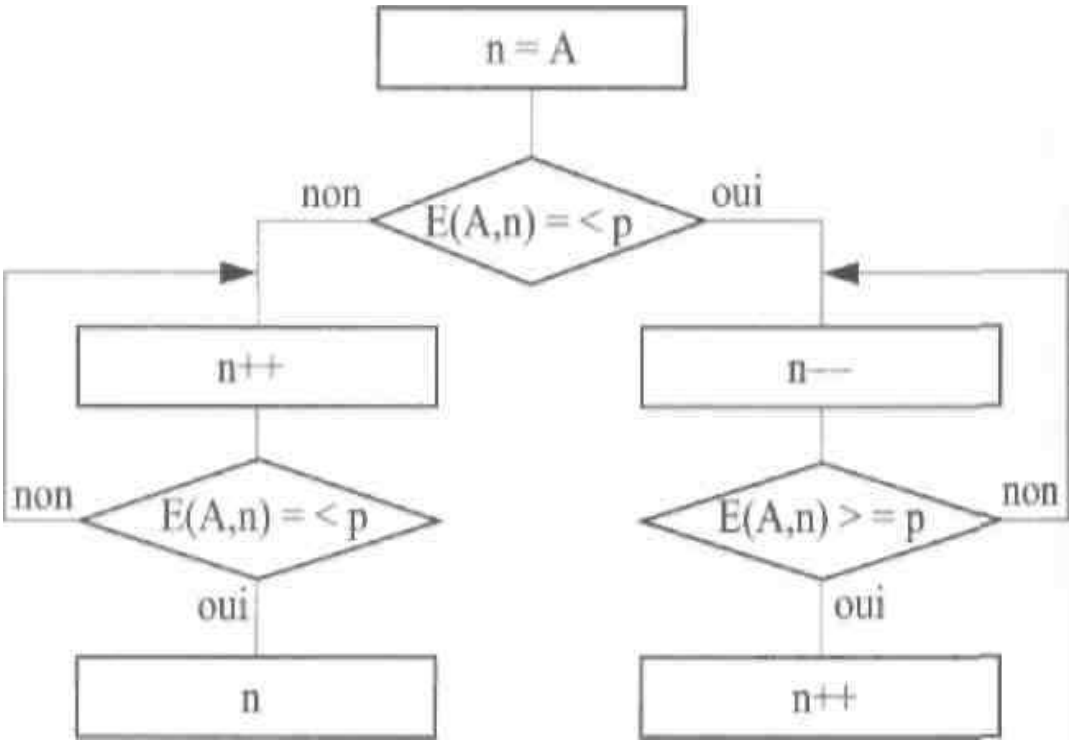

Figure 3.03: L'algorithme d'Erlang inverse

- A : trafic offert,
- n : nombre de circuit,
- E : formule d'Erlang,
- p : probabilité de blocage fixé par l'opérateur.

La bande passante nécessaire peut se calculer à partir du nombre de circuits et le débit par appel :

$$bande = D\acute{e}bit_{appel} \times nombre_{circuit} \quad (3.04)$$

La même méthodologie est appliquée pour les autres sites.

3.3. Le réseau dorsal IP/MPLS : [14] [15] [16] [17]

Cette partie offre les détails de calcul des capacités pour les différentes artères.

3.3.1. Choix de la topologie :

La topologie à mettre en œuvre est présentée dans la figure 3.04.

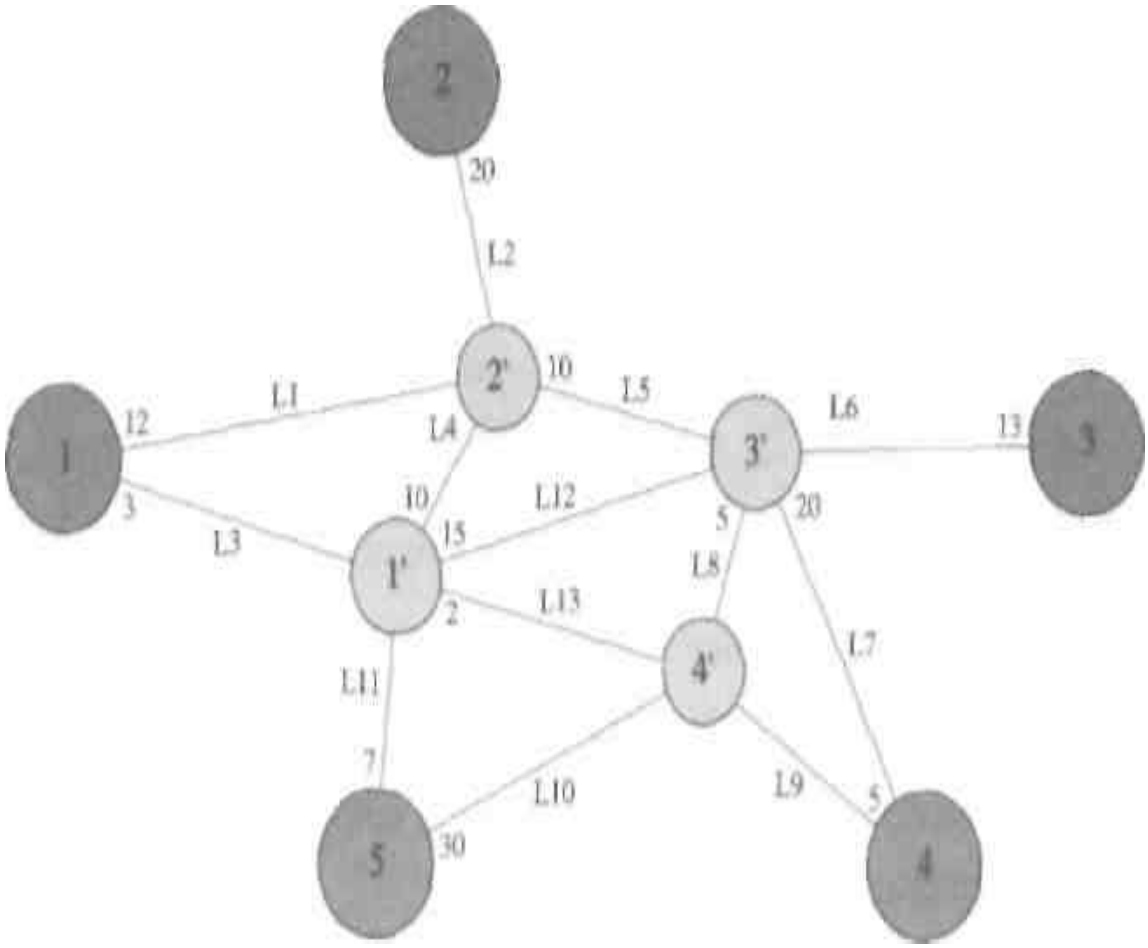

Figure 3.04: Topologie choisie du *backbone* MPLS

Le réseau dorsal MPLS est formé par cinq routeurs à la périphérie LER et quatre routeurs au cœur du réseau LSR (figure 3.04). Pour des raisons de simplification, nous supposons que les Edge Router n'effectuent pas d'opération de commutation des paquets. En plus, les flux entrants au domaine MPLS par un Ingress LSR ayant pour destination un Egress LSR qui lui est adjacent, doivent passer obligatoirement par un LSR appartenant au cœur du *backbone* MPLS. Les numéros en noir représentent les métriques des liens associés aux distances. Les L_i représentent les numéros des liens et ceci pour tout i de 1 à 13.

3.3.2. Le débit à la sortie du routeur d'accès LER (Label Edge Router) :

Supposons que chaque site est lié à un *edge router* et que le débit d'accès sera le trafic à l'entrée de chaque LER. MPLS est un protocole qui fonctionne entre la couche réseau et la couche liaison. Le format de l'entête MPLS est présenté dans la figure 3.05.

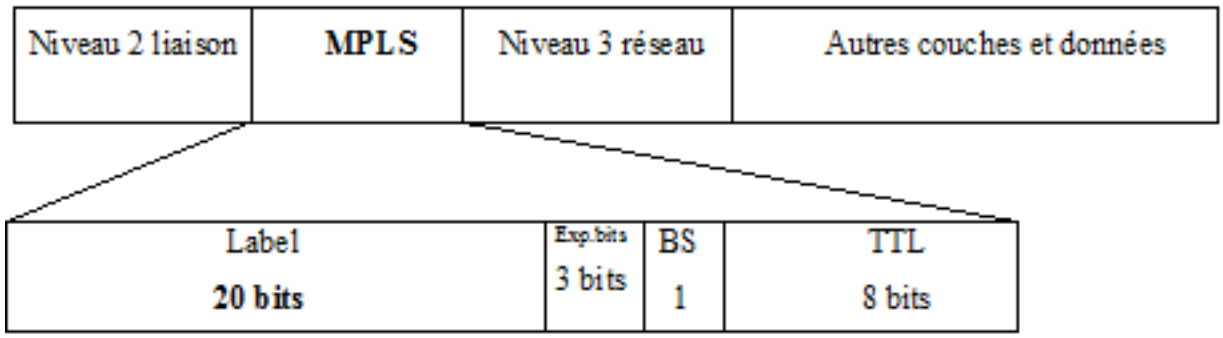

Figure 3.05: Format de l'entête MPLS

- *« Label » codé sur 20 bits* : référence utilisé pour router un paquet ;

- *Bits expérimentaux EXP sur 3 bits* : pour identifier différentes classes de trafics pour supporter les modèles de QoS de DiffServ ;

- *BS* : *Bottom Stacking* sur 1 bit : un *bottom stock* est mis à 1 pour indiquer la dernière pile entrée ;

- TTL : *Time To Live* sur 8 bits : il a la même signification qu'en IP.

4.3.2.1. Débit pour le cas de l'ATM et du Frame Relay :

Au dessus d'ATM, le label est inséré dans les champs VPI/VCI.

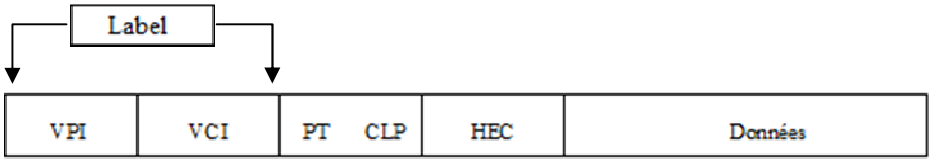

Figure 3.06: Encapsulation du paquet labellisé sur la couche ATM

Il en est de même pour le protocole FR puisque le label sera inséré dans le champ DLCI.

3.3.2.2. Débit pour le cas de l'Ethernet, PPP et HDLC :

Dans ce cas, le débit à la sortie du LER sera différent de celui à l'entrée puisque l'ajout d'un entête MPLS à quatre octets influera sur le débit. La procédure de calcul de débit semblera à celle de calcul du débit par appel, mais dans ce cas, l'entête MPLS sera ajouté entre les couches 2 et 3.

3.3.2.3. Estimation du trafic entre les Edge Router :

De cette manière le trafic entre les différents routeurs de la périphérie deux à deux, seront estimés pour construire une matrice de trafic. La distribution du trafic à l'entrée de chaque *Edge Router* se fait selon :

- le nombre d'utilisateurs liés à chaque site,
- le plus court chemin entre deux *Edge Routers.*

La matrice de trafic aura alors la forme suivante entre les différents nœuds :

$$T_{tra} = \begin{bmatrix} 0 & A_{12} & A_{13} & A_{14} & A_{15} \\ A_{21} & O & A_{23} & A_{24} & A_{25} \\ A_{31} & A_{32} & 0 & A_{34} & A_{35} \\ A_{41} & A_{42} & A_{43} & 0 & A_{45} \\ A_{51} & A_{52} & A_{53} & A_{54} & 0 \end{bmatrix} \quad (3.05)$$

A_{ij} : le trafic généré entre les nœuds d'extrémité i et j.

C'est une matrice à diagonale nulle puisque la communication intra-site n'est pas tenu en compte.

3.3.3. Calcul des capacités :

3.3.3.1. Capacité du plus court chemin entre deux LER :

Nous nous intéressons ici aux trafics A13 et A31 entre les nœuds 1 et 3. Ces trafics suivront le plus court chemin entre ces deux nœuds. Les différentes métriques possibles sont 31, 36, 23, 42, 35 et 50.

Donc le plus court chemin sera celui qui passe par les routeurs internes 1'- 4'- 3' à travers les liens L3, L13, L8 et L6. Ce chemin véhiculera la totalité du trafic entre les nœuds 1 et 3. Ce chemin doit être capable de supporter le maximum de charge entre ces deux nœuds.

Dans tous les cas, le trafic suit le plus court chemin pour des raisons de coût et de qualité de service (QoS) puisque le plus court chemin assure généralement un faible délai. Le chemin le plus court doit supporter la somme des deux trafics étant donné que les artères sont considérées en *full-duplex.*

Dans le processus de dimensionnement, à part la qualité de service, il faut aussi tenir compte de la disponibilité du réseau. Pour cela, si le plus court chemin est incapable de véhiculer le trafic entre deux sites donnés, alors les chemins alternatifs doivent partager ce trafic suivant des coefficients basés sur des métriques de distances croissantes. En d'autres termes, plus le chemin alternatif est court, plus la portion de trafic qu'il supporte est importante.

3.3.3.2. Capacité des liens individuels :

Dans ce cas, les différents trafics acheminés entre les différents sites seront considérés. Nous supposons pour cela qu'il existe des trafics entre les différents sites. Au pire des cas, chaque lien doit être dimensionné de telle façon qu'il supporte la totalité du trafic qui le traverse :

$$C_{kv} = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \alpha_{ijv} \; T_{ijv} \tag{3.06}$$

C_{kv} : la capacité du lien individuel pour le trafic voix.

$$T_{ijv} = T_{i \to j} + T_{j \to i} \tag{3.07}$$

T_{ijv} : le trafic entre les sites i et j ;

α_{ijv} : un coefficient de pondération attribué au chemin supportant le trafic voix entre le site i et j.

Etant donné qu'il existe plusieurs chemins entre deux nœuds i et j et que le plus court chemin doit supporter la totalité du trafic. Alors, les autres chemins doivent partager le trafic entre eux selon des coefficients de pondération.

3.3.4. Débit généré par la signalisation :

La signalisation engendre un trafic qui peut charger le réseau et il est parfois nécessaire de tenir compte de ce trafic lors du dimensionnement. Deux protocoles de signalisation sont mis en œuvre dans les domaines MPLS afin d'assurer la réservation des ressources et l'établissement des LSPs entre deux LER. Nous nous sommes intéressés dans ce projet au trafic généré par le protocole CR-LDP. Le choix du protocole CR-LDP est justifié par le fait qu'il fut développé pour fonctionner sur MPLS. Le protocole RSVP est alors adapté pour fonctionner dans un domaine MPLS. Durant la phase de signalisation, nous considérons uniquement le débit engendré par :

- les messages d'affectation de label « Label Request » et « Label Mapping » ;

- les messages de libération de LSP « Label Release » et « Label Withdraw ».

Le « Label Request » message est utilisé par un *upstream LSR* pour demander à un *downstream LSR* l'allocation de Label. Généralement, c'est une requête d'établissement de connexion.

Le message « Label Mapping » est utilisé par le *downstream* LSR pour annoncer la construction d'un label pour une adresse de destination. Il est envoyé après la réservation de ressources et l'établissement de connexion pour faire le *mapping* entre les *incoming label* et les *outgoing label*. Il apporte le label assigné à la connexion à partir du commutateur *downstream* au commutateur *upstream.*

Le message « Label Withdraw » demande la libération d'un LSP. Ce message est envoyé par un *downstream* LSR à un *upstream* LSR parce que c'est lui qui contribue à l'affectation de label.

Le message «Label Release » est employé par un *upstream* LSR pour confirmer la libération d'un LSP. Un *upstream* LSR peut envoyer un message « Label Release » sans recevoir un message « Label Withdraw » à partir d'un *downstream* LS.

Pour déterminer le trafic engendré par la signalisation, nous procédons comme suit :

- *déterminer le nombre d'appels entre le nœud i et le nœud j quelque soit i et j* :

$$N_{apl_{i\to j}} = \frac{T_{i\to j}}{D_{apl_i}} \tag{3.08}$$

$T_{i\to j}$: le trafic du site i au site j ;

D_{apl_i} : le débit par appel au niveau du site i ;

- *calculer le débit de la signalisation* : étant donné que les signalisations nécessaires à l'établissement et à la libération ne s'effectuent pas en même temps, les liens sont dimensionnés de manière à ce qu'ils supportent la signalisation qui fournit le plus de débit.

Les formats des quatre messages de signalisation présentés dans les quatre dernières figures, montrent que les messages d'établissement d'un LSP fournissent le maximum de débit. La signalisation entre deux sites tient compte du nombre d'appels et du débit de signalisation :

$$S_{i\to j} = N_{apl_{i\to j}} \times D_{sig_i} \tag{3.09}$$

D_{sig_i} : le débit de signalisation par appel au niveau du nœud i ;

$N_{apl_{i\to j}}$: le nombre d'appels du nœud i au nœud j.

$$D_{sig_i} = \frac{T_{sig}}{t_{apl}} \tag{3.10}$$

T_{sig} : taille de la signalisation par appel ;

t_{apl} : durée d'un appel de i à j.

Vu que les liaisons sont de type *full-duplex*, la signalisation est tenue en compte dans les deux sens:

$$S_{ij} = S_{i\to j} + S_{j\to i} \tag{3.11}$$

Concernant la détermination de la capacité des liens pour la signalisation, le principe est le même que pour le trafic voix.

3.4. Dimensionnement pour un réseau offrant le service data : [14] [15] [16] [17]

Jusqu'à aujourd'hui, le service data est le service dominant sur le réseau IP bien que nous cherchons à intégrer d'autres services temps réel. Mais ce service est moins prioritaire puisqu'il n'a pas de contraintes temps de transmission. Un consensus général existe sur le fait que le trafic data sur IP n'est pas poissonien. Cependant les recherches ne sont pas d'accord sur l'approche à adopter pour modéliser le trafic sur Internet.

Le modèle BMAP est exploité pour modéliser le trafic IP. Ce modèle est conforme au trafic data où le besoin en débit des utilisateurs est aléatoire. Parfois, les flux arrivent en lots, d'autres fois l'activité des utilisateurs est nulle.

L’estimation du trafic data généré par les utilisateurs est difficile. Pour le web par exemple, la mesure du trafic se base sur l'observation des traces des fichiers Log du côté client et du côté serveur (respectivement *Upload* et *Download).*

3.4.1. Etude d'un cas particulier :

En se basant sur les contrats SLA entre les opérateurs et les abonnés, et en partant d'un cas particulier de deux types de contrats nommés respectivement SLA1 et SLA2 :

- SLA1 : garantit un débit montant de 64 Kbit/s et un débit descendant de 128 Kbit/s ;
- SLA2 : garantit un débit montant de 128 Kbit/s et un débit descendant de 256 Kbit/s.

Etapes pour l'étude du trafic Data

- Chaque site possède un nombre d'abonnés bien particulier. Pour chaque site, un certain pourcentage d'utilisateurs demandent un service de type SLA1 alors que les autres demandent un service de type SLA 2.
- Chaque trafic transite à travers un *backbone* IP/MPLS pour atteindre le réseau internet qui est relié à un parmi les *Edge nodes* du réseau dorsal.

3.4.2. Principe de distribution de charge :

Nous avons adopté la même topologie que dans le cas de la voix sauf que l'on a ajouté au niveau du site 4 une connexion à un réseau internet. Par la suite, nous procédons comme suit :

- calcul du débit du lien montant au niveau de chaque site selon les différents types de contrats établis entre l'opérateur et les clients suivant un accès ADSL :

$$D_{up_i} = N_{1i} * 64 + N_{2i} * 128 \quad (3.12)$$

N_{1i} : représente le nombre d'abonnés du site i ayant un contrat SLA1 ;

N_{2i} : représente le nombre d'abonnés du site j ayant un contrat SLA2;

- calcul du débit des liens descendants selon les conditions imposées par les contrats :

$$D_{down_i} = N_{1i} * 128 + N_{2i} * 256 \quad (3.13)$$

3.4.3. Capacité des liens :

La détermination des capacités suivra le même principe que pour la voix.

3.4.3.1. Capacité du plus court chemin pour le trafic data :

Le trafic dans les deux sens montant et descendant suit le plus court chemin. Ce dernier sera dimensionné de façon à ce qu'il supporte la totalité du trafic afin d'assurer certaines exigences en qualité de service à savoir un délai faible et un coût minimal.

Il est essentiel de prévoir d'autres chemins alternatifs qui vont partager respectivement les trafics *upload* et *download.* Cette distribution de charge se fait selon des coefficients de pondération. Elle est utile surtout en cas de surcharge au niveau du plus court chemin ou lorsque ce chemin n'est pas fonctionnel.

3.4.3.2. Capacité des liens individuels pour le trafic data :

Dans ce cas, nous suivons le même principe que pour le trafic voix. En général, l'accès Internet est supposé possible à partir d'un nœud i :

$$C_{kd} = \sum_{\substack{j=1 \\ i \neq j}}^{n} \propto_{ijd} T_{ijd}$$

$$T_{ijd} = T_{i \to j} + T_{j \to i} \qquad (3.14)$$

C_{kd} : capacité du lien individuel k ;

$\propto_{ijd}$: coefficient de pondération attribué au chemin supportant le trafic data entre le nœud i et j.

3.4.4. Capacité des liens supportant le trafic voix et data :

Pour les liens supportant le trafic data et le trafic voix, la capacité du lien doit être capable de supporter les deux trafics en même temps. La capacité totale est égale à la somme des deux capacités. Cette capacité est répartie suivant des coefficients entre le trafic temps réel (voix) et le trafic élastique. Pour le trafic data, le calcule du débit de la signalisation est inutile, pour connaître la capacité des liens. En fait, le débit défini dans le contrat que l'opérateur doit garantir tient compte de la signalisation.

En outre, l'évolution du réseau peut aussi être tenue en compte. Pour cela, un certain pourcentage peut s’ajouter à la capacité du réseau.

3.5. Conclusion :

L'un des moyens de dimensionnement d’un réseau est de déterminer les capacités des différents liens au niveau de ce réseau. La détermination de ces capacités considère certains éléments tels que : le trafic généré par appels, le nombre d'appels et surtout la manière de répartition des différents trafics entre les différents sites. Le passage d'un service voix à un service data, modifie alors le comportement du réseau et la distribution de charge.

CHAPITRE 4 : LES OUTILS DE SIMULATION POUR L'ETUDE DES PERFORMANCES DES RESEAUX.

4.1. Introduction : nécessité d'utiliser des simulations pour les études de performances de réseaux :

Les simulateurs de réseaux modernes sont des outils de première importance pour obtenir des informations relatives aux performances des réseaux quand toute autre approche théorique semble impossible. Par exemple, lorsque la topologie du réseau à étudier est complexe ou que les protocoles sont difficiles à modéliser pour pouvoir résoudre les modèles mathématiquement. Dans la majorité des cas, une analyse réaliste des performances des réseaux nécessite la prise en compte de protocoles existants, des équipements existants et des modèles de trafic réalistes. Cette approche descriptive des réseaux ne donne pas une résolution analytique. En effet, le nombre de variables à prendre en compte, souvent dépendantes du temps, est trop grand.

Les outils de simulations sont alors utilisés pour évaluer le comportement d'un réseau ainsi que ses performances en débit, délai, gigue ; quand tout approche théorique s'avère impossible.

Les simulateurs de réseaux sont initialement développés pour aider à l'étude, la validation et la création de protocoles réseaux. Par la suite, ils se sont enrichis pour effectuer des études de performances. Cependant, l'évaluation des performances par simulation pose de nombreux problèmes d'interprétation qu'il importe de prendre en compte.

4.2. Les simulateurs : [18]

4.2.1. Préambule :

Il faut distinguer les outils qui sont un « enrobage » fait autour de langage de simulation. Ils se sont développés spécifiquement pour un certain type de problèmes. Les principaux progrès par rapport à la génération précédente proviennent d'efforts importants en termes de génie logiciel.

En premier lieu, la plupart d'entre eux se caractérise par une interface graphique conviviale. Le développement des modèles peut se faire en grande partie en dessinant la structure du modèle étudié, modèle comportant un certain nombre de briques prédéfinies et spécifiques. L'exploitation des résultats est grandement simplifiée par l'utilisation de tracé de courbes et l'exploitation des résultats statistiques, fournis automatiquement par le logiciel.

En second lieu, la plupart de ces outils fournissent des bibliothèques de modèles riches. Ces bibliothèques de modèles construisent rapidement des programmes et donnent des résultats. En outre, l'utilisation des modèles de base et la définition de nouvelles briques augmente sensiblement la réutilisabilité des modèles ainsi développés. Notons qu'un utilisateur avisé peut développer des modèles de base plus complexes en utilisant directement le langage de programmation intégré à l'outil. Par conséquent, ces outils peuvent être utilisés sans grande connaissance en informatique ou en simulation. C'est là que commencent les limites d'utilisation de tels logiciels.

Cependant, les problèmes sont de plusieurs ordres. D'une part, le problème le plus fréquemment rencontré lors de l'utilisation des modèles de base, provient de la complexité ou de la simplicité de ces briques de base. D'autre part, la méconnaissance par la plupart des développeurs de tels outils de notion de base en simulation, pose également un problème. Dans de nombreux outils, les intervalles de confiance ne sont pas fournis. Parfois s'ils le sont, aucune distinction n'est faite entre les critères de performances pour lesquelles des moyennes temporelles sont réalisées et ceux pour lesquels des moyennes par rapport aux événements sont effectuées. Voici une figure représentant la structure interne d'un simulateur.

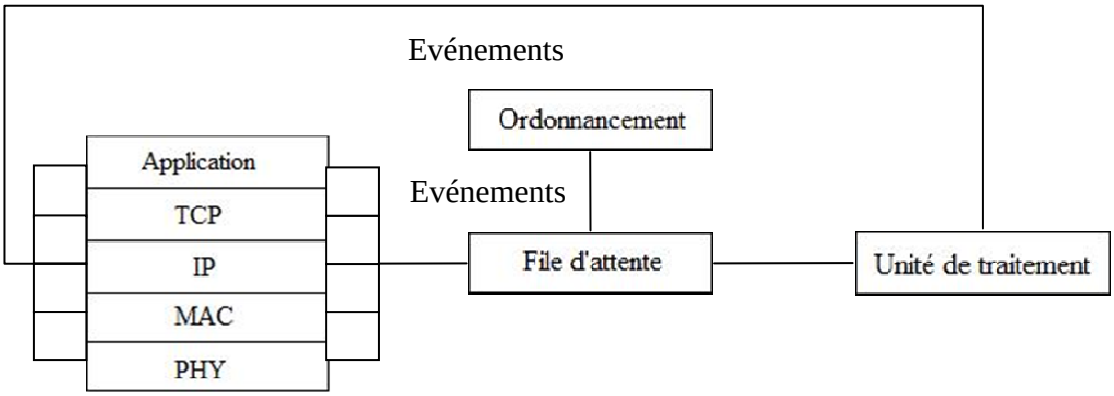

Figure 5.01: Structure interne d'un simulateur

4.2.2. NS-2 :

NS-2 ou Network Simulator-2 [18] est un simulateur de réseaux informatiques à évènements discrets orienté objets. Il fut développé en partie par le Network Research Group à Berkeley (LBNL). Actuellement, NS-2 fait partie intégrante du projet VINT, subventionné par le DARPA. Il a pour bût de mutualiser les outils liés aux études de performances dans les réseaux. Ce

simulateur est plutôt dédié aux technologies IP, aux protocoles identiques à TCP, aux différents protocoles de routage unicast et multicast, aux protocoles MAC 802.3, 802.11.

Par ailleurs, NS-2 donne également naissance à plusieurs sous-projets, comme ceux liés à l'émulation qui permettent d'injecter et de faire traiter par NS du trafic venant d'un véritable réseau actif ou encore SNS, qui permet de gérer des simulations de manière distribuée. NS-2 est le logiciel de simulation de référence dans le domaine des simulations réseaux, surtout des réseaux mobiles *ad hoc*. Effectivement, il possède une bibliothèque de modèles et de protocoles relativement riche et est libre de droit. La principale limite à l'utilisation de NS-2 vient de l'impossibilité de simuler des scénarios dans lesquels le nombre de nœuds augmente à raison de plus de 1 000 nœuds, cela est surtout visible dans les simulations des réseaux sans fil.

Finalement, bien que NS ne soit pas le pionnier dans le domaine de la modélisation des réseaux Internet en matière de logiciel, il y figure parmi les plus réputés. Grâce à sa bibliothèque de modèles et son importance dans la communauté scientifique, il demeure un outil de référence. En outre, les nombreux modèles parfois redondants et les problèmes de passage à l'échelle, attribue à NS-2 une structure interne plutôt anarchique. Cet outil est alors vieillissant et s'adapte mal parfois aux exigences des utilisateurs surtout, pour générer des modèles réalistes et à grandes échelles dans le domaine des réseaux sans fil.

4.2.3. GloMoSim/Qualnet :

GloMoSim [18] a été conçu par *UCLA computing laboratory* dans l'optique de résoudre le problème de passage à l'échelle des simulations. Il est disponible gratuitement mais n'est plus réellement maintenu, sauf dans la version commerciale du logiciel du Qualnet. GloMoSim effectue des simulations de manière distribuée sur plusieurs ordinateurs dans lesquels existent plusieurs millions de nœuds. Il utilise alors le langage Parsec, proche du langage C, mettant en œuvre des simulations à événements discrets de manière distribuée.

L'implantation de l'environnement de simulation est relativement fidèle au modèle OSI. L'API standard définit les interactions possibles entre les couches protocolaires et les nœuds. Cette approche stricte du modèle OSI permet aux utilisateurs d'implanter facilement de nouveaux protocoles, puisque les interfaces entre les couches et les fonctions de *callback* se définissent clairement. Ces fonctions jouent le rôle d'interception des événements et de leur aiguillage vers

des routines de traitements appropriées. Ainsi, l'utilisateur ne se souciera plus de la structure du simulateur et n'a plus qu'à implanter le mécanisme interne du protocole en question.

Le couple GloMoSim/Qualnet forme un ensemble cohérent et pratique pour la simulation de réseaux. Ce couple est performant en termes de rapidité d'exécution de modèles. Il donne aussi la parallélisation des simulations. Il possède une bibliothèque bien fournie en modèles surtout dans le domaine des réseaux sans fils. Cependant, les outils statistiques intégrés sont parfois inadaptés à une analyse de différentes valeurs en fonction du temps. Les utilisateurs devront donc générer leurs propres fichiers de traces. Toutefois, ce logiciel est un outil complet pour la simulation des réseaux informatiques surtout dans sa version commerciale.

4.2.4. Modline :

Cet outil a été créé autour de QNAP2 par Simulog [18]. Orienté file d'attente, il souffre d'un certain retard en termes de bibliothèques de modèles de réseaux. Son inconvénient repose sur le langage QNAP qui n'est seulement connu que par les spécialistes. En revanche, comme ses prédécesseurs QNAP et QNAP2, Modline résoud des modèles mathématiques et maîtrise bien les traitements statistiques.

QNAP est un « solveur » de réseaux de files d'attente, constitué d'un langage de spécification, d'un « solveur » analytique, d'un « solveur » Markovien et d'un modèle de simulation. Le « solveur » analytique donne la solution exacte pour un réseau de files d'attente vérifiant les conditions du théorème BCMP. Nous avons alors une solution à forme produit. Le « solveur » markovien, quant à lui, est utilisé pour tout type de réseau pouvant être représenté par un modèle markovien du premier ordre avec un nombre fini d'états. Lorsqu'aucun des deux solveurs précédents ne peut être appliqué, on procède à une simulation à événements discrets.

Bref, QNAP2 est un outil simple à utiliser, pédagogique et formateur. Il modélise facilement des problèmes qui pouvant être mis sous la forme de réseau de files d'attente.

4.2.5. Hyperformix Workbench :

Hyperformix Workbench n'est pas un simulateur de réseaux proprement dits [18]. En effet, les briques de bases du logiciel ne sont pas des protocoles réseaux, mais sont plutôt des blocs fonctionnels orientés files d'attente. Ce logiciel propriétaire effectue des simulations permettant une analyse de performance de gros systèmes logiciels de type *workflow*. Bien qu'il ne soit pas directement destiné à la modélisation de réseaux, son environnement de modélisation graphique

orienté file d'attente, comme pour Modline, offre des résultats positifs. Ces derniers ont été obtenus avec des modèles simples par rapport aux autres simulateurs, orientés simulation de protocoles réseaux Workbench. Ainsi ce simulateur construit de manière graphique, des réseaux de files d'attente pratiques, pour la modélisation des algorithmes utiles, par exemple.

Avec ce simulateur, les variables définies sont accessibles et offrent une possibilité d'analyse pour l'utilisateur. Les intervalles de confiance sont aussi disponibles pour valider les résultats de simulation ou pour piloter la durée des simulations. Chaque bloc fonctionnel utilisé graphiquement s'effectue en code écrit en langage C++. Selon sa volonté l'utilisateur peut enrichir les blocs fonctionnels déjà créés avec d'autres fonctionnalités. Cet outil de simulation reste donc un simulateur puissant, dont l'utilisation est assez instinctive. Cependant, ce simulateur nécessite une construction des modèles, propre à l'utilisateur. Tandis que pour les autres simulateurs fondés sur les protocoles réseaux, les couches et les différents leurs sont préalablement fournis.

4.2.6. OPNET :

OPNET [18] est développé pour la simulation de réseaux informatiques et de télécommunications. Commercialisé depuis quinze ans par la société MIL3 inc, il connaît un grand succès auprès des industriels. Il fournit de nombreuses bibliothèques de modèles de réseaux et reste ouvert, en raison des accès aux programmes sources de la plupart des modèles. OPNET Planner décrit les scénarios des simulations et est un outil d'analyse statistique.

OPNET est un logiciel complet et fourni en modèles. Cependant, son utilisation reste relativement ardue, par rapport à celle des logiciels fondés sur des algorithmes écrits en code ou en pseudo-code. Par hypothèse, un système reposant sur des composants hiérarchiques définis au travers d'automates, influence différemment les algorithmes réseaux. Ce fait ne se produit pas avec les simulateurs NS-2 et Glomosim.

4.2.7. J-SIM :

J-SIM [18] a une conception voisine de NS-2. Bien que ce logiciel soit écrit en java, il dispose d'une interface de commande en Otcl qui décrivant simplement des modèles relativement complexes. J-SIM a été conçu comme un simulateur modulaire où chaque protocole réseau est associé à une entité autonome. Le modèle choisi par le concepteur repose alors sur l'utilisation de composants autonomes plutôt que sur le paradigme classique de la programmation orienté objet. Cela évite le phénomène « hyper-spagetti » qui se produit lorsque les différentes entités sont

étroitement liées les unes aux autres. Il est alors impossible d'extraire un module pour le tester et corriger le programme qui lui correspond. Dans J-SIM, les entités sont reliées les unes aux autres grâce à l'utilisation de « contrats ».

J-SIM est pratique et ne nécessite pas un long apprentissage pour une utilisation correcte. Pourtant, J-SIM est relativement récent par rapport aux autres simulateurs, c'est pour cela que tous les protocoles présents dans le simulateur n'ont pas encore été suffisamment validés et testés. Quelques erreurs d'implantation y subsistent donc. De plus, ce simulateur possède une bibliothèque de modèle peu fournie, bien qu'il ne soit l'un des seuls à disposer d'un modèle très pointu pour la modélisation de réseaux de capteurs. C'est donc un logiciel qui n'est pas encore mature, dont l'évolution mérite d'être suivie, vu ses plusieurs fonctionnalités absentes dans les autres outils.

4.2.8. JiST/SWANS :

Le couple JiST/SWANS [18] constitue un logiciel Open Source dédié à la simulation des réseaux sans fils. Ils sont écrits en java. Le cœur du simulateur est réalisé par JiST et la partie contenant les différentes couches réseaux et la partie radio sont, quant à elles, implantées dans SWANS. JiST *(Java in Simulation Time)* transforme la machine virtuelle java en simulateur à évènements discrets dont les principales primitives sont écrites en « bytecode » java.

Le véritable apport de ce logiciel par rapport aux autres est d'avoir été implanté de manière véritablement modulaire. Chaque entité (nœuds, etc.) possède sa propre horloge interne ce qui lui permet de ne pas avoir à attendre de se synchroniser avec les autres entités pour pouvoir effectuer les traitements des messages. Le couple JiST/SWANS forme alors un outil performant qui ayant pour objectif principal, d'effectuer des simulations pouvant contenir des milliers de nœuds. Un travail a été fait pour réduire au maximum la place mémoire occupée par les entités dans le simulateur. Pour cela, un mécanisme de passage de messages entre les différentes entités se fait par références, sans aucune copie, ce qui à pour conséquence de réellement diminuer l'occupation mémoire de chaque modèle par rapport aux autres simulateurs. Le couple de logiciels JiST/SWANS réussit à allier des performances en taille de modèle simulable et en rapidité d'exécution des modèles.

Bien que l'approche suivie par les concepteurs de JiST et SWANS soit particulièrement séduisante, JiST est néanmoins un simulateur récent dans le domaine et reste sujet à de

nombreuses erreurs dans les modèles proposés. Le nombre de modèle déjà implantés dans le simulateur est relativement faible. Malgré sa jeunesse, JiST/SWANS est un simulateur de réseaux qui a pris en compte le passage à l'échelle des modèles à simuler et sûrement le plus performant des outils à l'heure actuelle, en termes de capacité de traitement sur un environnement de simlation qui n'est pas parallélisable. Ce logiciel en pleine évolution risque de devenir dans peu de temps pleinement opérationnel et réellement attrayant pour les utilisateurs.

4.2.9. OMNET ++ :

OMNET++ [18] est aussi un logiciel Open Source de simulation à évènements discrets. Contrairement aux logiciels présentés précédemment, il n'est pas réellement spécifique à la modélisation de réseaux informatiques. OMNET++ est un logiciel relativement récent et il est dans son implantation et son ergonomie, relativement proche du logiciel OPNET. C'est un logiciel écrit en C++ qui dispose d'une interface graphique de programmation relativement claire et intuitive. Ce logiciel offre aussi, en plus d'une interface graphique, une interface de programmation appelée langage GNED relativement proche de C. OMNET ++. Ce dernier est un logiciel qui, dans sa structure interne, est orienté « messages ». En d'autres termes, les modules modélisés dans ce logiciel, échangent des messages au travers de primitives de programmation bien structurées et standardisées. La conception de modèles est alors plus facile. Un modèle se crée donc de la manière à définir les entités, leurs entrées, leurs sorties ainsi que les liens entre les entités.

OMNET++ est certainement un simulateur qui regroupe beaucoup de qualités : il a une interface graphique, de nombreux modèles et des outils statistiques performants. Il offre la possibilité de faire des simulations en cluster. Cependant, OMNET++ est un outil multifonction difficile à aborder et compliqué à utiliser, surtout lors de la mise en œuvre des modèles qui font intervenir plusieurs couches réseaux. En effet, l'utilisateur doit alors entièrement spécifier les modules et les interactions entre les composants du logiciel et ce, pour chaque modèle que l'utilisateur veut simuler.

4.2.10. Akaroa-2 :

C'est un simple outil statistique qui se greffe sur un simulateur et un environnement de développement qui parallélise des simulations. Cet outil calcule des moyennes, des variances, ainsi que des intervalles de confiances sur des variables préalablement choisies dans la structure interne d'un simulateur. Les outils de simulations décrits précédemment ne possèdent pas d'outils

statistiques très pointus, effectuant une analyse fine des résultats de simulation. Akaroa-2 comble alors naturellement ce vide grâce aux nombreuses fonctionnalités statistiques implantées.

4.2.11. Simulateurs récents ou encore en développement :

Des simulateurs comme le NAB, le ReactiveML et le GTNetS sont en voie de conception.

Le NAB [18] est simulateur développé par l'Ecole polytechnique fédérale de Lausanne (EPFL) dans le cadre du projet MICS sur les réseaux mobiles de nouvelles générations. NAB a donc été initialement développé pour un type particulier de réseaux *ad hoc* et naturellement il privilégie la simulation de ce type de réseau. La particularité de NAB est d'être écrite dans le langage Ocaml proche de la philosophie du langage Lisp (langage fonctionnel). De plus, il fait des simulations à grande échelle, la simulation de 1000 nœuds ne pose aucun problème.

Le GTNetS (Georgia Tech Network Simulator) [18] est un autre simulateur écrit par l'équipe Maniacs travaillant à l'Université de Georgia Tech. Ce simulateur est intégralement écrit en C++. Il a la particularité de respecter avec précision la structure des couches réseaux des implantations classiques de telles que celles des systèmes Unix, ainsi que la véritable des différents PDU (Protocol data Unit) disponibles dans chacune des couches réseaux (très pratique pour comparer avec des traces de réseau réel). A part ces différentes fonctionnalités, GTNetS offre aussi une large bibliothèque de modèles qui peuvent être utilisés directement sans avoir besoin d'implanter quoi que ce soit.

4.2.12. Langage dédiée à la simulation :

ReactiveML [18] n'est pas un simulateur proprement dit, mais un langage proche d'Ocaml, c'est en fait une surcouche de ce langage. Il offre la possibilité de gérer les événements de manière dynamique (gestion dans le langage lui-même et non plus par le cœur du programme). ReactiveML a été utilisé pour faire un simulateur de réseau *ad hoc.* Il offre, grâce à la gestion fine des événements, un gain de performance par rapport à d'autres types de simulateurs qui n'utilisent ReactiveML que comme langage. Il offre donc des gains de performances non négligeables pour mettre en œuvre une simulation où le nombre de nœud est un facteur prépondérant.

SIMSCRIPT [18] est certainement le premier langage de programmation dédié à la simulation. Il a été développé par Harry Markowitz en 1963 lorsqu'il travaillait pour Rand Corporation. Ce langage de programmation qui, à l'origine, est dédié à la création de simulation à évènements

discrets, a largement influencé d'autres types de langage comme Simula. Ce dernier en reprend les grands principes en y ajoutant une modularité propre aux langages dits orientés objet. Aujourd'hui, SIMSCRIPT est utilisé et maintenu par CACI. CACI commercialise le simulateur SIMSCRIPT II.5 reposant toujours sur le même langage. Ce langage modélise facilement les performances des systèmes dits collaboratifs et transactionnels et entre autres, les systèmes « réseaux » de manière macroscopique et les systèmes fondés sur des réseaux de files d'attente. Avec SIMSCIPT II.5, la modélisation des systèmes de manière entièrement graphique est maintenant facile. Ce qui n'était pas le cas avec les versions précédentes telles que MODSIM III qui n'est plus maintenu aujourd'hui.

CHAPITRE 5 : SIMULATION SOUS NS-2

5.1. Présentation générale du simulateur ns2 :

NS2 est un outil de recherche utile pour le design et la compréhension des protocoles. Il permet à l'utilisateur de concevoir un réseau et de simuler les communications entre les nœuds. Le simulateur utilise le langage orienté objet OTCL dérivé du TCL pour la description des conditions de simulation sous forme de scripts. Dans ce script, l'utilisateur fournit la topologie du réseau, les caractéristiques des liens physiques, les protocoles utilisés, le type de trafic généré par les sources, les évènements.

Le résultat d'une simulation est un fichier texte contenant tous les évènements de la simulation comme le mouvement d'un nœud, les pannes dans les réseaux, l'envoi, la réception et la perte de paquets de données ou de routage. Ce fichier résultat est appelé fichier trace. Un traitement ultérieur de ce fichier en soustrait des informations utiles à l'évaluation des aspects réseaux à étudier.

Le simulateur permet de créer un fichier d'animation visualisant la simulation sur une interface graphique appelée NAM.

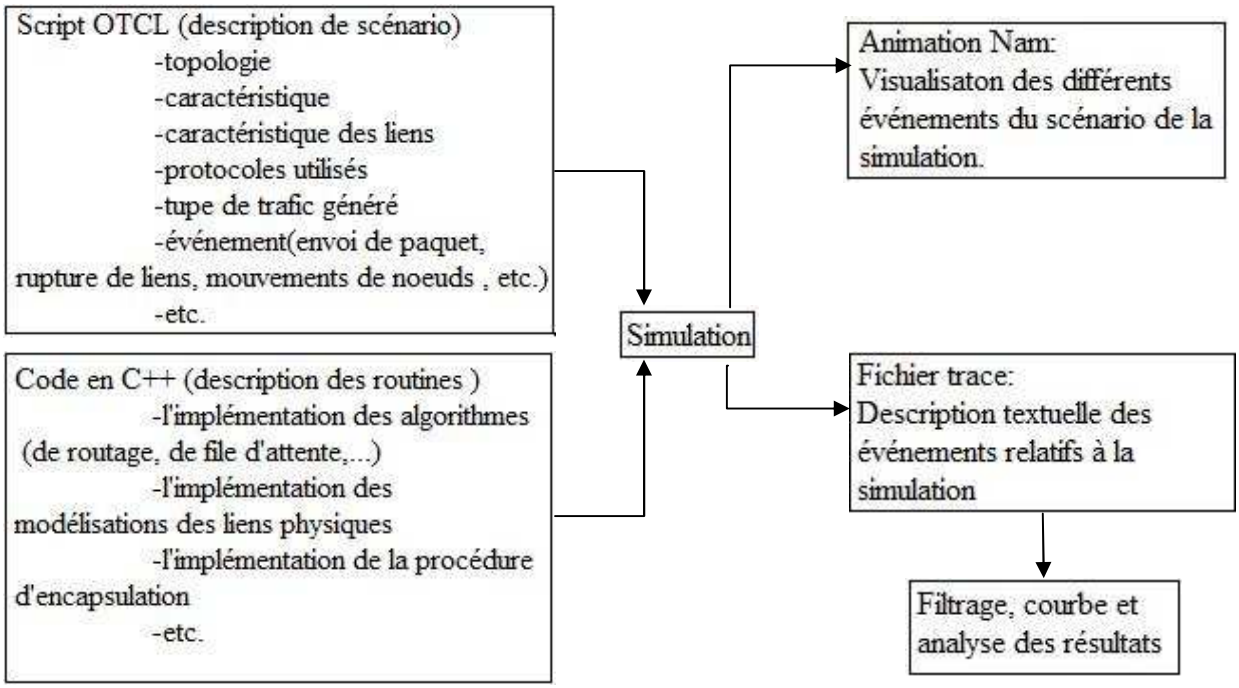

Figure 5.01 : Les étapes d'une simulation sur ns2

L’utilisation de NS-2 peut être :

- De base : le simulateur est utilisé tel quel, c'est-à-dire que les codes sont fournis par les développeurs. Le scénario ne change pas c'est-à-dire le script tcl ;
- Intermédiaire : les fonctionnalités écrites en C++ sont ajoutées, sans modifications du cœur du simulateur. Par exemple, l’apport des modifications à un algorithme de routage ;
- Avancée : nous développons notre propre code écrit en C++ et nous modifions le cœur du simulateur. Prenons l’exemple de l’ajout d’un protocole de couches 2 ou 3.

L’utilisation de ces outils et l’exploitation des résultats requièrent certaines compétences :

- Connaissance en C++ et maitrise des concepts de la programmation orientée objet pour l’édition des routines qui composent le simulateur ;
- Connaissance du langage script OTCL pour la description des simulations et le prétraitement des données ;
- Maîtrise de l’outil indispensable à l’extraction des données des fichiers (AWK) ;
- Capacité à produire des graphes à partir des fichiers produits, à l’aide de Matlab.

5.2. Simulation d’un réseau MPLS pour l’évaluation des performances d’un protocole de routage :

Dans cette partie, nous allons développer deux exemples de simulation. Le premier exemple explique comment créer une simulation d’un réseau MPLS contenant 13 nœuds que nous allons présenter dont 3 LER, l’une pour l’entrée et les 2autres pour la sortie du domaine MPLS, 8 LSR et les deux autres servent pour la source et la réception des données. Tandis que, le deuxième exemple, explique comment créer une simulation d’un réseau MPLS contenant cette fois-ci 15 nœuds dont 4 LER, 2 pour l’entrée dans le domaine et 2 pour la sortie, 7 LSR faisant partie du domaine MPLS et les 4 autres divisés en 2 parties, l’une étant pour les sources de données et l’autre pour celles des réceptions et comment utiliser les informations recueillis par NS pour évaluer le protocole de routage mis en œuvre.

5.2.1. Simulation d'un réseau MPLS d'une source vers une réception :

La topologie consiste à envoyer des paquets de données de la source vers la réception, node0 et node12. Nous interrompons des liens sur le chemin pris par les paquets pendant un certain temps, ils seront ré-routés vers un autre chemin pour accéder à la destination.

Topologie du réseau:

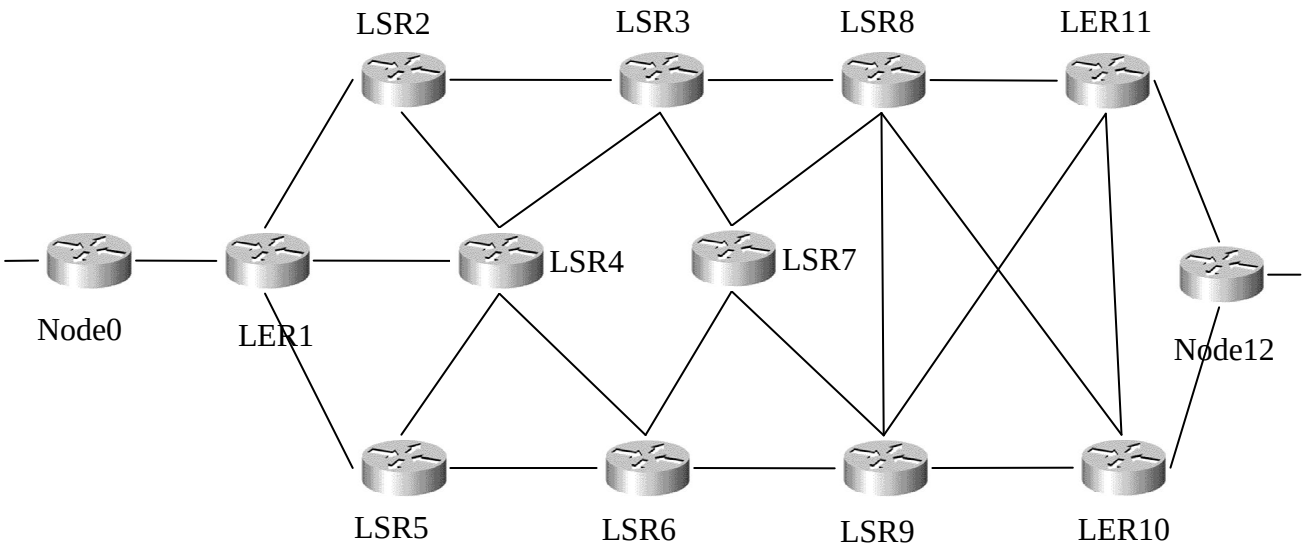

Figure 5.02 : Topologie du réseau MPLS

Voyons à présent le script permettant cette topologie du système :

Avant tout, nous avons besoin de créer un objet simulator que nous nommerons « ns ». Ceci est réalisé par la commande :

```
set ns [new Simulator]
```

Ensuite nous ouvrons les fichiers pour l'écriture des données trace pour NAM et Xgraph :

```
set nf [open mpls.nam w]
$ns namtrace-all $nf
set f0 [open mpls.tr w]
```

Puis la procédure Finish qui ferme le fichier trace et démarre le fichier Xgraph et NAM

```
proc finish {} {
        global ns nf f0
```

```
        $ns flush-trace
        close $nf
        close $f0
    exec nam mpls.nam &
        exit 0
}
```

Nous insérons cette commande pour le routage dynamique pour résoudre les problèmes de rupture de liens.

```
 #
$ns rtproto DV
```

Les lignes suivantes permettent la création des nœuds et de la configuration du domaine MPLS. Pour cette configuration de MPLS la commande sera précédée par : « node-config-MPLS ON » et sera succédée par : « node-config-MPLS OFF »

```
set node0  [$ns node]
$ns node-config -MPLS ON
set LSR1   [$ns node]
set LSR2   [$ns node]
set LSR3   [$ns node]
set LSR4   [$ns node]
set LSR5   [$ns node]
set LSR6   [$ns node]
set LSR7   [$ns node]
set LSR8   [$ns node]
set LSR9   [$ns node]
set LSR10  [$ns node]
set LSR11  [$ns node]
$ns node-config -MPLS OFF
set node12 [$ns node]
```

Un nouveau objet node est créé par la commande « $ns node ». Les codes ci-dessous créent les nœuds et les assimilent entre eux.

```
$ns duplex-link $node0 $LSR1  1Mb  10ms DropTail
$ns duplex-link $LSR1  $LSR2  1Mb  10ms DropTail
$ns duplex-link $LSR1  $LSR5   1Mb  10ms DropTail
$ns duplex-link $LSR1  $LSR4   1Mb  10ms DropTail
$ns duplex-link $LSR2  $LSR3   1Mb  10ms DropTail
$ns duplex-link $LSR2  $LSR4   1Mb  10ms DropTail
$ns duplex-link $LSR3  $LSR4   1Mb  10ms DropTail
$ns duplex-link $LSR3  $LSR7   1Mb  10ms DropTail
$ns duplex-link $LSR3  $LSR8   1Mb  10ms DropTail
$ns duplex-link $LSR4  $LSR5   1Mb  10ms DropTail
$ns duplex-link $LSR4  $LSR6   1Mb  10ms DropTail
$ns duplex-link $LSR5  $LSR6   1Mb  10ms DropTail
$ns duplex-link $LSR6  $LSR7   1Mb  10ms DropTail
$ns duplex-link $LSR6  $LSR9   1Mb  10ms DropTail
$ns duplex-link $LSR7  $LSR9   1Mb  10ms DropTail
$ns duplex-link $LSR7  $LSR8   1Mb  10ms DropTail
$ns duplex-link $LSR8  $LSR10  1Mb  10ms DropTail
$ns duplex-link $LSR8  $LSR11  1Mb  10ms DropTail
$ns duplex-link $LSR8  $LSR9   1Mb  10ms DropTail
$ns duplex-link $LSR9  $LSR10  1Mb  10ms DropTail
$ns duplex-link $LSR9  $LSR11  1Mb  10ms DropTail
$ns duplex-link $LSR10 $LSR11  1Mb  10ms DropTail
$ns duplex-link $LSR10 $node12 1Mb  10ms DropTail
$ns duplex-link $LSR11 $node12 1Mb  10ms DropTail
```

Ces lignes demandent au simulateur ‘ns’ de connecter les nœuds, par exemple le node0 et LSR1, avec un lien duplex, une bande passante de 1 Mégabit, un délai d’acheminement de 10 ms et une file d’attente DropTail c'est-à-dire FIFO.

Les lignes précédentes gèrent la disposition des nœuds dans le réseau.

```
$ns duplex-link-op $node0 $LSR1  orient right
```

```
$ns duplex-link-op $LSR1  $LSR2  orient right-up
$ns duplex-link-op $LSR2  $LSR3  orient right
$ns duplex-link-op $LSR3  $LSR4  orient down
$ns duplex-link-op $LSR4  $LSR5  orient left
$ns duplex-link-op $LSR1  $LSR5  orient right-down
$ns duplex-link-op $LSR5  $LSR6  orient right
$ns duplex-link-op $LSR6  $LSR7  orient up-right
$ns duplex-link-op $LSR7  $LSR8  orient down
$ns duplex-link-op $LSR8  $LSR9  orient left-down
$ns duplex-link-op $LSR9  $LSR10  orient left-down
$ns duplex-link-op $LSR10 $LSR11 orient right
$ns duplex-link-op $LSR11 $node12 orient right
```

Ce procédé place le coût du lien le long d'une seule direction, sa valeur par défaut est '1' :

```
$ns cost $LSR4 $LSR5 3
$ns cost $LSR5 $LSR4 3
```

Nous avons deux options pour l'installation, la configuration de l'agent LDP dans tout le nœud et l'emplacement de la fonction de restauration de path qui réachemine le trafic.:

- New qui crée un nouveau chemin alternatif s'il n'existe pas ;
- Drop qui ne crée aucun nouveau chemin alternatif.

Nous allons ajuster la longueur de la boucle pour adresser tout le LSRs :

```
for {set i 1} {$i < 12} {incr i} {
        set a LSR$i
        for {set j [expr $i+1]} {$j < 12} {incr j} {
                set b LSR$j
                eval $ns LDP-peer $$a $$b
        }
        set m [eval $$a get-module "MPLS"]
        $m enable-reroute "new"
}
```

Les couleurs de ldp-message dans NAM :

```
$ns ldp-request-color       blue
$ns ldp-mapping-color       red
$ns ldp-withdraw-color      yellow
$ns ldp-release-color       orange
$ns ldp-notification-color  green
```

Nous définissons ensuite la stratégie de déclenchement, et marque le mode de commande de distribution et l'étiquette d'arrangement d'attribution et de distribution. Quand la ligne suivante est omise, la stratégie de déclenchement est placée dans data-driven.

```
Classifier/Addr/MPLS set control_driven_ 1
Classifier/Addr/MPLS enable-on-demand
Classifier/Addr/MPLS enable-ordered-control
```

La stratégie de déclenchement est définie par LSR par la commande suivante :

```
[$LSR5 get-module "MPLS"] enable-control-driven
[$LSR2 get-module "MPLS"] enable-data-driven
```

La commande suivante allume toutes les traces au stdout :

```
Agent/LDP set trace_ldp_ 1
Classifier/Addr/MPLS set trace_mpls_ 1
```

Nous utiliserons 'List' pour l'établissement des programmes des événements.

```
$ns use-scheduler List
```

Nous définissons le procédé pour créer une circulation de CBR et pour la relier à un agent UDP

```
proc attach-expoo-traffic { node sink size burst idle rate } {
        global ns


        set udp [new Agent/UDP]
```

```
        $ns attach-agent $node $udp

        set traffic [new Application/Traffic/Exponential]
        $traffic set packetSize_ $size
        $traffic set burst_time_ $burst
        $traffic set idle_time_ $idle
        $traffic set rate_ $rate
        $traffic attach-agent $udp

        $ns connect $udp $sink
        return $traffic
}
```

Nous créons le point de convergence du trafic et l'attache au noeud12

```
set sink0 [new Agent/LossMonitor]
$ns attach-agent $node12  $sink0
```

Nous créons une source de trafic démarrons par le node0 et se terminant au node12 :

```
set src0 [attach-expoo-traffic $node0  $sink0 200 0 0 400k]
```

Nous créons ensuite un agent TCP qui le connecte à une application comme le FTP ou le Telnet, qui produit les données.

```
set tcp [new Agent/TCP]
$ns attach-agent $node0 $tcp
set ftp [new Application/FTP]
$tcp set packetSize_ 1024
$ftp attach-agent $tcp

set sink [new Agent/TCPSink]
$ns attach-agent $node12 $sink

$ns connect $tcp $sink
```

La procédure suivante vide les paquets envoyé/reçu au message de sollicitation (en cas de source tcp) :

```
proc monitor {} {
  global tcp

  $tcp instvar ndatapack_
  puts "packets send: $ndatapack_"
  $tcp instvar nackpack_
  puts "packets received: $nackpack_"
}
```

Nous définissons la procédure qui enregistre périodiquement la bande passante reçue par le point de convergence du trafic écrite dans le fichier f0 :

```
set totalpkt 0
proc record {} {
      global sink0 f0 totalpkt
         set ns [Simulator instance]
                  set time 0.005
                  set bw0 [$sink0 set bytes_]
                  set now [$ns now]

                  # Calcule la bande passante (en Mbit/s) et l'écrit dans le fichier f0
                  puts $f0 "$now [expr $bw0/$time*8/1000000]"

                  # Remet à zéro la valeur du bites sur le point de convergence
                  $sink0 set bytes_ 0

                  # Remettre le procédé à plus tard
                  $ns at [expr $now+$time] "record"
                  set bw0 [expr $bw0 / 200]
                  set totalpkt [expr $totalpkt + $bw0]
         }
```

Le sous-programme qui permet de vider le nombre de paquets reçus et de calculer la procédure de commande du message de sollicitation.

```
proc recv-pkts {} {
    global totalpkt
    flush stdout
    puts "The Number of Total received packet is $totalpkt"
}
```

Le sous-programme qui démarre la procédure "record" :

```
$ns at 0.0   "record"
```

Le sous-programme qui démarre la source :

```
$ns at 0.1   "$src0 start"
```

L'événement suivant définit l'interruption des liens . Par exemple, nous ferons en sorte que le lien entre LSR2 et LSR 3 soit interrompu pendant 0.1 seconde.

```
$ns rtmodel-at 0.3 down $LSR2 $LSR3
$ns rtmodel-at 0.4 down $LSR1 $LSR4
$ns rtmodel-at 0.3 down $LSR8 $LSR10
$ns rtmodel-at 0.4 down $LSR9 $LSR10
```

Nous définissons la procédure quand les liens seront rétablis :

```
$ns rtmodel-at 0.4 up   $LSR2 $LSR3
$ns rtmodel-at 0.6 up   $LSR1 $LSR4
$ns rtmodel-at 0.5 up   $LSR8 $LSR10
$ns rtmodel-at 0.5 up   $LSR9 $LSR10
```

Les résultats de trace (MPLS/LDP packets) à un LSR donné sont vidés au message de sollicitation :

```
$ns at 0.1 "[$LSR1 get-module MPLS] trace-mpls"
#$ns at 0.1 "[$LSR3 get-module MPLS] trace-LDP"
```

L’arrêt de la source est de 0.6 secondes du début de l’émission des paquets :

```
$ns at 0.6 "$src0 stop"
```

Nous faisons appel à une procédure "recv-pkts" :

```
$ns at 0.7 "recv-pkts"
```

Ces commandes montrent l'erb/lib/pft-table du LSR :

```
$ns at 0.7 "[$LSR1 get-module MPLS] erb-dump"
$ns at 0.7 "[$LSR1 get-module MPLS] lib-dump"
$ns at 0.7 "[$LSR1 get-module MPLS] pft-dump"
```

Nous faisons appel à la procédure "finish" pour terminer le programme :

```
$ns at 0.7 "finish"
```

La simulation démarre par la commande ci-dessous :

```
$ns run
```

Nous devons maintenant enregistrer le fichier tcl puis démarrer la simulation dans le terminal du système par la commande «./ns mpls.tcl ».

Figure 5.03 : Fenêtre Terminal

Sur nam, nous obtenons la figure suivante :

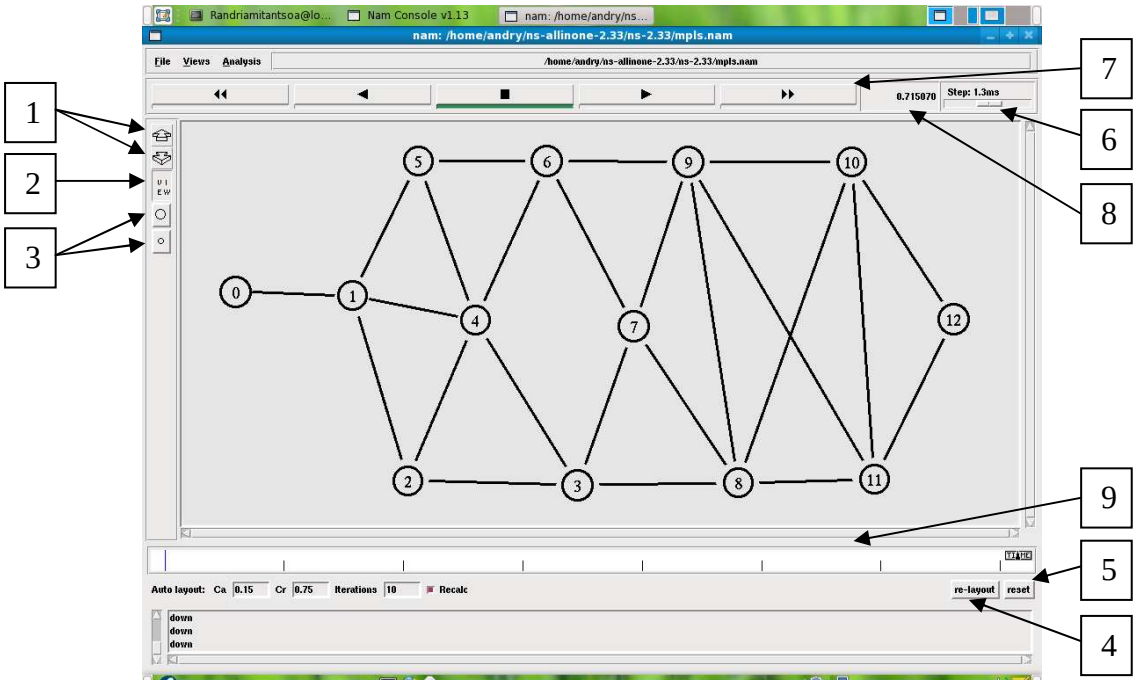

Figure 5.04 : Présentation de nam et topologie du réseau

Nam, nous propose différentes options pour l'amélioration de la qualité de vue de l'utilisateur. Par exemple, nous avons :

1- Une fonction ' Zoom Avant' et 'Zoom Arrière' ;
2- Un bouton 'EDIT/VIEW' permettant d'éditer la position du nœud dans le réseau et de visualiser sa position ;
3- Deux boutons permettant l'agrandissement et le rétrécissement des nœuds ;
4- Un bouton 're-layout' pour permettre la ré-disposition automatique des nœuds dans le réseau ;
5- Une fonction 'reset' qui remet tout le réseau dans le désordre ;
6- Un curseur pour une bonne visualisation de la simulation en ms ;
7- Des boutons pour le démarrage, l'avance rapide, le recul, le démarrage inverse et pour l'arrêt de la simulation dans nam.

A part, ces différentes fonctions pratiques pour l'utilisateur nam a aussi :

8- Un cadran pour la durée de la simulation en seconde;
9- Un autre pour son évolution dans le temps.

Pour démarrer alors la simulation dans nam, nous cliquons sur le bouton ‘play’. Nous pouvons voir les routeurs envoyer des requêtes et répondre à celles des autres pour l’acheminement des paquets vers la destination en suivant le chemin le plus court.

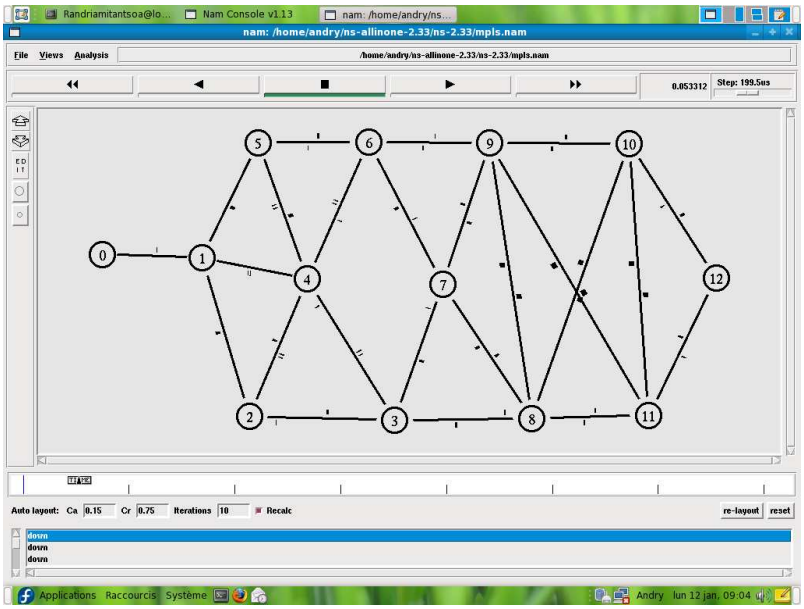

Figure 5.05 : Etape 1 de la simulation d’un réseau MPLS

A l’instant 0.1seconde, le nœud de départ commence à émettre des paquets sur le chemin le plus court :

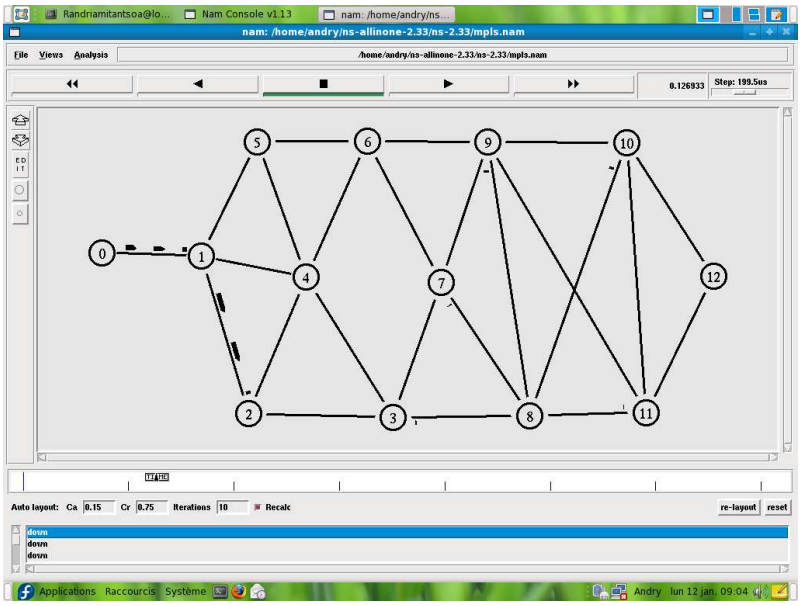

Figure 5.06: Etape 2 de la simulation d’un réseau MPLS

Pour cette topologie, les paquets ont empruntés les nœuds de 0 à 12 en passant par 1, 2, 3, 8 et 10.

A un temps arbitraire, nous interrompons l'émission de deux liens entre les nœuds 2, 3 et 8,10. Ainsi les paquets, suivant les interruptions, seront déviés vers un autre routeur pour arriver à destination.

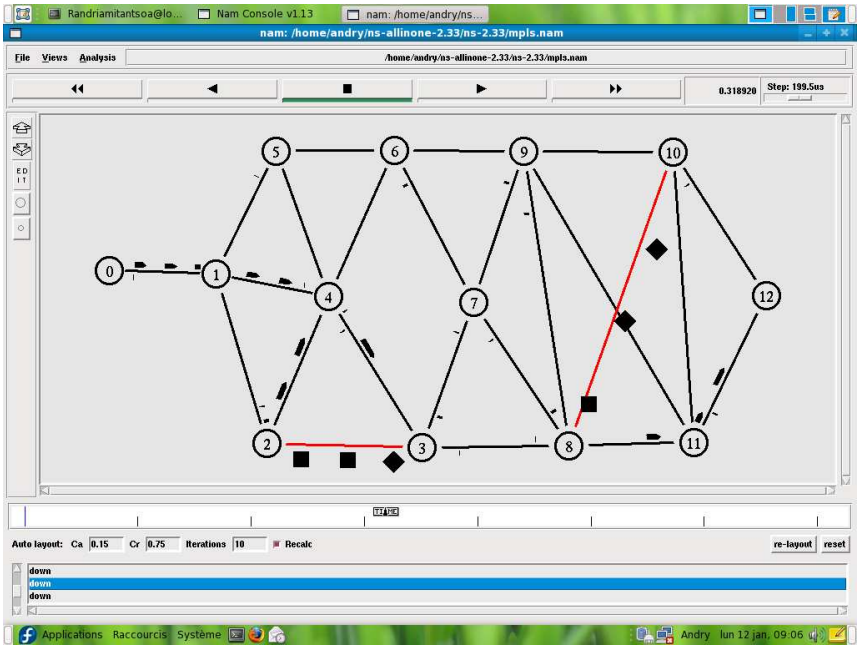

Figure 5.07 : Etape 3 de la simulation d'un réseau MPLS

A la fin de la simulation, nous aurons un fichier trace « mpls.tr » qui contient tous les événements relatifs aux déplacements des paquets, à l'envoi, à la réception et à la perte de paquets de données et de routages.

5.2.2. Simulation d'un réseau ad hoc dense :

Dans cet exemple, nous montrons la possibilité de ns2 pour créer une simulation d'un réseau ad hoc plus dense (quelques dizaines de nœuds). Le réseau ad hoc utilise les informations recueillies par ns2 pour évaluer le protocole de routage mis en œuvre. Nous pouvons alors consulter le script entier dans l'annexe 6.

5.2.3. Le script tcl :

Tout d'abord nous devons changer le nombre de nœuds :

```
set val (nn) 100
```

Ensuite nous spécifions la variable val(sc) le « node-movement file ».

Ces deux fichiers contiennent le détail du trafic généré par chaque nœud et les mouvements de chaque nœud pendant la durée de simulation.

Ces fichiers sont créés automatiquement par les programmes « setdest » et « cbrgen ».

```
set val (cp) "/root/examples/scen_traff_100"

set val (sc) "/root/examples/scen_mvt_100"

# Define node movement model

puts "Loading connection pattern…"

source $val (cp)

# Define traffic model

puts "Loading scenario file…"

source $val (sc)
```

5.2.4. Création du node-mouvement file:

Le *Node-movement generator* (setdest) est un programme disponible sous le répertoire «~ns/indep-utils/cmu-sce-gen/setdes ». Il crée de façon automatique un fichier contenant un scénario de mouvement aléatoire d'un nombre donné de nœuds mobiles dans une topologie donnée.

La syntaxe est la suivante :

```
.setdest [-n num_of_nodes] [-p pausetime] [-s maxspeed] [-t simtime]\

[-x maxx] [-y maxy] > [outdir/movement-file]
```

Avec

[-n num_of_nodes] : le nombre de nœuds dans la simulation

[-p pausetime] : le temps de pause lors du changement

[-s simtime] : la durée de simulation

[-x maxx] [-y maxy] : la dimension de la topologie

[outdir/movement-file] : le chemin et nom du fichier sortie

Ainsi il en résulte la commande ci-après :

./setdest –n 100 –p 0.0 -s 10.0 -t 200 -x 500 -y 500 > scen_mvt_100

Cette commande permet de créer un scénario de mouvement pour une simulation qui dure 200 s, qui contient 100 nœuds et qui se déplace dans un carré de 500 m x 500 m à une vitesse maximale de 10m/s.

Le résultat sera stocké dans le fichier « scen_mvt_100 ». Ce fichier contient des entrées de la forme :

```
$node_(0) set X_ 5.0

$node_(0) set Y_ 2.0

$node_(0) set Z_ 0.0
```

En effet, cette partie définit les positions initiales des 100 nœuds. La deuxième partie du fichier « scen_mvt_100 » contient des entrées de la forme :

$ns_ at 2.000000000000 “$node_(0) setdest 90.441179033457

44.896095544010 1.37355690010”

Ces entrées définissent le mouvement que prendra chaque nœud (instant, destination et vitesse).

5.2.5. Création du «Traffic-connection file »

La création du « Traffic-connection file » se fait par le programme « cbrgen.tcl » qui est disponible sous le répertoire « ~ns/indeputils/cmu-scen-gen ».

Ce programme crée des scénarios de connexions et d’échanges de trafic (tcp ou cbr) entre un nombre donné de nœuds et un débit donné.

La syntaxe est la suivante :

```
ns cbrgen.tcl [-type cbr | tcp]  [-nn nodes]  [-seed seed]  [-mc connections]

[-rate rate]  >  scen_traff_100
```

Avec

[- type cbr | tcp] : le type de trafic entre les nœuds

[-nn nodes] : le nombre de nœuds dans la simulation

[-seed seed] : un nombre utilisé pour la fonction aléatoire

[-mc connections] : le nombre de connexion à établir

[-rate rate] : le débit entre les nœuds (pkt/s)

scen_traff_100 : le nom du fichier de sortie

Ainsi il en résulte la commande suivante :

```
ns cbrgen.tcl –type cbr –nn 100 –seed 1.0 –mc 20 –rate 4.0 > scen_traff_100
```

Elle permet de créer vingt connexions cbr dans une topologie contenant 100 nœuds mobiles et d'avoir un débit de 4 pkt/s.

Le fichier « scen_traff_100 » contient des entrées de la forme :

```
Set ydp_(0) [new Agent/UDP]

$ns_ attach-agent $node_(2) $udp _(0) set null_(0) [new Agent/Null]

$ns_ attach-agent $node _(3) $null_(0) set cbr_(0) [new Application/traffic/CBR]

$cbr_(0) set packetSize_ 512

$cbr_(0) set interval_0.25

$cbr_(0) set random_ 1

$cbr_(0) set maxpkts_10000

$cbr_(0) set attach-agent $udp_ (0)
```

```
$ns_ connect $udp_ (0) $null_ (0)

$ns_ at 82.557023746220864 “$cbr_ (0) start”
```

Cette entrée crée une connexion UDP/CBR du nœud 2 au nœud 3 à l’instant 82.557023746220864.

5.2.6. Le traitement du fichier trace et l’analyse des résultats :

Dans l’exemple précédent, nous aurons à la fin de la simulation un fichier trace appelé « simple.tr ».

Il existe deux types de fichiers trace :

- Le « standard format » : c’est le format par défaut ;
- Le « revised format ».

Le « revised format » est le format généré par ns si nous ajoutons la commande au script tcl.

```
$ns use-newtrace
```

Une entrée du fichier trace en « revised format » est de cette forme :

```
s –t 0.267662078 –Hs 0  -Hd 1 -Ni 0 6nx 5.00 –Ny 2.00  –Nz  0.00  -Ne

-1.000000 –N1 RTR –Nw --- -Ma 0 –Md 0 –Ms 0 –Mt 0 –Is 0.255 –Id -1.255 –It

message  -Il 32 –If 0 –Ii 0 –Iv 32
```

Chaque entrée correspond à une action effectuée par un nœud (à travers un agent (c~4) ou le routeur (c~3) :

-s : action (send),

-t : temps,

-N1: trace level (AGT,RTR,MAC),

-It: packet type (tcp, ack, message),

-Ii: id unique,

Nous pouvons trouver tous les détails du fichier trace dans « ns Manual ». Le traitement de ce fichier nécessite l'utilisation d'un langage de manipulation de fichier tel que AWK.

5.2.7. Conclusion :

Comme tout outil de simulation, ns2 étudie l'existant, la conception, la validation et l'évaluation des performances et ceci dans le domaine des protocoles et mécanismes réseau. C'est un simulateur extrêmement flexible. Ns2 permet l'étude des cas difficiles à reproduire dans la réalité. Il offre la souplesse pour pouvoir varier aisément une multitude de paramètres du réseau, ce qui n'est pas le cas si l'on simule sur un réseau réel, ns2 permet un gain de temps et d'argent.

D'un autre côté, ns2 offre des inconvénients des simulateurs tels que la simplification et l'abstraction du monde réel, la difficulté de tester certains aspects, et la puissance de calculs requise qui croît exceptionnellement avec la complexité du système simulé. De plus, ns2 souffre de l'état primitif aussi bien de ces outils de collecte et traitement des résultats, que de son interface graphique. Les concepteurs de ns2 ne manquent pas de mettre en garde les utilisateurs sur l'aspect non achevé de ce simulateur. Il ne s'agit pas d'un produit fini, de nouveaux bogues sont souvent trouvés et d'autres constamment introduits par les utilisateurs programmeurs. De plus, on remarque parfois des problèmes d'incompatibilité entre certaines implémentations de ns et certaines versions de Linux.

CONCLUSION

Le protocole MPLS s'applique dans les réseaux informatiques et de télécommunication. Il offre, par rapport au protocole IP, les avantages du mode connecté tout en conservant la souplesse du mode non connecté. D'une part, cette technologie diminue le temps de transit dans les réseaux. D'autre part, l'ingénierie de trafic offre une haute disponibilité dans un réseau IP. De plus, MPLS prévoit les éventuelles congestions dans le réseau. Il concrétise également la mise en œuvre de la QoS et attribue un service voix sur un réseau IP. Finalement, MPLS se voit faire une gestion souple des VPN.

En outre, les réseaux se constituent d'un empilement de protocoles ayant chacun leur protocole de signalisation et d'administration ainsi que des techniques de configuration spécifiques. MPLS optimise les performances du réseau et simplifie la gestion de la bande passante, en s'insérant entre IP et le protocole de transport de niveau 2. Toutefois, une redéfinition de l'architecture globale facilitera l'administration globale de G-MPLS qui étend le domaine jusqu'au niveau optique, en correspondant un label à une longueur d'onde.

Actuellement, les giga-routeurs et les Téra-routeurs concurrencent MPLS au niveau de la diminution du temps de transit.

ANNEXES

ANNEXE 1: DATAGRAMME IP

Définition du datagramme :

C'est un groupe logique d'information transmis sans qu'il ait été au préalable un circuit virtuel. La transmission n'est pas garantie, elle est dite *best effort*.

Définition du datagramme IP :

Un datagramme IP peut contenir un segment CP, un message ICMP, ARP, RARP ou encore OSPF. Les différents champs de l'en-tête IP sont alignés sur des mots de 32 bits, si aucune option n'est invoquée cet en-tête comporte 20 octets. Les bits sont représentés dans l'ordre dans l'ordre d'émission sur le support.

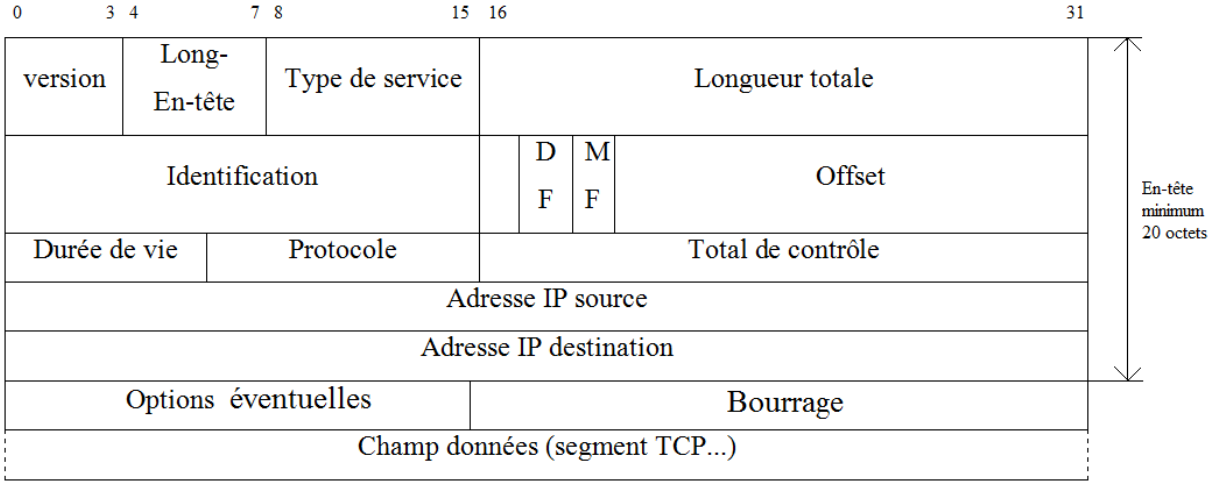

Figure A1.01 : Structure du datagramme IP

Le numéro de version sur 4 bits permet d'identifier la version du protocole IP utilisé. La présence de cette information autorise la cohabitation de plusieurs versions du protocole IP. La version courante est la version 4, cependant la version 6 (IPv6) est en cours de déploiement dans l'Internet.

Le champ longueur d'en-tête sur 4 bits (IHL) indique, en multiple de mots de 32 bits, la longueur de l'en-tête. La valeur courante, lorsqu'aucune option n'est invoquée, est 5 (20 octets).

ANNEXE 2: MODELE OSI

Contrairement aux technologies du téléphone, qui ont été dans la plupart des cas développées par une seule société, et qui n'était pas spécialement prévues pour être compatibles avec les systèmes des autres sociétés. Les technologies pour les réseaux de données ont été mises au point par des groupements d'entreprises, d'organisations de standardisation et de gouvernements. Elles ont été conçues pour être compatibles non seulement avec les équipements des différents fabricants de matériels réseaux, mais aussi avec d'autres systèmes, allant des ordinateurs aux *mainframes*, et avec toute sorte de système d'exploitation et d'applications. Pour atteindre cela, il est indispensable d'établir des standards solides. Bien que ce soit, de l'IEEE qu'émanent la majorité des spécifications pour les technologies de communication de données. Ces standards furent créés en suivant les directives du mode OSI composé de sept couches :

- Couche 7 : Application
- Couche 6 : Présentation
- Couche 5 : Session
- Couche 4 : Transport
- Couche 3 : Réseau
- Couche 2 : Liaisons de données
- Couche 1 : Physique

L'organisation de cette liste est capitale pour comprendre la plupart des réseaux de communication modernes. Elle devrait aussi être apprise par cœur par celui qui a l'intention d'étudier les réseaux informatiques.

Les couches 5 à 7 sont parfois regroupées sous l'appellation commune de *couches supérieures.* Généralement les ordinateurs hôtes sont les seuls appareils à intervenir sur les couches 4 à 7 du modèles OSI. Bien que TCP/IP puisse être expliqué grâce au modèle OSI, dans la pratique TCP/IP lui est très souvent comparé.

ANNEXE 3 : CELLULES ATM

Les paquets ATM possèdent une taille fixe et son alors appelés *cellules.* Chaque cellule est constituée d'un en-tête de 5 octets suivi par 48 octets de données. Ce type de paquets facilite le routage et augmente son efficacité. Les appareils d'un réseau ATM utilisent le système d'adressage NSAP ou bien *Network Service Access Point*. Ce standard ISO avait à l'origine été conçu pour la pile de protocoles OSI. La longueur d'une adresse NSAP peut varier entre 7 et 20 octets, ce qui veut dire qu'une adresse NSAP peut être plus longue qu'une adresse IPv6, dont la taille est fixée à 16 octets, soit 128 bits.

Pour plus de flexibilité dans l'attribution des adresses, l'ISO attribue un identifiant unique, nommé APl ou *Authority and Format Indicator* à des autorités de nommage chargées d'un pays ou d'une région particulière. Cet identifiant indique le nom de l'autorité ainsi que le format des autres éléments de l'adresse NSAP. L'autorité de nommage divise son espace d'adresses en différents domaines. Chaque domaine d'adresses possède une plage d'adresses qu'il peut utiliser pour le champ IDI ou *Initial Domain Identifier* de l'adresse NSAP. L'IDP ou *Initial Domain Part* est composé des champs AFI et IDI et identifie un domaine d'adresses de façon unique sur l'ensemble de l'espace de noms ATM.

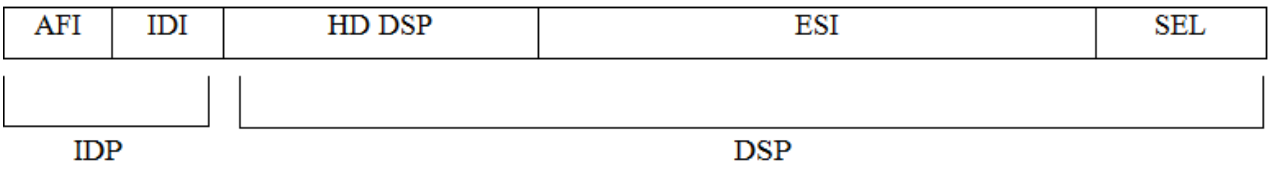

Figure A3.01 : format d'une adresse NSAP

Le champ IDI d'une adresse NSAP peut être de 3 formats différents :

- E.164 : utilisées dans les réseaux téléphoniques ;
- ICD : prévues pour les réseaux privés ;
- DCC : prévues également pour les réseaux privés.

ANNEXE 4 : PROTOCOLES DE ROUTAGE

RIP

C’est du type vecteur distance, RIUP est le premier protocole interne. Il est utilisé dans la communauté Internet, il est aujourd’hui remplacé par OSPF. Malgré une convergence lente et un trafic de gestion important, RIP reste le protocole de routage le plus employé.

OSPF

C’est un protocole interne à état des liens utilisés dans Internet. Les informations d’états sont diffusées sur une adresse de multicast réservée à OSPF pour pouvoir éviter l’inondation.

IS-IS

C’est le protocole de routage interne de l’ISO. C’est aussi un protocole à état des liens.

IGRP

C’est un protocole propriétaire de la société Cisco du type vecteur distance. Cependant, IGRP utilise une métrique construite qui prend en compte le délai d’acheminement, le débit, la fiabilité, la charge, du réseau et le MTU (Maximum Transfert Unit).

EIGRP

Le protocole EIGRP est un protocole de routage propriétaire développé par Cisco qui est basé sur le protocole IGRP. L’EIGRP offre une convergence plus rapide, une évolutivité améliorée et un traitement plus efficace des boucles de routage.

BGP

C’est un protocole qui définit les échanges à l’intérieur du domaine (iBGP) et entre systèmes de bordure (eBGP).

ANNEXE 5: SCRIPT TCL DE « SIMULATION D'UN RESEAU AD HOC »

```
set val (chan)     Channel/WirelessChannel

set val (prop)     Propagation/TwoRayGround

set val(netif)     Phy/WirelessPhy

set val(mac)       Mac/802 11

set val (ifq)      Queue/DropTail/PriQueue

set val (11)       LL

set val (ant)      Antenna/OmniAntenna

set val (x)        500

set val (y)        500

set val(ifqlen)    50

set val(seed)      0.0

set val (adhocRouting)   DSR

set val(nn)        10

set val(cp)        "./scen traff 100"

set val (se)       "/scen mvt 100"

set val (stop)     200.0

set ns    [new Simulator]

# setup topography object set topo     [new Topography]

# create trace object for ns and nam

set tracefd      [open wirelessl-out.tr w]

set namtrace       [open wirelessl-out.nara w]
```

```
$ns  trace-ail $tracefd

$ns  namtrace-all-wireless $namtrace $val(x) $val(y)

Stopo load flatgrid $val(x) $val(y)

set god  [create-god $val(nn)]

ttglobal node setting

set chan 1  [new $val(chan)]

$ns  node-config        -adhocRouting $val{adhocRouting) \
                        -llType $val(ll) \
                        -macType $val(mac) \
                        -ifqType $val(ifq) \
                        -ifqLen $val(ifqlen) \
                        -antType $val(ant) \
                        -propType $val(prop) \
                        -phyType $val(netif) \
                        -channel $chan 1 \
                        -topoinstance $topo \
                        -agentTrace ON \
                        -routerTrace ON \
                        -macTrace OFF
                        -movernentTrace OFF

for {set i 0} {$i < $val(nn) } {incr i} { set node ($i) [$ns  node]

 puts "Loading connection pattern..."
```

```
source $val(cp)
puts "Loading scénario file..."
source $val(se)
for (set i 0} {$i < $val(nn) } {incr i} { $ns  at $val(stop).0 "$node ($i) reset"

$ns  at $val(stop).0001 "stop"
$ns  at  $val(stop).2 "puts \"NS EXITING...\"  ; $ns
hait"
proc stop {} {
        global ns  tracefd
        global ns  namtrace
        $ns  flush-trace
        close $tracefd
        close $namtrace
        exec nam wirelessl-out.nam &
        exit 0

puts "Starting Simulation..."

$ns  run
```

BIBLIOGRAPHIE

[1] P.-A. Goupille : *Technologie des ordinateurs et des réseaux,* Dunod : Paris, 2000.

[2] R. Parfait : *Les réseaux des télécommunications,* Lavoisier : Paris, Hermès Science Publication, 2002.

[3] C. Servin, *Réseaux et Télécoms,* DUNOD, Paris, 2003

[4] Cisco Systems, *Implementing Cisco MPLS*, version 2.0, 2003.

[5] Girard, R. Page, *Dimensioning of telephone networks with non hierarchical routing and trunk reservation*, Network Planning Symposium, vol.3, p. 228-236, Juin 1986.

[6] J. Guichard, I. Pepelnjak, *MPLS and VPN architecture*, Edition Cisco Press, 31 Octobre 2000.

[7] Girard, *Routing and Dimensioning in Circuit Switched Networks,*Addison-Wesley Publishing Company, 1990.

[8] P.G. Harrisoon, N.M. Patel, *Performance Modeling of Communication Networks and Computer Architectures,* International Computer Science, Addison-Wesley, 1993.

[9] L. Kleinrock, *On the modeling and analysis of computer networks",* Proceedings of the IEEE, vol.81, n°8, p.1179-1191, août 1993.

[10] L. Kleinrock, *Queueing Systems,* vol.1 et 2, John Wiley et Sons, 2001.

[11] A.M. Law, W.Delton, Simulation, *modeling and analysis,* McGraw-Hill, 2ème edition, 1991.

[12] J.-L. Melin, *Qualité de service sur IP*, Eyrolles, 2002.

[13] D. Ros, R. Marie, *Los characterisation in high speed networks trough simulation of fluid models,* Telecommunication Systems, février 2001.

[14] M. Schwartz, *Telecommunication Networks-Protocols, Modelling and Analysis,* Addison-Wesley Publishing Company, 1987.

[15] L.Toutain, *Réseaux locaux et Internet,* Hermès, 2ème édition, 1999.

[16] UIT, Planitu, *Introduction à la théorie de base de télétrafic,* Planitu, Doc 19, F., 2000

[17] UIT, *Réseaux IP, tarification des services des télécommunications* , rapport final, janvier 2003.

[18] M. Becker et A.-L. Beylot, *Simulations des réseaux*, Lavoisier,2006

RENSEIGNEMENT SUR L'AUTEUR

Nom : RANDRIAMITANTSOA

Prénoms : Andry Auguste

Adresse : Lot III V 42 Anosizato Est

101 Antananarivo

Madagascar

E-mail : andriau_23@yahoo.fr

Titre du mémoire :

PLANIFICATION, MODELISATION ET SIMULATION DES RESEAUX PAR LE PROTOCOLE MPLS

Nombre de page : 94

Nombre de tableaux : 1

Nombre de figures : 26

Mots clés : routage, multiprotocole, réseaux, simulation, ns-2, dimensionnement, modélisation.

RESUME

Actuellement, les réseaux de communications ne cessent d'évoluer dans le monde, les besoins en matière de nouvelles technologies incitent les chercheurs à se mettre en phase. MPLS en est un fruit sur le domaine de réseau de télécommunication. Elle offre une qualité de service important. En effet, elle diminue le temps de transit accorde une haute disponibilité dans le réseau en question. De plus, elle évite les éventuelles congestions et offre un service voix sur un réseau IP. Toutefois le protocole MPLS participe à une gestion souple des VPN. Par ailleurs, quelques outils de simulation donnent une possibilité de plusieurs performances de réseaux. Finalement, une simulation particulière de MPLS sous le simulateur NS-2 nous démontre ses avantages évidents.

ABSTRACT

Nowadays, the communication networks do not cease evolving in the world; the needs as regards new technologies encourage the researchers to be put in phase. MPLS is one product on the field of telecommunications network. It offers a quality of significant service. Indeed, it decreases the time of transit grants a high availability in this network. Moreover, it avoids the possible congestions and offers a service voice on a network IP. However, protocol MPLS takes part in a flexible management of the VPN. In addition, some tools for simulation give a possibility of several performances of networks. Finally, a particular simulation of MPLS with NS-2 simulator shows us its obvious advantages.