Ministère de l'Enseignement Supérieur et de Recherche Scientifique
UNIVERSITE KASDI-MERBAH OUARGLA
Faculté des Nouvelles Technologies de l'Information et de la Communication
Département d'Informatique et des Technologies de l'Information

Mémoire de Master Professionnel

Domaine : Informatique et Technologie de l'Information

Filière : Informatique

Spécialité : Administration et Sécurité des réseaux

Présentée par :

Mohammed Nafaa GHANIA

Mohamed Redha KORICHI

Thème :

Vers algorithme optimal pour la gestion de la congestion dans les réseaux MPLS

Soutenue le : 02 juillet 2019
Devant le jury composé de :

Dr. Benmir Abdetkader	Université de Ouargla	Président du jury
Dr. Korichi Ahmed	Université de Ouargla	Superviseur
Dr. Djelaili Karima	Université de Ouargla	Membre du jury

juillet 2019

الحمد والشكر

بسم الله والحمد لله رب العالمين وصلوات الله وسلامه على خاتم النبيين وإمام المرسلين سيدنا محمد صل الله عليه وسلم وعلى آهله وصحبه وسلم ، وعلى التابعين لهم ومن تبعهم بإحسان إلى يوم الدين .

الحمد لله الذي بحمده تتم النعم والصالحات و لك الحمد ربنا حتى ترضى، ولك الحمد إذا رضيت، ولك الحمد بعد الرضى ، اللهم إن نعمك كثيرة علينا لا نحصيها ولا نحصي ثناء عليك ولا نقدر وأنت سبحانك كما أثنيت على نفسك وأنت سبحانك غني عن العالمين.

سبحانك يا ربنا لك الحمد والشكر حمداً كثيراً طيباً مباركاً فيه. اللهم لك الحمد والشكر كما ينبغي لجلال وجهك وعظيم سلطانك وعلو مكانك اللهم لك الحمد والشكر مليء السموات والأرض وما بينهما ومليء ما شئت من بعد.

الحمد والشكر لله على ما أكرمنا به الله من نجاح وتميز في دراستنا والحمد لله على التوفيق والسداد لقوله تعالى : **{وَمَا تَوْفِيقِي إِلَّا بِاللَّهِ ۚ عَلَيْهِ تَوَكَّلْتُ وَإِلَيْهِ أُنِيبُ (88) }** سورة هود

فالحمد لله على ما يسر لنا في كتابة هذه المذكرة وإنجاز هذا العمل المتواضع. فـالحمد لله حمد الشاكرين ، والحمدلله في كل وقت وحين. الحمد لله حمداً على كل النعم.. والحمد لله على حمد النعم.. والحمد لله حمداً يليق برب النعم.. فيا ربنا لك الحمد إذا أعطيت، ولك الحمد إذا أخذت. الحمد لله ما انتهى درب ولا ختم جهد ولا تم سعي إلا بفضله الحمدُ لله على البلوغ ثم الحمدُ لله على التمام والحمدُ لله على الختام.

قال عز من قائل : **{قُلْ إِنَّ صَلَاتِي وَنُسُكِي وَمَحْيَايَ وَمَمَاتِي لِلَّهِ رَبِّ الْعَالَمِينَ (162) لَا شَرِيكَ لَهُ ۖ وَبِذَٰلِكَ أُمِرْتُ وَأَنَا أَوَّلُ الْمُسْلِمِينَ (163) }** سورة الأنعام

نسال الله الكريم دوام التقدم والتيسير اللهم ارزقنا نجاحاً في كل أمر، ونيلاً لكل مقصد، وارزقنا القمة في درجات العلم. اللهم اهدنا لما تحب و ترضى، اللهم هبنا علمًا ينتفع به، وعملًا يقربنا إليك، وهبنا صبرًا وثباتًا في الدين. اللهم اجعل شبابنا عبرة وقدوة للخير، اللهم قوي إيماننا وبإيماننا قوي عملنا، و بعملنا زد من نجاحاتنا. اللهم ارزقنا قوة الحفظ، وسرعة الفهم، وصفاء الذهن، اللهم ألهمنا الصواب في الجواب، وبلغنا أعلى المراتب في الدين والدنيا والآخرة، وحفظنا وأصلحنا وأصلح بنا الأمة. اللهم أخرجنا من ظلمات الدهر، وأكرمنا بنور الفهم، وافتح علينا بمعرفة العلم، وحسّن أخلاقنا بالحلم، وسهل لنا أبواب فضلك، وانشر لنا من خزائن رحمتك يا أرحم الراحمين. اللّهم إنّا نسألك التوفيق والهداية، والرشد والإعانة، والرضى والصيانة، والحب والإنابة، والدعاء والإجابــة، اللّهم ارزقنا نوراً في القلب، وزينة في الوجه، وقوة في العمل. اللهم إني أسألك الهدى والتّقى والعفاف والغنى. اللهمّ إنّا نسألك خير المسألة، وخير الدعاء، وخير النجاح، وخير العمل، وخير الثواب، وخير الحياة، وخير الممات، وثبّتنا، وثقّل موازيننا، وحقق إيماننا، وارفع درجاتنا، وتقبّل صلاتنا، واغفر خطيئتنا، ونسألك الدرجات العُلى من الجنة اللهم آمين يا رب العالمين.

وصل الله على سيدنا محمد وعلى آهله وصحبه وسلم ، سبحان ربك رب العزة عما يصفون وسلام على المرسلين والحمد لله رب العالمين.

Remerciements

Nous tenons à exprimer toute notre gratitude et notre reconnaissance envers Monsieur le Dr. Ahmed Korichi, Maitre de Conférences Habilité en informatique à l'Université de Ouargla, qui a dirigé ce travail, pour son soutien, pour la sympathie qu'il nous a témoignée et pour la liberté de recherche qu'il a bien voulue nous laisser,

Nous tenons à exprimer toute notre reconnaissance au doctorante Mme Khadîdja SAKHRI. Nous la remercions pour ses orientations, ses aides et ses conseils.

Nous remercions vivement Messieurs les membres du jury, de nous avoir fait l'honneur de jugés notre travail.

Nous voudrions exprimer ma reconnaissance envers les amis et collègues qui nous ont apporté leur support moral et intellectuel tout au long de nos démarches.

Nous ne pourrions terminer sans remercier nos parents, nos familles, ainsi que tous nos amis qui nous a encouragé à poursuivre mes études. Que nos parents tout particulièrement trouvent ici notre reconnaissance.

Mohammed Nafaa Ghania.

Mohamed Redha Korichi.

Résumé :

Le fait que nous ayons atteint aujourd'hui la vitesse d'accès à Internet, la diversité et la qualité des services constituent un progrès majeur dans nos vies. Chacun de ces services attire de nouveaux utilisateurs, augmente la quantité totale de données et augmente le nombre de flux de données à livrer. Cela a conduit à un problème majeur de congestion. Ce qui nécessite le besoin d'ingénierie du trafic (TE) et de la qualité de service (QoS). La principale caractéristique de l'ingénierie du trafic est de contrôler le routage pour optimiser l'utilisation des ressources et la performance du réseau, tout en garantissant la qualité de service. Il existe de nombreuses technologies prenant en charge l'ingénierie du trafic, les plus courantes étant MPLS, où cette dernière devient la technologie dominante pour la transmission des données.

La technique MPLS (**Multi-Protocol Lable Switching**) est particulièrement adaptée à l'ingénierie du trafic car elle repose sur différentes méthodes indépendantes de l'acheminement IP, elle est plus efficace et contribue à une distribution optimale du lecteur réseau et à une réduction des encombrements.

Dans ce mémoire, nous proposons un nouveau moyen de réduire l'encombrement des réseaux MPLS à l'aide du mécanisme d'équilibrage de charge. L'idée de base est que les LSP dévient efficacement des liens les plus occupés du réseau vers des liens non exploités. Nous utilisons le concept de seuil pour déterminer la congestion, considérons la priorité du flux pour accélérer le temps élevé de transmission du flux de priorité, réduire le délai d'une partie à une autre et restreindre la capacité disponible pour déterminer le chemin de déviation approprié. Enfin, nous utilisons des simulations OMNeT ++ pour valider notre approche et démontrer son efficacité.

Mots-clés: Réseau MPLS; Ingénierie de trafic MPLS (TE) ; Chemin dévié ; L'équilibrage de charge ; Seuil.

Abstract:

The fact that we have reached today the speed of access to the Internet, the diversity and quality of services is a major advance in our lives. Each of these services attracts new users, increases the total amount of data and increases the number of data streams to be delivered. This led to a major problem of congestion. This requires the need for Traffic Engineering (TE) and Quality of Service (QoS). The main feature of traffic engineering is to control routing to optimize resource utilization and network performance, while ensuring quality of service. There are many technologies that support traffic engineering, the most common being MPLS, where the latter becomes the dominant technology for data transmission.

MPLS (**Multi-Protocol Lable Switching**) is particularly suited to traffic engineering because it relies on different methods independent of IP routing, is more efficient, and contributes to optimal distribution of the network drive and reduced congestion.

In this memory, we propose a new approach to reduce congestion in MPLS networks by using a load balancing mechanism. The key idea is to efficiently deviate the LSPs from the most congested links in the network to the underutilized ones. We use the concept of threshold to determine the congestion, and consider the flow priority to speed up the high transmission time of the priority flow, minimize the end-to-end delay, and the free capacity constraint to select the adequate deviation path. Also used simulation experiments to validate our approach and show its efficacy.

Keywords*:* MPLS Network; MPLS Traffic Engineering (TE); deviated path; Load Balancing; threshold.

ملخص :

أن ما وصلنا إليه اليوم من سرعة الوصول إلى الانترنت وتنوع وجودة الخدمات يعتبر تقدما كبيرا في حياتنا .وكل من هذه الخدمات تجذب مستخدمين جدد ، وتزيد من إجمالي كمية البيانات ، وتزيد من عدد تدفقات البيانات التي سيتم تقديمها. وقد أدى هذا إلى مشكلة كبيرة من الازدحام .الأمر الذي يتطلب الحاجة إلى هندسة المرور (TE) ، وجودة الخدمة (QoS). الميزة الرئيسية للهندسة المرورية هي التحكم في التوجيه لتحسين استخدام الموارد وأداء الشبكة ، مع ضمان جودة الخدمة. هناك العديد من التقنيات التي تدعم هندسة المرور ، وأكثرها شيوعًا هي MPLS ، حيث تصبح هذه الأخيرة التكنولوجيا المهيمنة لنقل البيانات.

اقتصرت هذه الأطروحة على الشبكات MPLS , تعد تقنية MPLS مناسبة بشكل خاص للهندسة المرورية لأنها تعتمد على طرق مختلفة ، مستقلة عن توجيه IP ، وهي أكثر كفاءة ، فهي تساهم في التوزيع الأمثل لحملة الشبكة وتقلل من الازدحام.

في هذه المذكرة، نقترح طريقة جديدة لتقليل الازدحام في شبكات MPLS باستخدام آلية موازنة التحميل. الفكرة الأساسية هي أن تنحرف LSPs بكفاءة عن الروابط الأكثر ازدحامًا في الشبكة إلى الروابط غير المستغلة. لتحديد الازدحام نستخدم مفهوم العتبة ، إضافة إلى أولوية التدفق لتسريع وقت الإرسال، وتقليل التأخير من طرف إلى آخر ، وتقييد السعة الحرة لتحديد مسار الانحراف المناسب، كما نستخدم تجارب المحاكاة ببرنامج OMNeT++ للتحقق من صحة نهجنا وإظهار فعاليته.

الكلمات المفتاحية: شبكة MPLS ؛ هندسة المرور TE-MPLS ؛ المسار المنحرف ؛ موازنة التحميل ؛ العتبة.

Table des matières

Liste des figures

Liste des tables

Liste des acronyms

TE	Traffic engineering
QoS	Quality of Service
MPLS	Multi-Protocol Lable Switching
VPN	Virtuel Private Network
VoIP	Voix sur IP
LSR	Label Switch Router
LER	Label Edge Router
LSP	Label Switch Path
FEC	Forwarding Equivalence Classes
OSI	Open Systems Interconnection
BGP	Border Gateway Protocol
FIB	Forwarding Information Base
ATM	Asynchronous Transfer Mode
LDP	Label Distribution Protocol
RSVP	Resource Reservation Protocol
MP-BGP	Multi-Protocol Border Gateway Protocol
IGP	Interior Gateway Protocol
DiffServ	Differentiated Services
OSPF	Open Shortest Path First
IS-IS	System intermediaries system
CBR	Constraint-Based Routing
CSPF	Constrained Shortest Path First
CoS	Class of service
IETF	Internet Engineering Task Force
GMPLS	Generalized MPLS
IP	Internet Protocol
CEF	Cisco Express Forwarding
TSLB	Topology-Based Static Load-Balancing Algorithm
IEEE	Institute of Electrical and Electronics Engineers
IntServ	Integrated Services
RSLB	Resource-Based Static Load-Balancing Algorithm
ISO	International Organization for Standardization

MIRA	Minimum Interference Routing Algorithm
LMIR	Light Interference Routing
LTE	Long Term Evolution
DYLBA	Dynamic Load Balancing Algorithm
ODLB	Optimal Dynamic Load Balance Algorithm
MATE	Adaptive Trafic Engineering
DORA	Dynamic Online Routing Algorithm
DEPR	Distributed Explicit Partial Rerouting
LIOA	Least Interference Optimization Algorithm
MHA	Minimum Hop Algorithm
QTA	Queue Tuning Algorithm
PPBS	Path-based Bandwidth Scheme
OMNet++	Objective Modular Network Testbed in C++
FBLB	Feedback Based Load Balancing
LBWDP	Load Balancing over Widest Disjoint Paths
PER	Prediction of Effective Repartition
PEMS	PEriodic Multi-Step Routing Algorithm
EF	Expedited Forwarding
AF	Assured Forwarding
BEF	Best-Effort Forwarding
LBDP	Load Balancing Algorithm using Deviation Path
UB	Bande passante utilisée
FC	Free Capacity
NED	NEtwork Descrption
MAC	Medium Access Control
MANET	Mobile Adhoc Network
LIB	Label Information Base
DBNET	Data Base NETwork
LBMPLS	Load Balancing Multi-Protocol Lable Switching.

Chapitre 1 Introduction Générale

Contenu

1.1. Introduction

Aujourd'hui, le développement technologique et la croissance rapide de l'Internet auxquels le monde, en termes de rapidité et de disponibilité de nouveaux services, tels que les applications temps réel (audio/vidéo) et d'autres, entraînant à une augmentation considérable dans la charge du réseau et le volume du trafic, en augmentant ainsi le problème de congestion; ce qui nécessite le besoin d'ingénierie du trafic (Traffic engineering (TE)), et de qualité de service (QoS). La principale caractéristique de l'ingénierie du trafic est le contrôle du routage visant à améliorer l'utilisation des ressources et les performances du réseau, tout en garantissant la qualité de service. Il existe de nombreuses techniques prenant en charge l'ingénierie du trafic, la plus courante étant la technologie MPLS, cette dernière est devenue la technologie dominante pour la transmission de données.

La technologie MPLS est particulièrement bien adaptée à l'ingénierie du trafic car elle est basée sur différents itinéraires, indépendamment du routage IP, elle est plus efficace; elle contribue à une répartition optimale de la charge du réseau et réduit la congestion.

La congestion se manifeste généralement en deux scénarios: **[1]**

- 1 : lorsque les ressources du réseau sont insuffisantes ou inadéquates pour prendre en charge la charge proposée.
- 2 : lorsque les flux du trafic sont mappés de manière inefficace sur les ressources disponibles; entraînant à une surutilisation des sous-ensembles de ressources réseau, tandis que d'autres restent sous-utilisés.

Le premier type de problème de congestion peut être résolu en augmentant la capacité du réseau, ce qui est extrêmement coûteux. Le second type de problèmes de congestion, à savoir ceux résultant d'une répartition déséquilibrée du trafic réseau, peut être résolu en adoptant des règles d'équilibrage de charge. Par conséquent, la question la plus cruciale consiste à équilibrer le trafic réseau tout en maintenant une allocation efficace des ressources et à améliorer les performances du réseau et la qualité de service Internet.

Dans ce contexte, l'objectif principal de ce travail est de proposer un nouvel algorithme optimal pour la gestion de la congestion dans les réseaux MPLS. Dans ce contexte, nous avons étudié les avantages des technologies et des réseaux MPLS et leur rôle dans l'ingénierie du trafic pour réduire la congestion, en proposant une nouvelle approche de gestion du flux en utilisant le mécanisme Load Balancing qui est basé sur des approches qui ont été étudiées dans la littérature et de prendre

en considération les problèmes de recherche et des conclusions qui sont apparues dans nos enquêtes.

1.2. Tendance technologique

Le fait que nous avons atteint aujourd'hui la vitesse d'accès à Internet, la diversité et la qualité des services ce sont un progrès majeur dans nos vies. Mais avec cette croissance rapide de l'Internet, il est nécessaire de créer des réseaux plus grands, plus rapides et plus sûrs. La plupart des réseaux modernes visent à fournir une variété de services sur une infrastructure unique qui transfère tous les types de données, et la gestion de tels réseaux est devenue très importante, ce qui entraîne à une utilisation de la technologie MPLS, qui suscite actuellement beaucoup d'attention.

Les réseaux MPLS ont des fonctionnalités diverses, bien qu'ils fonctionnent sur les réseaux IP, mais ils supportent de nombreux autres protocoles pour une intégration facile. Ils garantissent également la sécurité et la qualité des services (QoS), et supportent les réseaux privés virtuels (Virtuel Privat Network (VPN)); ainsi que le cryptage et l'authentification, sans oublier l'ingénierie du trafic, qui est la principale caractéristique de MPLS. Ce dernier (TE) permet de contrôler le transfert de données, d'utiliser de manière optimale les ressources et d'améliorer les performances du réseau en éliminant la congestion.

1.3. Contexte du mémoire

Nous présentons dans ce chapitre une vue générale sur le contexte de ce mémoire qui porte sur les réseaux MPLS. Nous reformulons précisément ce domaine en indiquant ce qui nous aidera dans notre étude et le but de notre travail qui est représenté dans la spécification d'un nouvel algorithme optimal pour la gestion de la congestion dans les réseaux MPLS.

Nous y présentons une certaine terminologie, les caractéristiques et les exigences globales de ces réseaux, les applications qui peuvent être envisagées dans ces réseaux en plus de quelques travaux de normalisation et de standardisation lancés afin de répondre aux spécificités des réseaux MPLS.

1.4. Les réseaux MPLS

Dans le routage IP sur Internet, chaque routeur doit prendre une décision de transfert indépendante pour chaque paquet en se basant uniquement sur l'en-tête de la couche réseau du paquet. Ainsi, chaque fois qu'un paquet arrive sur un routeur, ce dernier doit «réfléchir» au prochain envoi du paquet. Pour ce faire, le routeur se réfère à des tables de routage complexes. Le processus est répété à chaque saut au long de la route jusqu'à ce que le paquet atteigne finalement sa destination. Tous ces sauts et toutes ces décisions de routage individuelles entraînent à des performances médiocres pour des applications urgentes telles que la vidéoconférence ou la voix sur IP (VoIP).

MPLS a été créé à la fin des années 90 pour offrir une alternative plus efficace au routage IP traditionnel. Multi-protocole Label Switching (MPLS) résout ce problème en établissant des itinéraires prédéterminés et hautement efficaces.

Contrairement à ce que l'on pourrait penser, Multi-protocole Lable Switching (MPLS) n'est pas un type de connexion Internet (comme une fibre ou un Internet lié), mais plutôt une "technique". Parmi les définitions MPLS, on trouve :

1 - Search Enterprise WAN définit MPLS comme «un protocole pour accélérer et façonner les flux de trafic réseau», ce qui signifie essentiellement que MPLS trie et hiérarchise vos paquets de données en fonction de leur classe de service (par exemple, données de téléphone IP, vidéo ou Skype). **[2]**

2 - Multi-protocole Label Switching (MPLS) est un mécanisme utilisé dans les infrastructures de réseau informatique pour accélérer le temps nécessaire à la transmission d'un paquet de données d'un noeud à un autre. Il permet aux réseaux informatiques d'être plus rapides et plus faciles à gérer en utilisant des étiquettes de chemin court au lieu d'adresses réseau longues pour le routage des paquets réseau. **[3]**

3 - Multi-protocole Label Switching (MPLS) est une technique de routage indépendante du protocole IP, conçue pour accélérer et façonner les flux de trafic sur les réseaux de grandes entreprises et de fournisseurs de services. MPLS permet de transmettre la plupart des paquets de données à la couche 2 - le niveau de commutation - plutôt que de devoir être transmise à la couche 3 - le niveau de routage. **[4]**

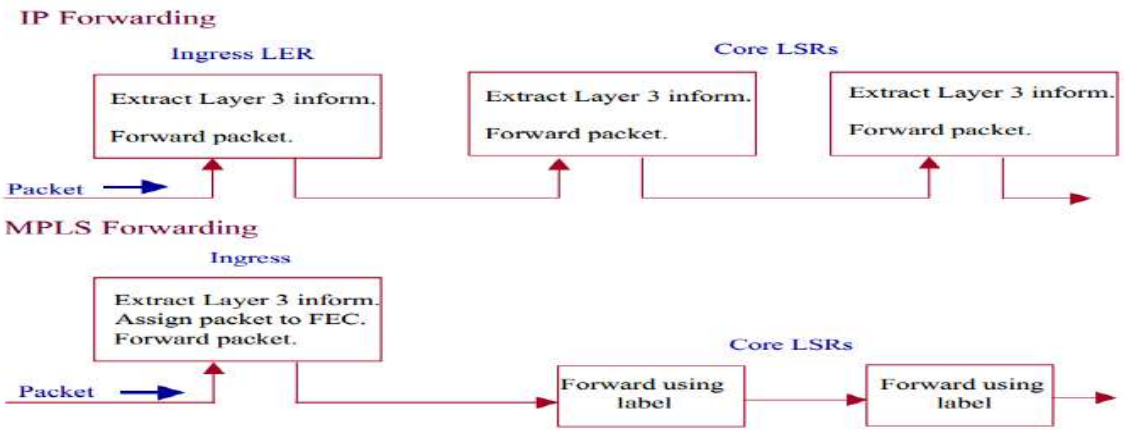

Figure 1.1 : Comparaison des techniques de transmission IP et MPLS

- Dans un réseau MPLS, le tout premier routeur reçoit un paquet qui détermine l'intégralité de la route du paquet, dont l'identité est rapidement transmise aux routeurs suivants à l'aide d'une étiquette qui se trouve dans l'en-tête du paquet. Bien que le matériel de routeur soit amélioré de façon exponentielle depuis le développement de MPLS (ce qui a quelque peu réduit son importance en tant que technologie de gestion du trafic plus efficace), il reste important et populaire en raison de ses nombreux autres avantages, notamment la sécurité, la flexibilité et l'ingénierie du trafic.

- Avec MPLS, la première fois qu'un paquet entre dans le réseau, il est affecté à une classe d'équivalence de transmission (FEC) spécifique, indiquée par l'ajout d'une séquence de bits courte (l'étiquette) au paquet. Chaque routeur du réseau dispose d'un tableau indiquant comment gérer les paquets d'un type spécifique de FEC. Ainsi, une fois que le paquet est entré sur le réseau, les routeurs n'ont plus besoin d'effectuer une analyse d'en-tête.

  Au lieu de cela, les routeurs suivants utilisent l'étiquette en tant qu'index dans une table qui leur fournit un nouveau FEC pour ce paquet.

- Cela donne au réseau MPLS la possibilité de gérer de manière cohérente des paquets qui présentent des caractéristiques particulières (provenant par exemple de ports ou transportant du trafic de types d'applications particuliers). Les paquets acheminant du trafic en temps réel, tels que la voix ou la vidéo, peuvent facilement être mappés sur des itinéraires à faible latence sur le réseau, ce qui est difficile avec le routage classique. L'essentiel de l'architecture réside dans le fait que les étiquettes fournissent un moyen d'attacher des informations supplémentaires à chaque paquet, informations qui vont bien au-delà de ce que les routeurs avaient auparavant.

1.4.1. Exigences globales

En plus des spécificités du réseau MPLS, Il est nécessaire de mentionner les concepts et composantes les plus importants, à savoir:

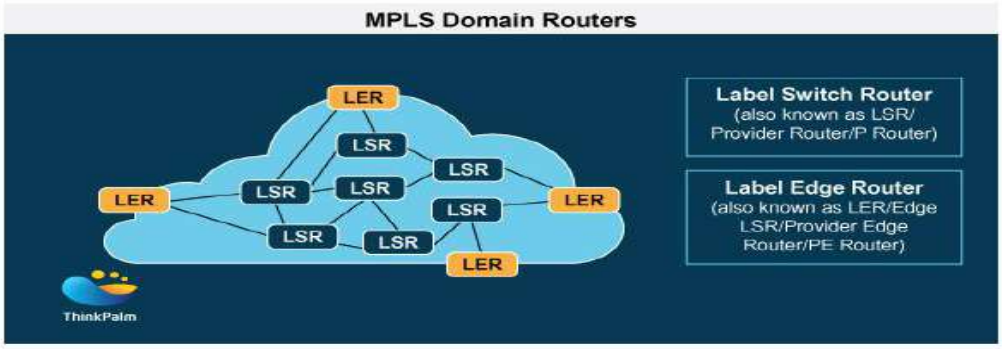

Figure 1.2 : Les Routeurs de domaine MPLS

Figure 1.3 : Un domaine MPLS

- **Label Switch Router (LSR)** : est un dispositif capable de transmettre des paquets à la couche 3 et de transmettre des trames qui encapsulent le paquet à la couche 2. Le mécanisme d'échange d'étiquettes est implémenté à la couche 2. **[5]**
- **Label Edge Router (LER)** : est à la fois un routeur et un commutateur de couche 2 capable de transférer des trames MPLS vers et depuis un domaine MPLS. Il établit la liaison FEC IP vers MPLS, y compris l'agrégation des flux entrants. Il communique également avec les LSR MPLS intérieurs pour échanger des reliures d'étiquettes. Souvent désigné sous le nom de LSR d'entrée ou de sortie, car il est situé à la périphérie d'un domaine MPLS. **[5]**
- **Label Switch Path (LSP)** : est un chemin commuté d'entrée-sortie construit par les nœuds MPLS pour transmettre les paquets encapsulés MPLS d'une FEC particulière à l'aide du mécanisme de transfert d'échange d'étiquettes. Il est similaire au concept de canaux virtuels dans un contexte ATM. **[5]**
- **Forwarding Equivalence Classes (FEC)** : est un ensemble de paquets traités de manière identique par un routeur, c'est-à-dire transmis par la même interface avec les mêmes sauts et étiquettes, et affectés à la même classe de service. Lorsqu'un paquet entre dans le domaine MPLS au niveau du nœud d'entrée, il est mappé dans une FEC. Le mappage peut être effectué en fonction d'un certain nombre de facteurs, à savoir le préfixe d'adresse, la paire d'adresses source / destination ou l'interface d'entrée. Au moment actuel, il existe trois éléments FEC définis, un préfixe d'adresse, un identifiant de routeur et un flux (port source / de destination et adresses IP). Un groupe de paquets IP qui sont transmis sur le même chemin et traités de la même manière et qui peuvent être mappés sur une seule étiquette par un LSR, comme la figure suivante : **[5]**

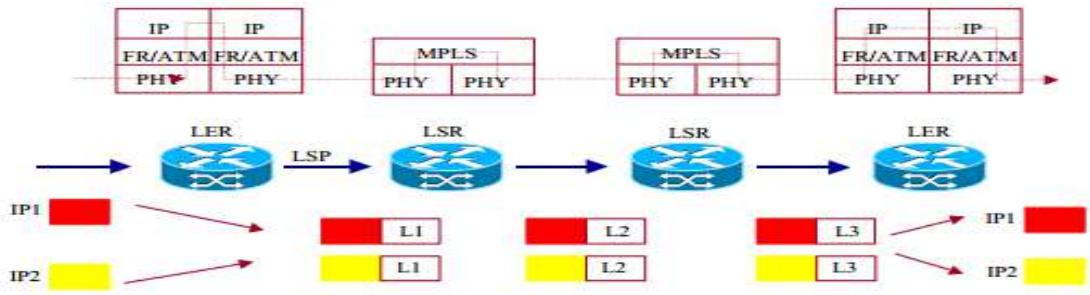

Figure 1.4: Forwarding Equivalence Classes

- **Label** : est un identifiant court, de longueur fixe et localement significatif utilisé pour identifier une FEC. Un paquet peut être attribué à une FEC en fonction de l'adresse de destination de la couche réseau; Toutefois, Lable ne code directement aucune information provenant de l'en-tête de la couche réseau. Un paquet label est un paquet dans lequel une étiquette a é té codée. **[5]**

  L'en-tête MPLS 32 bits contient les champs suivants: **[6]**

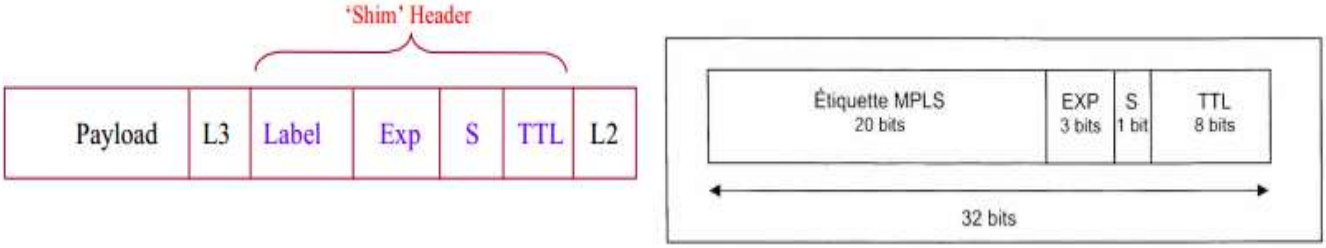

Figure 1.5 : Contenu de l'entête MPLS

- Label value (20 bits) : (Label) contient toutes les informations pour les routeurs MPLS afin de déterminer où le paquet doit être transmis.
- Expérimental (3 bits): (Exp) les bits expérimentaux sont utilisés pour que la qualité de service (QoS) définisse la priorité que doit avoir le paquet, puisqu'il ne s'agit que de 3 bits, les valeurs disponibles vont de 0 à 7 en fonction de la qualité de service, telles que la priorité IP, la classe de service ou la qualité de service pour déterminer le type de service offert aux paquets.
- Bottom-of-Stack (1 bit): (S) indique au routeur MPLS s'il s'agit de la dernière étape du voyage et qu'il n'y a plus Labels à prendre en compte. Cela signifie généralement que le routeur est un routeur de sortie. Définir s'il s'agit "1" du dernier en-tête MPLS.
- Time to live (8 bits): identifie le nombre de sauts que le paquet peut effectuer avant d'être rejeté. ce qui évite des boucles infinies.

1.4.2. Caractéristiques

- **Son Classement dans le modèle OSI :**

  Il y a beaucoup de confusion quant à savoir si MPLS est une couche 2 ou 3 service, mais MPLS ne correspond pas parfaitement dans la hiérarchie sept couches OSI, et est parfois classée comme couche 2.5. En fait, l'un des principaux avantages de MPLS est qu'il sépare les mécanismes de transmission du service de liaison de données. En d'autres termes, MPLS

peut être utilisé pour créer des tables de transfert pour n'importe quel protocole. Plus précisément, les routeurs MPLS établissent un chemin (LSP), un chemin prédéterminé pour acheminer le trafic dans un réseau MPLS, en fonction des critères de la FEC. Il est seulement après un LSP a été établi que la transmission MPLS peut se produire. Les LSP sont unidirectionnels, ce qui signifie que le trafic de retour est envoyé sur un autre LSP. Lorsqu'un utilisateur final envoie du trafic sur le réseau MPLS, une étiquette MPLS est ajoutée par un routeur d'entrée MPLS situé au bord du réseau.

- **Multi-protocole : [7]**

  Tout simplement, MPLS est indépendante du protocole principal. Il établit simplement sur le protocole L2 et fournit le réseau de transport rapide et efficace sur le réseau à commutation de paquets. Ainsi, dans la partie transport du réseau, MPLS diffère pour des raisons très importantes :

  1. Il correspond à n'importe quel protocole L2 ci-dessous

  2. Il vérifie le paquet IP des deux quand ils arrivent à la porte du réseau de transport et envoie les itinéraires prédéterminés les plus efficaces.

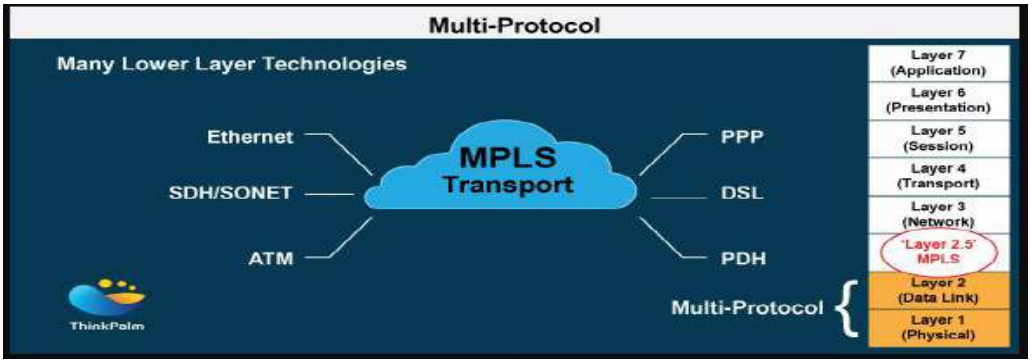

Figure 1.6 : Pourquoi multi-protocole

- **Un fonctionnement diffèrent du réseau : [4]**

Dans un réseau MPLS, chaque paquet est étiqueté lors de son entrée dans le réseau du fournisseur de services par le routeur d'entrée (LER). C'est également le routeur qui décide du LSP que le paquet prendra jusqu'à ce qu'il atteigne l'adresse de destination. Tous les routeurs (LSR) suivants effectuent un transfert de paquet basé uniquement sur ces Labels MPLS - ils ne cherchent jamais aussi loin que l'en-tête IP. Enfin, le routeur de sortie supprime les étiquettes et transmet le paquet IP d'origine vers sa destination finale. Lorsqu'un LSR reçoit un paquet, il effectue une ou plusieurs des actions suivantes:

• Push: ajoute Label. Cette opération est généralement effectuée par le routeur d'entrée.

• Swap: remplace Label. Ceci est généralement effectué par les LSR entre les routeurs d'entrée et de sortie.

• Pop: supprime Lable. Ceci est le plus souvent fait par le routeur de sortie.

Ce schéma illustre comment un simple réseau MPLS fonctionne.

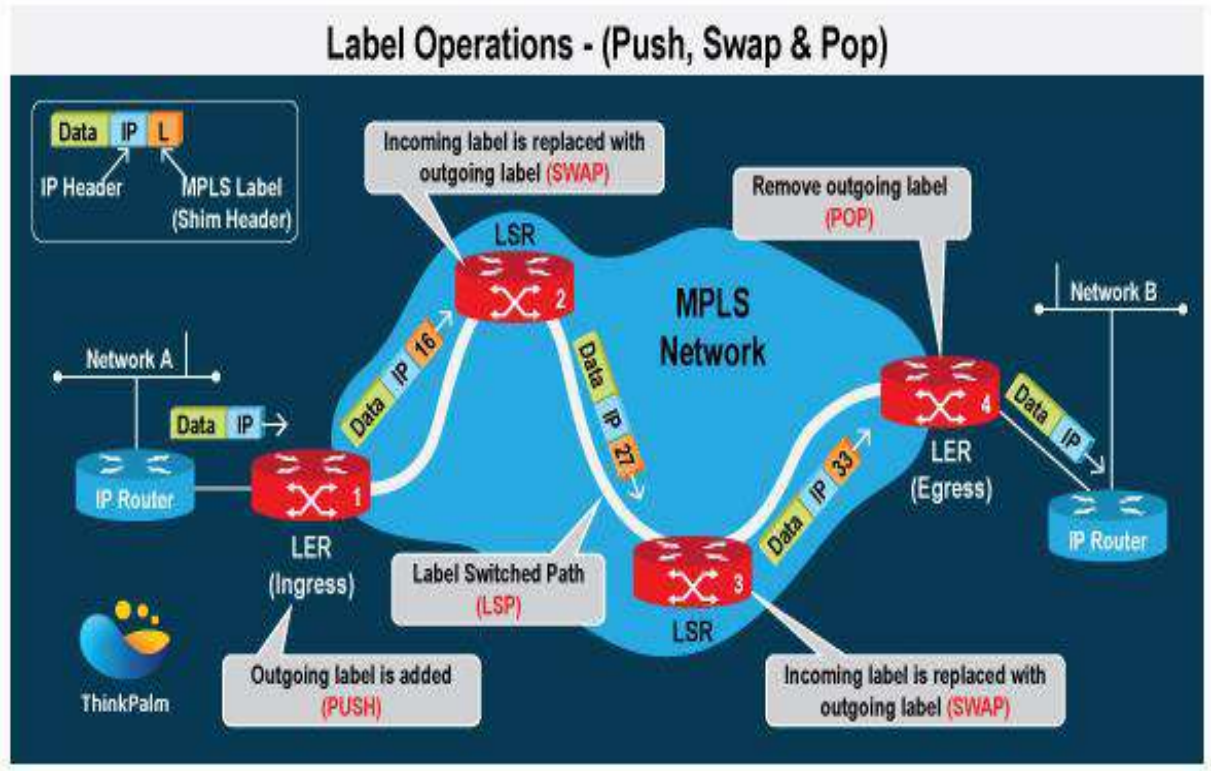

Figure 1.7 : Réseau MPLS simple.

- **Plus rapide que la méthode standard :**
  MPLS est considéré comme le plus rapide et le plus efficace dans le processus de routage ou de Label Switching de la méthode traditionnelle ou classique.
- **Utilisation d'une infrastructure de réseau unifiée :**
  L'utilisation d'un réseau unifié, ce qui est particulièrement utile pour les fournisseurs de services, il n'est pas nécessaire de lancer plusieurs services ou plusieurs réseaux tels que Frame-Relay ou ATM chacun séparément.

- **Noyau sans BGP : [8]**

  Il est également connu que le protocole de routage pour la connexion des fournisseurs de services est BGP avant tous les routeurs, tous les routeurs du réseau du fournisseur de services doivent exécuter ce protocole, un protocole BGP ou un autre protocole, l'informations BGP via la redistribution mais après la technologie MPLS, il n'est plus nécessaire d'exécuter le protocole BGP sur tous les routeurs, d'ajouter de la charge au routeur et de gaspiller une grande partie de sa mémoire, le protocole BGP s'exécute seulement sur les routeurs situés sur les terminaux ou aux limites du réseau du fournisseur.

- **Le Fast Reroute : [9]**

  MPLS Fast Reroute améliore la convergence en cas de panne. En pré-calcul des chemins de sauvegarde pour les pannes de liaison potentiels ou nœud.

  - Dans un réseau IP normal: Le meilleur chemin est calculé à la demande lorsqu'une défaillance est détectée. Le recalcule des meilleurs chemins et la transmission de ces modifications au matériel du routeur peuvent prendre plusieurs secondes, en particulier sur un routeur occupé. Une boucle de routage transitoire peut également se produire, chaque routeur des réseaux prenant connaissance du changement de topologie.

  - Avec MPLS Fast Reroute : Le meilleur calcul du chemin suivant se produit avant que l'échec se produit réellement. Les chemins de sauvegarde sont préprogrammés dans la FIB du routeur en attente d'activation, ce qui peut se produire en quelques millisecondes après la détection d'une défaillance, parce que le chemin entier est défini dans le LSP, les boucles de routage ne peuvent pas se produire pendant la convergence, même si le chemin est brièvement sous-optimal.

1.4.3. MPLS : Applications et Protocoles

1.4.3.1 Applications : [10]

- **MPLS Virtual Private Network (MPLS - VPN):**

Cette application est populaire parmi les fournisseurs de services. Tout client peut disposer d'un réseau virtuel via le réseau MPLS et connecter tous les sites clients, même si les sites sont connectés à différents fournisseurs de services. Le client conserve ainsi sa vie privée et sécurise ses informations.

- **MPLS Traffic Engineering (MPLS - TE):**

Cette application est utilisée pour fournir un niveau de service spécifique aux clients, notamment pour garantir la qualité du son et des images et pour assurer une certaine bande passante au client. Cette application permet aux routeurs de connaître la bande passante disponible et de déterminer s'il existe une congestion entre les expéditeurs et les destinataires.

- **MPLS - TE avec VPN :**

Cette application est une combinaison des deux applications précédentes.

- **MPLS Any Transport over MPLS (MPLS - AToM):**

Comme mentionné précédemment, le réseau MPLS peut charger n'importe quel service L2, tel que Frame-Relay, ATM, Ethernet, PPP et HDLC, et connecter des sites clients à ce protocole, les ordinateurs clients situés à différents endroits se considérant comme directement connectés.

1.4.3.2 Les protocoles : [11]

- **Label Distribution Protocols :**
  - LDP: Label Distribution Protocol : Il est utilisé dans : - IP-MPLS - et MPLS AToM.
  - RSVP: Resource Reservation Protocol : Il est utilisé dans MPLS TE.
  - MP-BGP: Multi-Protocol Border Gateway Protocol: Il est utilisé dans MPLS VPN.

- **MPLS Signaling Protocols: [9]**

Pour utiliser un LSP, il doit être signalé sur tous vos routeurs. Un LSP est un tunnel à l'échelle du réseau, mais une étiquette n'est qu'une valeur de lien local. Un protocole de signalisation MPLS mappe les LSP sur des valeurs d'étiquette spécifiques. Il existe deux principaux protocoles de routage MPLS en usage aujourd'hui:

- Label Distribution Protocol (LDP) : Un protocole simple et non contraint (ne prend pas en charge l'ingénierie du trafic).

- Protocole de réservation de ressources avec ingénierie de trafic (RSVP-TE): Un protocole plus complexe, avec plus de temps système, mais qui inclut également une prise en charge de l'ingénierie du trafic via les réservations de ressources réseau.

La plupart des réseaux complexes devront en réalité utiliser les deux protocoles. LDP est généralement utilisé par les services MPLS VPN (transport de données), mais RSVP-TE est nécessaire pour les fonctionnalités d'ingénierie du trafic.

1.4.4.1 Traffic Engineering (TE)

1.4.4.1 Vue générale : [12]

- L'une des applications MPLS est l'ingénierie du trafic (TE), utilisée pour manipuler le trafic afin de l'adapter à un réseau particulier. TE est important pour les fournisseurs de services à utiliser leurs réseaux avec une grande efficacité et flexibilité.

- Dans un réseau IP traditionnel, les paquets de données sont transmis étape par étape. Chaque routeur situé entre la source et la destination effectue une recherche d'itinéraire et sélectionne le chemin le moins coûteux sur lequel transférer les paquets. L'inconvénient est que si un chemin se révèle optimal en raison de son faible coût, chaque routeur du réseau aura tendance à utiliser ce chemin pour transmettre des paquets. C'est la même lorsque d'autres sous-utilisés, mais plus des chemins de coûts sont disponibles. Cette approche de la transmission de données peut entraîner des problèmes de performances, tels que la perte de paquets ou la latence sur le chemin choisi.

- Certaines technologies de la couche 2 (L2), telles que l'ATM, fournissent des fonctionnalités TE pouvant être utilisées pour l'ingénierie des flux de trafic entre les sources et les destinations. Cependant, ce ne convient pas correctement lorsqu'une connectivité maillée complète est requise entre les différents nœuds. Parce que le routage IP traditionnel est basé uniquement sur l'adresse de destination, les réseaux IP ne disposent pas de mécanisme TE à eux seuls. Une seule option pouvant être utilisée pour le trafic d'ingénierie est la métrique associée à IGP (Interior Gateway Protocol), qui peut être réglée pour préférer un chemin particulier. Toutefois, ne s'applique pas dans les grands réseaux. IP peut être utilisé sur ATM dans un modèle de superposition pour implémenter TE, mais il pose des problèmes d'évolutivité.

- L'ingénierie de trafic désigne le processus de sélection des chemins LSP choisis par le trafic de données afin d'équilibrer la charge sur liens différents, routeurs et commutateurs du réseau. Ceci est le plus important dans les réseaux où plusieurs chemins parallèles ou alternatifs sont disponibles. L'objectif de l'ingénierie du trafic est de faciliter le fonctionnement efficace et fiable des réseaux IP tout en optimisant simultanément l'utilisation des ressources et les performances du réseau. Avant MPLS TE, cette technique était possible avec IP ou ATM en fonction du protocole utilisé entre une paire de routeurs de périphérie dans un réseau.

- Avec l'ingénierie du trafic (TE), au lieu que des décisions d'acheminement soient prises à chaque saut, le routeur d'entrée de tête de l'exploitant du réseau détermine le chemin source à destination pour un trafic spécifique. Ainsi, le trafic qui aurait emprunté un chemin optimal mais encombré peut être dirigé sur des chemins sous-utilisés du réseau, ce qui permet de répartir la charge de bande passante sur différentes liaisons.

1.4.4.2 Exigences MPLS TE : [12]

- La distribution des informations est un protocole d'état des liens, tel qu'OSPF (Open Shortest Path First) ou IS-IS (Système intermédiaire à système), qui est nécessaire pour découvrir la topologie. Ces protocoles ont été améliorés pour transporter des informations supplémentaires liées à TE, telles que la bande passante disponible et d'autres paramètres connexes.
- Le calcul de chemin est un routage basé sur les contraintes (Constraint-Based Routing (CBR)) utilisé pour trouver le chemin le plus court vers un réseau particulier qui répond aux besoins en ressources du flux de trafic. L'algorithme Constrained Shortest Path First (CSPF) (Contrainte du chemin le plus court en premier) qui fonctionne sur la tête de tunnel est utilisé pour cette fonctionnalité.
- La configuration de chemin est un protocole de signalisation permettant de réserver les ressources pour un flux de trafic et d'établir le LSP pour un flux de trafic. Le protocole de réservation de ressources (RSVP) est utilisé à cette fin et a été amélioré avec des extensions TE pour le transport label et la construction du LSP. Une alternative au protocole RSVP pour MPLS TE est le routage contraint avec le protocole LDP (Label Distribution Protocol).

- le transfert du trafic est un composant qui correspond au plan d'acheminement de MPLS TE et transfère le trafic à travers un tunnel MPLS TE, construit sous la forme d'un LSP par le module d'établissement de trajet, et basée sur les informations disponibles à partir d'autres composants.

1.4.4.3 Resource Reservation Protocol –Traffic Engineering (RSVP-TE):

- Le protocole de réservation de ressources (RSVP) réserve des ressources le long du chemin de bout en bout d'un flux de trafic dans un réseau IP. La demande RSVP consiste en une FlowSpec qui spécifie l'exigence de qualité de service (QoS) pour le flux de trafic et d'une FilterSpec qui définit le flux qui doit recevoir la priorité QoS. Une fois que la bande passante nécessaire est réservée le long du chemin avec RSVP, l'application qui a fait la demande commence à transmettre le trafic. RSVP est souvent utilisé par les applications temps réel pour configurer des réservations de bande passante.

- Le protocole de signalisation RSVP a été étendu avec les fonctionnalités MPLS pour prendre en charge MPLS TE. Cela a permis à RSVP de configurer LSP dans un réseau MPLS TE. Avec RSVP-TE, le routeur de tête envoie un message RSVP PATH qui vérifie la disponibilité des ressources demandées sur tous les LSR dans le chemin sur lequel le tunnel TE doit être créé. Lors de la réception du message PATH, le routeur final dans le chemin confirme ensuite la réservation avec un message RSVP RESERVATION, qui confirme l'attribution d'un LSP à un tunnel TE. Ce message est ensuite propagé en amont vers le routeur de tête de réseau via tous les LSR situés sur le futur chemin de tunnel TE.

- Une fois que tous les LSR du chemin acceptent et confirment le LSP, le LSP MPLS TE est opérationnel. Le routeur de tête de réseau peut alors diriger le trafic à travers de nouveaux tunnels en fonction des besoins et votre réseau MPLS optimisé pour le trafic est prêt.

1.4.4.4 Les avantages de l'ingénierie du trafic (TE) : [13]

L'ingénierie du trafic dans MPLS implique la technique de direction du trafic qui circule dans un réseau. Plusieurs procédures de routage implémentent le transfert de paquets pour une transmission sécurisée. Les avantages suivants améliorent l'ingénierie du trafic :

- Minimiser la congestion du réseau : un réseau MPLS peut implémenter TE pour réduire le blocage du réseau et améliorer les performances. Les tunnels dirigent le trafic du chemin encombré vers le chemin sous-utilisé disponible pour alléger la congestion du trafic.

- MPLS Fast Reroute en cas de défaillance de lien / nœud : la fonctionnalité de redirection rapide MPLS permet de gérer les défaillances de liaison ou de nœud en dirigeant le trafic encapsulé vers un chemin secondaire préconfiguré en cas de défaillance du chemin primaire.

- classe de service (CoS) : ce champ de 3 bits détermine la valeur de la CoS sur la base de laquelle le trafic dans sa file d'attente prioritaire est utilisé pour la transmission.

- Identification du trafic client : l'ingénierie du trafic MPLS classe le trafic client en fonction du fournisseur de service utilisé dans le réseau MPLS. C'est uniquement dû à la fonctionnalité CoS de la technologie qui catégorise le trafic.

- MPLS TE fournit une approche intégrée de l'ingénierie du trafic en combinant les capacités d'ingénierie du trafic d'ATM avec la flexibilité et la différenciation de classe de service (CoS) de l'IP. La nature du MPLS TE permet d'éviter les problèmes associés au modèle de superposition. Comme pour ATM ou Frame Relay, Label Switching Path (LSP) construit automatiquement par MPLS TE contrôle le chemin d'un flux de trafic vers une destination particulière, plutôt que le transfert purement basé sur la destination

1.4.4.5 Les inconvénients de l'ingénierie du trafic : [13]

Bien que MPLS apporte plusieurs avantages à sa mise en œuvre, il existe peu de retraits basés sur le réseau utilisé et la bande passante disponible. Retrouvez ci-dessous les inconvénients techniques:

- Utilisation excessive des liaisons secondaires : en cas de défaillance des liaisons, la redirection rapide de MPLS TE utilise des tunnels de secours pour rediriger le trafic sur la liaison secondaire.

- Configuration de chemin manuelle: pour implémenter l'ingénierie du trafic, les chemins nécessitent une configuration manuelle indépendamment de la présence du protocole Internet pour le routage de paquets.

- Dépendance de protocole pour le réacheminement automatique: Si un réseau utilise le protocole de routage OSPF (Open Shortest Path First), le calcul automatique du chemin et le réacheminement systématique du trafic IP peuvent être effectués dans des chemins TE MPLS. Ça s'applique même aux réseaux utilisant le protocole IS-IS (Intermediate System to Intermediate System).

- Variation des performances dans MPLS Fast reroute : les mécanismes de qualité de service maintiennent la bande passante pour les tunnels fonctionnant en secours. Comme les nœuds intermédiaires du chemin TE ne sont pas configurés manuellement, le trafic utilisant la redirection rapide sur le chemin alternatif trébuchera sur des défaillances de liaison. De plus, la configuration manuelle des nœuds intermédiaires dans le réseau MPLS n'est pas réalisable car la technologie manque de telles options.

- Absence de système de mappage systématique: le mappage dynamique du trafic IP sur des chemins MPLS TE n'est pas réalisable, car les routeurs configurant le chemin ne reconnaissent pas la topologie des paramètres régionaux OSPF suivants.

1.4.4. Travaux de standardisation et de normalisation.

Dans quelques années, la technologie MPLS (Multi-Protocol Label Switching) est passée de la technologie exotique à l'outil principal utilisé par les fournisseurs de services pour créer des services générateurs de revenus. MPLS a été créé dans la seconde moitié des années 1990 afin que les routeurs puissent éviter de rechercher des itinéraires dans les tables de routage, améliorant ainsi la vitesse de circulation du trafic réseau. En 1994, Toshiba a présenté ses premières idées à l'IETF (Internet Engineering Task Force), qui était des précurseurs de normes MPLS actuelles. Le principe MPLS a été proposé en 1996 par Ipsilon Networks comme une technologie *d'IP Switching* fonctionnant sur ATM et Cisco

comme Tag Switching. Le nom a été modifié pour Label Switch lorsque l'IETF a créé un MPLS working group pour unifier MPLS en 1997, et normalisé sous la norme RFC 3031.

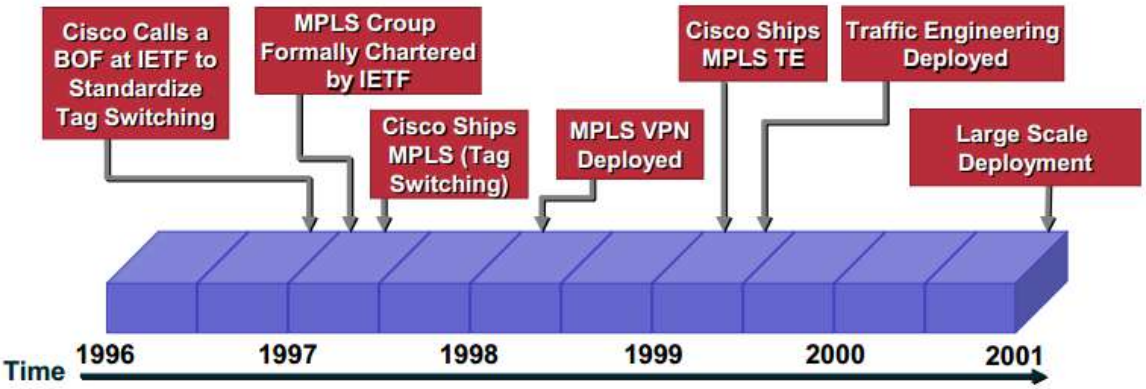

Figure 1.8 : Evolution de MPLS. [11]

L'évolution de standard MPLS a traversé les étapes suivantes:

1- Normalisation de la MPLS (Multi Protocol Label Switching) Basé sur le trafic IP. **RFC 3031** (architecture), **RFC 3036** (LDP)….
2- Ajout du service «Traffic Engineering» (MPLS-TE), la possibilité de forcer les paquets à prendre la route la plus courte non IGP. C'est toujours la principale application de MPLS aujourd'hui. **RFC 2702.**
3- Développement de MPLS Basé sur les longueurs d'ondes et ajout d'un nouveau protocole, LMP (Link Management Protocol) pour la gestion des liens et des erreurs. **RFC 4327.**
4- GMPLS (Generalized MPLS) : Extension de MPLS-TE qui permet aux LSR de supporter Plusieurs types de commutation, Paquets, TDM, Lambdas, Fibres ect..., généralisation de la définition d'un label, ajout de la signalisation via le protocole RSVP-TE, et amélioration des protocoles de routage pour décrire la topologie du réseau : OSPF et IS-IS. **RFC 3471.**

1.5. Problématique

L'avènement d'un accès rapide à Intemet pour tous et partout a simplifié l'accès aux services et aux connaissances, ce qui constitue un réel progrès dans notre façon de vivre. Cependant‹ l'un des problèmes cruciaux que doit résoudre un fournisseur de services consiste à savoir comment réduire au minimum la congestion de son réseau.

L'ingénierie du trafic (TE) est devenue un impératif essentiel pour les fournisseurs de services Internet afin d'optimiser l'utilisation des ressources réseau existantes et de maintenir la qualité de service (QoS) globale souhaitée avec moins de ressources réseau. L'un de ses objectifs principaux est le Load Balancing. Le Load Balancing est un aspect important de l'ingénierie du trafic. Il peut utiliser certains mécanismes pour mapper une partie du trafic sur certaines routes sous-utilisées pour éviter les embouteillages sur le chemin principal choisi dans le réseau (eg. Le chemin le plus court pour certains protocoles) et pour améliorer le débit total du réseau et l'utilisation des ressources du réseau.

MPLS présente de nombreux avantages soutenir l'ingénierie du trafic. MPLS permet un routage explicite efficace des LSPs, ce qui peut être utilisé pour optimiser l'utilisation des ressources du réseau et améliorer les performances du trafic.

Load Balancing est un aspect clé de MPLS TE, il nécessite une capacité à contrôler les flux de trafic avec précision. Par conséquent, la question cruciale est de savoir comment équilibrer le trafic réseau tout en maintenant une allocation de ressources efficace et en améliorant les performances du réseau et la qualité de service Internet et comment réduire au minimum la congestion.

Dans ce travail, nous proposons d'étudier les techniques de l'ingénierie de Traffic (TE) dont le mécanisme Load Balancing technique d'équilibrage de charge, pour minimiser la congestion dans le contexte des réseaux MPLS.

1.6. Objectives et Contribution

L'objectifs de notre travail est l'étude et la modélisation des techniques de TE dans les réseaux MPLS d'une façon générale et en particulier les algorithmes de gestion de flux pour réduire la congestion, qui permettent d'améliorer l'utilisation des ressources et les performances du réseau, tout en garantissant la qualité de service.

Nous adoptons un regard nouveau sur les algorithmes existant, cette nouvelle vision adopte des approches nouvelles pour découvrir les itinéraires satisfaisant un certain critère pour réduire la congestion.

Dans ce contexte, nous avons pu apporter un nouvel algorithme de gestion de la congestion dans les réseaux MPLS en utilisant un mécanisme d'équilibrage de charge. L'idée principale est de dévier efficacement les LSP des liens les plus encombrés du réseau vers ceux qui sont sous-utilisés.

1.7. Structure de thèse

Ce manuscrit de thèse est composé de six chapitres, organisés comme suit :

- **Le deuxième chapitre** : dans ce chapitre, nous passons en revue les algorithmes de gestion de la congestion existant dans réseau MPLS.

- **Le troisième chapitre** : est dédié à la présentation de nouvel algorithme de gestion de flux basé sur le mécanisme Load Balancing.

- **Le quatrième chapitre :** Dans ce chapitre nous analysons la performance de cet algorithme par simulations en utilisant OMNET++.

- **Dans le dernier chapitre :** nous conclurons cette étude par une synthèse de notre contribution et nous montrerons à quel point notre objectif a été atteint. Nous exposerons ensuite les perspectives et les futurs travaux envisageables pour approfondir cette étude.

Chapitre

2 Revue de littérature sur les algorithmes Load Balancing dans les réseaux MPLS

Contenu

2.1. Introduction :

MPLS est un mécanisme de réseau de télécommunication de hautes performances qui dirige les données d'un nœud de réseau vers un autre en se basant sur des étiquettes de chemin court plutôt que sur des adresses réseaux longues, en évitant les recherches complexes dans une table de routage. L'ingénierie du trafic (TE) est la technologie d'ingénierie de réseau qui permet de mapper un flux d'activités sur l'accès physique réel, mais qui peut également optimiser automatiquement les ressources du réseau afin de répondre aux exigences de performances spécifiques des services d'application, avec macro-régulation et micro-contrôle. MPLS et MPLS-TE jouent un rôle clé dans la gestion des ressources des réseaux de transmission de paquets contemporains, ainsi que dans la fourniture de services de réseau de virtualités de bout en bout. Dans ce contexte, la planification des itinéraires et l'allocation des ressources dans les réseaux MPLS, conformément aux exigences MPLS-TE, revêtent une importance capitale pour l'optimisation de l'utilisation du réseau. L'un de ses objectifs principaux est d'équilibrer les charges sur un réseau, c'est-à-dire que l'équilibrage de la charge est un aspect important de l'ingénierie du trafic. Il peut utiliser certains mécanismes pour mapper une partie du trafic sur les routes surutilisées sur certaines routes sous-utilisées afin d'éviter l'encombrement sur l'itinéraire de plus court chemin et pour promouvoir le débit total du réseau et l'utilisation des ressources du réseau. MPLS présente de nombreux atouts pour la prise en charge de l'ingénierie de trafic. Dans cet article, nous examinons divers algorithmes liés à l'équilibrage de la charge dans MPLS.

Dans ce chapitre, nous allons passer en revue les concepts et les enjeux liés à notre problématique traitée dans ce mémoire. Vu que la finalité de notre travail est de proposer un nouvel algorithme optimal pour la gestion de la congestion dans les réseaux MPLS, ce chapitre aborde essentiellement le MPLS - Traffic Engineering. Nous commencerons par une brève présentation du Traffic Engineering dans les réseaux MPLS et par la suite nous définirons les besoins du Traffic engineering pour les MPLS. Puis nous présenterons le concept d'équilibrage de charge (Load Balancing); ainsi nous définirons les besoins de Load Balancing. Nous allons dresser un état de l'art des différents algorithmes de Load Balancing dans les réseaux MPLS, nous proposerons aussi une classification de ces algorithmes et par la suite nous présenterons, une étude comparative et synthétique, tout en soulignant les avantages et les limites de chaque algorithme. Nous terminerons ce chapitre par une conclusion.

2.2. MPLS – TE : [1]

L'ingénierie du trafic (TE) est la technologie d'ingénierie de réseau qui permet de mapper un flux d'activités sur l'accès physique réel, mais qui peut également optimiser automatiquement les ressources du réseau afin de répondre aux exigences de performances spécifiques des services d'application, avec macro-régulation et micro-contrôle. L'ingénierie du trafic a pour objectif principal la promotion des opérations du réseau efficaces et fiables tout en optimisant l'utilisation des ressources du réseau afin d'améliorer ses performances. L'ingénierie du trafic (TE) résout les problèmes de descente de la qualité de service qui résultent du flux d'activités entre les ressources disponibles et que l'efficacité de la cartographie n'est pas grande en raison de l'optimisation et de la restriction des réunions, tout simplement constituées de services intégrés (Intserv) et de services différenciés (Diffserv) insuffisants des aspects macro-économiques de l'utilisation des ressources du réseau.

MPLS facilite l'engagement des ressources réseau de manière à équilibrer la charge face à une demande donnée et à s'engager à prendre en charge les différents niveaux de support afin de répondre aux diverses exigences du trafic d'utilisateur. La possibilité de définir de manière dynamique des itinéraires, de planifier les engagements de ressources en fonction de la demande connue et d'optimiser l'utilisation du réseau est appelée «Traffic Engineering».

Le processus de routage du trafic de données afin d'équilibrer la charge du trafic sur les divers liens, routeurs et commutateurs du réseau s'appelle MPLS-TE. MPLS et MPLS-TE jouent un rôle clé dans la gestion des ressources des réseaux de transmission de paquets contemporains, ainsi que dans la fourniture de services de réseau virtualisés de bout en bout. Dans ce contexte, la planification des itinéraires et l'allocation des ressources dans les réseaux MPLS, conformément aux exigences MPLS-TE, revêtent une importance capitale pour l'optimisation de l'utilisation du réseau.

Load Balancing est un aspect essentiel de MPLS-TE. Il s'agit du processus d'ajustement de la charge de travail d'un ensemble de systèmes ou d'applications afin de s'assurer qu'aucun système ou aucune application n'est sous-utilisé ou sur-utilisé. Cela nécessite une capacité à contrôler les flux de trafic avec précision.

Utiliser certains mécanismes pour mapper une partie du trafic sur les routes surutilisées sur certaines routes sous-utilisées afin d'éviter les encombrements sur la route du plus court chemin. Il

permet un routage explicite et efficace du LSP, qui peut être utilisé pour optimiser l'utilisation des ressources du réseau et améliorer les performances du trafic.

2.3. Raisons d'utiliser du Traffic Engineering dans MPLS: [1]

1- La congestion du réseau dû à l'évolution des modèles du trafic.

2- La meilleure utilisation de la bande passante disponible

3- Route autour des liens / nœuds défaillants.

4- Créer de nouveaux services de lignes louées virtuelles

5- Planification de la capacité.

2.4. Load Balancing : [1]

Load Balancing est une division même du travail de traitement entre deux ordinateurs ou plus et / ou processeurs, liaison réseau, périphériques de stockage ou autres périphériques, offrant ainsi un service plus rapide et plus efficace. En termes simples, l'équilibrage de la charge est le processus par lequel le trafic IP (Internet Protocol) entrant peut être distribué sur plusieurs serveurs. Cela améliore les performances des serveurs, optimise leur utilisation et garantit qu'aucun serveur n'est surchargé, qu'il soit réalisé via un logiciel, du matériel ou les deux, et qu'il utilise souvent plusieurs serveurs qui semblent être un seul et même ordinateur (également connu sous le nom de clustering d'ordinateur).

En règle générale, deux serveurs Web ou plus sont utilisés dans un schéma d'équilibrage de charge. Si l'un des serveurs commence à être surchargé, les demandes sont transférées à un autre serveur. Ce processus réduit le temps de service en permettant à plusieurs serveurs de gérer les demandes. La durée du service est réduite en utilisant un équilibreur de charge pour identifier le serveur ayant la disponibilité appropriée pour recevoir le trafic. Le processus, très généralement, est simple. Une demande de page Web est envoyée à l'équilibreur de charge, qui la transmet à l'un des serveurs. Ce serveur répond à l'équilibreur, qui à son tour envoie la demande à l'utilisateur final. L'équilibrage de charge permet au service de continuer même en cas de panne du serveur en raison d'une panne du serveur ou de sa maintenance.

L'affectation d'un facteur de charge: La charge fait référence à un numéro attribué à une demande de service en fonction du temps requis pour exécuter ce service. Les charges sont affectées aux services afin que le système puisse comprendre la relation entre les demandes. Pour suivre la quantité de travail ou la charge totale, exécutée par chaque serveur dans une configuration, l'administrateur attribue un facteur de charge à chaque service et demande de service. Un facteur de charge est un nombre indiquant le temps nécessaire à l'exécution d'un service ou d'une demande. Sur la base de ces chiffres, des statistiques sont générées pour chaque serveur et conservées sur le tableau d'affichage de chaque machine. Chaque tableau d'affichage conserve une trace de la charge cumulative associée à chaque serveur. Ainsi, lorsque tous les serveurs sont occupés, le système peut sélectionner celui qui a la charge la plus faible.

2.5. Load Balancing dans MPLS : [1]

Lorsque plusieurs tunnels TE ont le même coût, le trafic peut être équilibré entre eux. Le trafic peut également être équilibré entre le chemin IP natif et les tunnels TE si le coût du routage est identique. Cette situation a quelques restrictions. Lorsque vous effectuez un équilibrage de charge sur des tunnels TE, cet équilibrage peut même consister en un équilibrage des charges inégal. L'équilibrage de charge du trafic est pondéré proportionnellement aux besoins en bande passante des tunnels TE. Si vous disposez d'un tunnel de 80 Mo et d'un autre de 20 Mo de bande passante réservée, le rapport d'équilibrage de la charge est de 4: 1 ou le premier tunnel devrait recevoir quatre fois plus de trafic que le second.

Toutefois, le rapport d'équilibrage de la charge est approximatif, car Cisco Express Forwarding (CEF) ne dispose que de 16 compartiments de hachage. Lorsqu'un LSR effectue l'équilibrage de charge sur un ou plusieurs chemins IP et un ou plusieurs tunnels TE, l'équilibrage des coûts est toujours égal. Cela signifie que chaque chemin reçoit le même volume de trafic. Plusieurs tunnels TE peuvent être pratiques lorsque la quantité de bande passante à réserver entre une paire de routeurs est supérieure à la capacité en bande passante des liaisons. Vous pouvez ensuite tout simplement créer plusieurs tunnels TE avec chacun un morceau de la bande passante requise.

2.6. Raisons d'utiliser de Load Balancing dans MPLS: [1]

1. TE est devenu un impératif essentiel pour que les FAI optimisent l'utilisation des ressources du réseau existantes et maintiennent la qualité de service souhaitée avec moins de ressources du réseau.

2. MPLS offre de nombreux avantages en termes d'ingénierie de la circulation.

3. L'équilibrage de la charge est un aspect clé de MPLS TE, il nécessite une capacité à contrôler les flux de trafic avec précision.

4. Utilise certains mécanismes pour mapper une partie du trafic sur les routes sur utilisées sur certaines routes sous-utilisées afin d'éviter les embouteillages sur l'itinéraire du plus court chemin.

5. Promouvoir le débit total du réseau et l'utilisation des ressources du réseau.

6. Lorsqu'il existe plusieurs routes parallèles connectant deux nœuds, la tâche de l'équilibrage de charge consiste à mapper les flux de trafic sur ces routes pour obtenir le débit du réseau maximal et améliorer simultanément l'utilisation totale des ressources du réseau.

7. Il permet un routage explicite efficace du LSP, qui peut être utilisé pour optimiser l'utilisation des ressources du réseau et améliorer les performances du trafic.

8. Par exemple, plusieurs chemins peuvent être utilisés simultanément pour améliorer les performances d'une source donnée à une destination.

2.7. Les différents algorithmes pour Load Balancing dans les réseaux MPLS:

Dans le contexte des réseaux MPLS, plusieurs méthodes ont été développées pour éliminer la congestion et l'équilibrage de la charge du réseau. Différents algorithmes ont été introduits par différents chercheurs. Nous examinons ici quelques algorithmes relatifs à l'équilibrage de charge (Load Balancing) qui sont classés en deux parties:

2.7.1 Les algorithmes Load Balancing statique:

Dans **[Keping Long *et al*, 2001]**, les auteurs proposent un algorithme d'équilibrage de charge statique basé sur la topologie (**Topology-Based Static Load-Balancing Algorithm -TSLB**) ; cet algorithme est amélioré à partir de l'algorithme Shortest-Path. Dans cet algorithme, le nouveau flux de trafic est d'abord acheminé via Shortest-Path; si Shortest-Path a une capacité insuffisante pour satisfaire la demande de bande passante du nouveau flux de trafic, ce flux de trafic sélectionnera l'itinéraire le plus long; Si aucune route ne satisfait la demande de bande passante du nouveau flux de trafic, l'algorithme échoue. Ce dernier réduit considérablement la congestion dans le chemin le plus court. Mais cet algorithme a ses inconvénients et entraîne à une faible utilisation des ressources du réseau. Nous pouvons donc constater que, sous certaines conditions spéciales, cet algorithme distribue les ressources de manière déraisonnable, ce qui peut entraîner à un faible débit et une faible utilisation des ressources du réseau.

Dans **[Keping Long *et al*, 2001]**, les auteurs proposent un algorithme d'équilibrage de charge statique basé sur des ressources (**Resource-Based Static Load-Balancing Algorithm -RSLB-**), lorsqu'un nouveau flux de trafic arrive, Ingress sélectionnera un itinéraire dont la capacité disponible est la plus proche de la demande de bande passante du flux de trafic (mais plus grande que la demande). Ainsi, il peut réserver une route de plus grande capacité pour le flux de trafic à venir qui a une demande de bande passante plus importante. Mais, parce que le flux de trafic est mappé sur un itinéraire dont la capacité disponible est très proche de sa demande de bande passante; La capacité de la liaison ne convient pas bien à la fluctuation du débit d'envoi du flux de trafic et peut entraîner à une perte de données à un moment spécial lorsque le débit en rafale du flux de trafic dépasse la capacité de la liaison.

- **L'inconvénient des algorithmes d'équilibrage de charge statique:**

Ces deux algorithmes ont tous les deux leurs inconvénients. Dans certains cas particuliers, ils peuvent entraîner à une faible utilisation de la source. La raison fondamentale en est que ces algorithmes sont tous les deux statiques. En l'absence de priorité, une fois qu'un ER-LSP est établi, ses ressources ne peuvent pas être préemptées par d'autres ER-LSP et il ne peut plus se rediriger vers un autre meilleur itinéraire (sauf qu'une erreur de liaison se produit).

2.7.2 Les algorithmes de Load Balancing dynamique :

Dans **[Gustavo B *et al*, 2004]**, Une technique d'interférence minimale augmentant le nombre de chemins acceptés dans le réseau est proposé comme l'une des techniques les plus efficaces pour améliorer le routage en ligne: l'algorithme de routage d'interférence minimale (**minimum interference routing algorithm MIRA**). MIRA identifie les chemins «critiques» et évite de les surcharger afin de minimiser le futur taux de rejet des requêtes. L'idée principale de MIRA est d'éviter le routage sur les liens qui pourraient « interférer » avec la configuration éventuelle des chemins. Ces liaisons sont appelées liaisons « critiques » et possèdent la propriété suivante: lorsque leur capacité est réduite d'une unité de bande passante, le flux de données maximal entre une paire de nœuds de destination source est également réduit d'une unité de bande passante. Le but de MIRA est donc de trouver un chemin réalisable contenant le moins de liens critiques. L'idée de routage sur des liaisons non interférentes est bonne et logique; Cependant, MIRA présente deux faiblesses, les calculs de route ; MIRA peuvent être coûteux en calcul et une utilisation de réseau déséquilibrée ne se traduit pas nécessairement par des gains de performance équivalents.

Dans **[Gustavo B *et al*, 2004]**, Les auteurs proposent un nouvel algorithme appelé « **Light Interference Routing (LMIR)** », vise à optimiser l'utilisation des ressources avec une complexité de calcul réduite. Comme d'autres algorithmes de routage d'interférence minimaux existants, LMIR est un algorithme en ligne qui ne fait aucune hypothèse concernant le trafic du réseau. Son principal avantage par rapport aux approches précédentes est la faible complexité de calcul qui conduit à une plus grande évolutivité.

Les auteurs **[Elio Salvadori *et al*, 2003]** proposent un algorithme : **Dynamic Load Balancing Algorithm**. Dans DYLBA, la recherche d'un autre itinéraire est effectuée pour tous les LSP traversant les liaisons les plus encombrées. DYLBA peut prendre en compte simultanément les caractères de la topologie du réseau et la demande de bande passante du trafic. Dans des conditions de charge faible, les flux de trafic peuvent être cartographiés sur les itinéraires à capacité courte et élevée; lorsque la charge devient lourde, un faible flux de trafic peut être redirigé vers un autre itinéraire approprié, de manière à réserver un itinéraire à grande capacité pour un flux de trafic important. L'algorithme DYLBA peut améliorer considérablement le débit du réseau et l'utilisation des ressources. L'algorithme DYLBA

peut rapidement mapper un nouveau flux de trafic sur un itinéraire approprié. Mais la principale restriction de la technologie DYLBA est qu'il ne doit pas y avoir un très grand nombre de routes parallèles entre Ingress et Egress et que chaque flux de trafic ne doit pas être redirigé très fréquemment**.** Malgré des résultats encourageants, cet algorithme est très exigeant du point de vue des calculs en raison de la vaste recherche effectuée pour trouver le meilleur LSP à réacheminer. Les résultats de la simulation montrent que l'algorithme fonctionne mieux que MIRA dans des conditions spécifiques du trafic du réseau, en réduisant la probabilité de rejet du LSP et les retards moyens de bout en bout.

Les auteurs **[Tijani Mohammed *et al*, 2007]** proposent un nouvel algorithme appelé **Optimal Dynamic Load Balance Algorithm - ODLB -** est supérieur à DYLBA en termes de taux d'utilisation de la bande passante et de la fréquence de réacheminement du trafic en cours. ODLB fournit la solution optimale pour résoudre les problèmes d'équilibrage de charge sur les itinéraires parallèles à partir de la même source vers la destination. Il recherche de manière exhaustive tous les scénarios de différentes combinaisons de trafic sur les itinéraires. La recherche exhaustive est utilisée parce qu'elle garantit la solution optimale avec une vitesse rapide lorsque le nombre de routes parallèles et le nombre de trafic ne sont pas importants. Dans la situation réelle, un réseau comportant de nombreuses routes parallèles acheminant le trafic avec la même priorité n'est pas un cas courant. Par conséquent, l'algorithme ODLB combinant une recherche exhaustive est une solution très pratique pour s'attaquer aux problèmes réels du TE. ODLB réduit la fréquence de réacheminement du trafic en cours, car les nouveaux itinéraires sont calculés uniquement en cas de congestion. En cas d'encombrement, le calcul est effectué simultanément à l'acheminement du trafic en cours. Cela peut entraîner à un certain retard dans le transport du nouveau trafic, mais l'ancien trafic reste sur les itinéraires calculés précédemment sans interruption.

Les auteurs ont travaillé sur l'équilibrage de la charge dans un réseau MPLS à l'aide d'outils en ligne. Avec cette approche, Ils proposent un algorithme d'équilibrage de charge appelé **MATE (Adaptive Trafic Engineering) [Anwar Elwalid *et al*, 2001].** L'objectif principal de MATE est d'éviter les encombrements du réseau en équilibrant de manière adaptative la charge entre plusieurs chemins, sur la base d'une mesure et d'une analyse de l'encombrement des chemins. MATE adopte une approche minimaliste en ce sens que les nœuds intermédiaires ne sont pas obligés d'effectuer une ingénierie de trafic ou des mesures

en plus du transfert normal de paquets. De plus, MATE n'impose aucune planification, gestion de mémoire tampon, ni caractérisation a priori du trafic aux nœuds. MATE est destiné au trafic ne nécessitant pas de réservation de bande passante. Les résultats de notre simulation montrent que MATE peut éliminer efficacement les déséquilibres de trafic pouvant survenir entre plusieurs LSP. Nous observons que, dans de nombreux cas, il est possible de réduire considérablement les taux de perte de paquets élevés en transférant correctement une partie du trafic vers des LSP moins chargés.

Les auteurs **[R. Boutaba *et al*, 2002]** proposent un nouvel algorithme de routage appelé **Dynamic Online Routing Algorithm (DORA).** DORA vise à utiliser efficacement les ressources du réseau existantes et à réduire les congestions sur le réseau en mappant soigneusement les chemins sur le réseau. Le problème de l'établissement de chemins garantis par la bande passante a été étudié ailleurs. L'objectif principal de DORA est de répartir uniformément les chemins avec la bande passante réservée sur le réseau afin de permettre à davantage de chemins futurs d'être acceptés dans le réseau et d'équilibrer la charge de trafic. Lors du calcul du chemin, l'opération principale dans DORA consiste à éviter le routage sur les liaisons qui ont le potentiel élevé de faire partie d'un autre chemin et qui disposent d'une faible bande passante résiduelle. Cet algorithme est inspiré de l'algorithme MIRA, mais il fonctionne mieux en termes de taux de rejet de configuration de chemin et de pourcentage de réacheminement en cas de défaillance du lien / nœud avec une complexité de calcul beaucoup moins grande.

Dans **[Sherif Ibrahim Mohamed *et al*, 2006]**, les auteurs proposent le schéma **DEPR (Distributed Explicit Partial Rerouting)** comme suppression des encombrements réactifs pour les réseaux MPLS. DEPR supprime les encombrements en redirigeant les LSP traversant le lien encombré vers un itinéraire partiel défini autour du lien encombré (contrairement aux autres méthodes qui rétablissent l'itinéraire complet de la source à la destination). Il a été démontré que le système s'adapte rapidement à la dynamique du réseau et s'adapte bien aux grands réseaux. Il améliore également le débit global du réseau et les performances de délai. Un autre aspect qui rend le réacheminement partiel plus attrayant est que le fait de décaler le trafic du lien encombré peut avoir un impact négatif sur les liens du nouveau chemin alternatif, même s'il semble mieux. Il est clair que, moins la longueur du chemin alternatif, plus le potentiel impact négatif sur les liens du réseau. De plus, il n'est pas nécessaire d'avoir toutes les informations sur l'état du réseau sur chaque nœud, mais

uniquement dans la région d'exploration du nœud. De plus, il s'adapte bien aux grands réseaux. En DEPR, chaque nœud surveille l'utilisation de ses liens directs. Si un lien est trouvé encombré, le nœud établit un chemin partiel pour réacheminer le trafic de certains LSP. Si plusieurs chemins partiels existent, la préférence est donnée à celui avec la plus faible utilisation.

Les auteurs **[A.B. Bagula *et al*, 2004]** présentent un **Least Interference Optimization Algorithm (LIOA)** est présenté et qui réduit le brouillage entre les flux concurrents en équilibrant le nombre et la quantité de flux acheminés par une liaison afin de réaliser un routage efficace des LSP garantis par la largeur de bande dans MPLS. Les premiers résultats de la simulation montrent que les performances de LIOA sont meilleures que celles de plusieurs algorithmes de routage bien connus, tels que les algorithmes MHA (Minimum Hop Algorithm), OSPF (Ouvert le plus court chemin), CSPF (Contrainte Shortest Path First) et MIRA (Minimum Interference Routing Algorithm). Plusieurs paramètres de performance, notamment le rejet du LSP en cas d'encombrement, le réacheminement réussi des LSP en cas de défaillance d'une liaison unique et la facilité de mise en œuvre.

Dans **[Bing-feng Cui *et al*, 2004]**, Les auteurs proposent un algorithme de réglage de la file d'attente **(Queue Tuning Algorithm QTA)**, cet algorithme prend en charge la qualité de service tout en réalisant un équilibrage de charge pour optimiser l'utilisation des ressources du réseau. La simulation indique que l'algorithme fonctionne bien dans les réseaux MPLS. Cet algorithme prend en charge la qualité de service du trafic tout en améliorant l'efficacité des LSP grâce à une distribution dynamique du trafic en fonction de l'état des LSP. Dans l'algorithme, Load Balancing est obtenu par le changement de mappage de différentes files d'attente sur différents LSP.

Le rôle des chemins parallèles à commutation d'étiquettes (LSP) dans l'équilibrage de charge et le routage basé sur des contraintes est étudié. Un algorithme dans **[J. Tang *et al*, 2005]** appelé schéma de bande passante basée sur un chemin parallèle (**Path-based Bandwidth Scheme - PPBS)** est proposé pour utiliser les LSP parallèles dans le choix d'un chemin de contrainte de bande passante. L'algorithme PPBS proposé a pour conséquence que les besoins en bande passante d'un flux de trafic global ne peuvent être satisfaits que par un certain nombre de LSP. Par conséquent, une tâche nécessaire du routeur d'entrée consiste à répartir le trafic de l'agrégat de flux sur tous les chemins de

support de manière appropriée. Les chemins calculés dans cet algorithme sont des chemins parallèles en ce sens qu'aucun chemin ne partage le même lien. Un avantage de ces chemins parallèles réside dans le fait que les corrélations ou les interférences de trafic entre les chemins de la même paire de source-destination sont supprimées et que les chemins parallèles peuvent être considérés comme indépendants les uns des autres, ce qui facilite grandement le provisionnement de la bande passante et l'analyse des performances. L'inconvénient de cette méthode est que le nombre de chemins parallèles entre une paire dépend de la connectivité du réseau, ou plus spécifiquement des ports des routeurs de commutation d'étiquettes, y compris le nœud d'entrée. L'algorithme PPBS fournit une méthode pour étendre l'algorithme de routage existant basé sur les contraintes afin d'utiliser les LSP parallèles lors du choix d'un chemin avec une bande passante suffisante.

Dans **[J. Tang *et al*, 2005]**, en conjonction avec le schéma PPBS, un algorithme **Feedback Based Load Balancing (FBLB)** est proposé pour répartir correctement le trafic sur les LSP parallèles déterminés par le PPBS. Cet algorithme FBLB s'appuie sur les paquets de signalisation pour renvoyer les informations d'état du réseau à la source. Par conséquent, les sources peuvent ajuster la distribution du trafic dans chaque LSP avec précision et rapidité. Les résultats de la simulation montrent que l'algorithme FBLB est simple et efficace. Cependant, compte tenu du fait qu'un protocole de signalisation est déjà utilisé dans les réseaux MPLS, la mise en œuvre de la FBLB nécessiterait une extension légère ou modérée du protocole existant. Par conséquent, le coût de mise en œuvre ne serait pas élevé par rapport aux avantages obtenus.

Dans **[K. Abboud *et al*, 2008]**, Les auteurs proposent un algorithme hybride appelé **LBWDP (Load Balancing over Widest Disjoint Paths)** qui tire partie des avantages de la sélection du chemin de WDP et du mécanisme de division du PER.

LBWDP utilise l'algorithme existant WDP (Widest Disjoint Paths) qui se concentre sur la sélection de bons chemins. Cette approche repose principalement sur deux concepts: largeur du trajet et distance du trajet. La largeur de chemin est un moyen de détecter les goulots d'étranglement dans le réseau et de les éviter si possible. L'algorithme WDP sélectionne les chemins candidats en fonction du calcul de la largeur des bons chemins disjoints en ce qui concerne les liens de goulot d'étranglement. La largeur d'un chemin est définie comme la largeur de bande résiduelle de son lien de goulot d'étranglement. Le principe de WDP est de sélectionner un nombre limité de chemins. Un chemin est ajouté

au sous-ensemble de bons chemins si son inclusion augmente la largeur de ce sous-ensemble. À l'inverse, un chemin est supprimé si cela ne réduit pas la largeur du sous-ensemble de bons chemins. Pour l'étape de fractionnement du trafic.

LBWDP utilise l'algorithme PER (Prediction of Effective Repartition). Il consiste en deux étapes: calculer une probabilité de distribution et sélectionner un chemin à l'aide de la méthode Gradient avec un trafic en bande passante en contrainte. Ce système d'ingénierie du trafic sera utile pour réduire la probabilité d'encombrement en minimisant l'utilisation du lien le plus utilisé du réseau.

Les auteurs **[K. Abboud *et al*, 2008]** proposent un **PEMS (PEriodic Multi-Step Routing Algorithm)**. Son objectif est de développer une méthode de routage qui optimise la différenciation des services expérimentés des classes de trafic en termes de deux métriques, délai et bande passante disponible. Les classes gérées par PEMS sont les suivantes: Expedited Forwarding (EF) pour le trafic sensible aux retards, comme le trafic VoIP, Assured Forwarding (AF) pour la bande passante douce et le trafic garantissant la perte, comme la vidéo, et Best-Effort Forwarding (BEF) pour assurer la qualité minimale au mieux les services comme ftp ou e-mail. Le PEMS vise essentiellement à minimiser l'utilisation maximale de la liaison sur le réseau tel que l'algorithme LBWDP et à offrir en outre une qualité de service différente à chaque classe, en particulier pour garantir un faible délai à la classe EF.

Dans **[Fenglin Li *et al*, 2012]**, Les auteurs proposent un algorithme d'équilibrage de charge utilisant un chemin de déviation **LBDP(Load Balancing Algorithm using Deviation Path)** bascule un flux sélectionné en chemin de déviation au moment où un encombrement est sur le point de se produire, il utilise l'algorithme du chemin le plus court pour sélectionner le chemin de déviation, mais cet algorithme peut générer beaucoup plus de trafic sur certains nœuds que d'autres, il choisit également le chemin de déviation qualifié parmi un nombre limité de chemins sélectionnés; Dans le protocole LBDP, un chemin de déviation qualifié est trouvé uniquement lorsqu'aucune boucle n'existe et que la bande passante disponible est suffisamment large; La probabilité de sélectionner des chemins non qualifiés ici est assez élevée. En l'absence de chemin de déviation qualifié trouvé, la congestion se produira.

- ✓ Cependant, ces techniques se sont concentrées sur Load Balancing en déplaçant le trafic des liens encombrés vers des liens moins utilisés en utilisant un réacheminement centralisé. Ces techniques partagent la propriété que le réacheminement est effectué de bout en bout, ce qui signifie que le réacheminement est effectué complètement pour tout le chemin, de la source à la destination. Pour les grands réseaux, cela serait inefficace pour les raisons suivantes:

  - L'établissement d'une nouvelle route alternative du noeud source à la destination nécessite que toutes les informations d'état de la topologie soient disponibles au niveau des nœuds sources, ce qui rend la solution non évolutive.

  - Une fausse redirection au niveau des nœuds sources se produirait en raison de retards dans le transfert des informations d'état du réseau aux nœuds sources.

  - Lente adaptation à la congestion, notamment avec une dynamique de réseau rapide.

2.8. Conclusion :

Contrôler le trafic au sein des réseaux MPLS et minimiser les problèmes telle que la congestion sont devenus un impératif pour de bonnes performances du réseau et une efficacité et une qualité de service améliorées. La technologie de l'ingénierie du trafic a été émergée pour résoudre ce problème et a joué un grand rôle à cet égard. Les algorithmes Load Balancing dans les réseaux MPLS présentés précédemment en tant que modèles d'ingénierie du trafic, ont effectivement contribué à améliorer les performances et l'efficacité du réseau, chacun à sa manière et en fonction de ses conditions.

Cependant, cela ne signifie pas qu'ils soient complets, sans faille, chacun des algorithmes possède des avantages pour servir un aspect particulier, et des défauts n'aident pas les autres aspects.

Dans ce contexte, l'objectif principal de nos travaux est de proposer une nouvelle approche de gestion du flux pour réduire de la congestion en utilisant les notions de Load Balancing.

Chapitre

3 la présentation de nouvel algorithme dans les réseaux MPLS basé sur le mécanisme Load Balancing.

Contenu

3.1 Introduction

À travers ce qui a été mentionné dans les deux chapitres précédents sur les concepts et bases des réseaux MPLS, ainsi que sur ses avantages et applications telles que l'architecture de trafic, qui joue un rôle crucial dans l'amélioration des performances et de l'efficacité du réseau. Nous avons également discuté du problème le plus important auquel les réseaux MPLS peuvent être confrontés, à savoir le surpeuplement. Pour remédier à ces problèmes et les réduire, les chercheurs ont mis au point de nouvelles approches et méthodes pour les résoudre.

Dans ce sujet, nous aborderons dans ce chapitre un nouvel algorithme basé sur le mécanisme **L**oad **B**alancing pour les réseaux MPLS (**MPLS** network) appelé **LBMPLS**, l'idée de base est de dévier efficacement les LSP des liens les plus occupés vers des liens non exploités. Nous utilisons le concept de seuil pour déterminer la congestion et considérons la priorité du flux pour accélérer le temps de transfert.

Dans ce chapitre, nous allons entrer dans la description de cet algorithme, puis nous écrivons l'algorithme, après quoi nous dessinons un diagramme résumant cet algorithme, nous terminons ce chapitre par une conclusion.

3.2 Description de l'algorithme LBMPLS:

K est un réseau MPLS contenant deux hôtes: un noeud source (hôte1) et un noeud de destination (hôte2). Lorsque hôte1 envoie un paquet à hôte2, le paquet est routé sur le réseau en passant par les routeurs LSRs (LSR1, LSR2, … .LSR7)

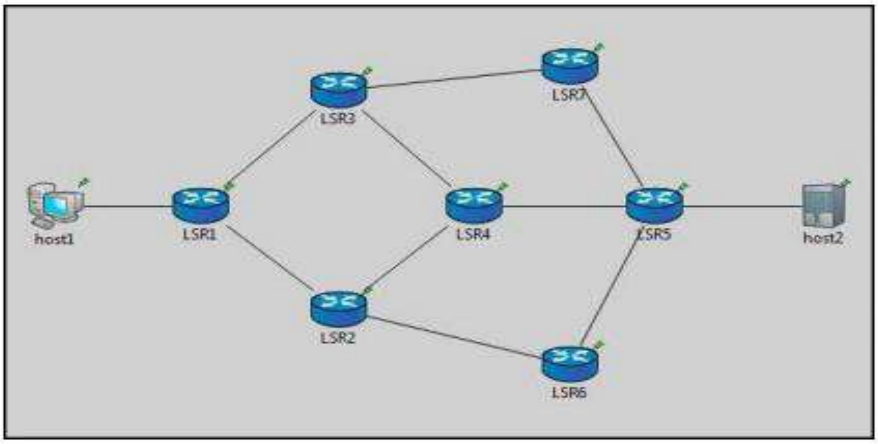

Figure 3.1 : K1: MPLS-TE Network.

3.2.1 Créez une base de données pour la topologie du réseau :

Nous supposons que G (X, U) est un multigraphe d'un fragment du réseau K, où X = {x1, x2, x3,… xn} l'ensemble des vertices qui sont des nœuds de réseau (routeurs LSR) et U = {u1, u2, u3… un} un ensemble d'arcs représentant les branches du réseau sont présentés par des chemins réseau. Nous utilisons u (i, j) pour désigner le lien entre le noeud i et le noeud j.

Une base de données de topologie de réseau est une liste de précédents et de suivantes de chaque nœud du réseau, et son symbole **BDMNET**

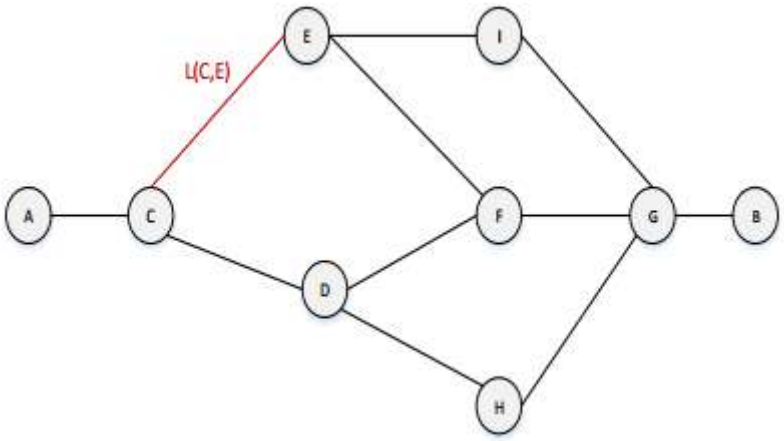

Figure 3.2 : K2: Réseau MPLS-TE.

Nœuds	Liste de Précédents	Liste de Suivantes
A	**/**	**C**
C	**A**	**D - E**
D	**C**	**F - H**
E	**C**	**F - I**
F	**D – E**	**G**
I	**E**	**G**
H	**D**	**G**
G	**F - I – H**	**B**
B	**G**	**/**

Tableau 3.1 : BDMNET

Exemple : **Suiv [G] = {B, Suiv [B]}**

Pré [G] = {F, I, H, Pré [F], Pré [I], Pré [H]}

3.2.2 Trouver un chemin de déviation :

Pour définir le nœud de déviation d'un lien u, nous allons utiliser BDMNET définis ci-dessus.

Nous supposons que **u1** est un chemin partant des nœuds x1 à xi

u1 = {x1, x2… .xk,… xi}, et qu'il existe un autre chemin appelé **u2** de x'1 à x'i

u2 = {x'1, x'2,… ..x'k ,… .x'i}.

Si x1, x2,… . xk = x'1, x'2,… .x'k **et** xk + 1 ∈ Suiv [xk], **et** x'k + 1 ∈ Suiv [x'k], **et** xk + 1 ≠ x'k + 1, alors le nœud xk est le nœud de déviation et u2 est un chemin de déviation de u1 contre le noeud xk**.**

3.2.3 Définition des structures de données :

- Nous construisons ensuite une structure de données nommée ER-LSP pour décrire une réservation de ressource d'un ER-LSP.

ER-LSP = {<Trafic-ID>, <LSP-ID>, <UB>}

- **Trafic-ID** : est l'identification du trafic qu'ER-LSP.
- **LSP-ID** : est l'identification de ce chemin ER-LSP,
- **UB** : est le débit de bande passante utilisé d'ER-LSP.

- Nous supposons que **ɣ** est un seuil, nous construisons ensuite une structure de données pour les LSP encombrés nommée **CL. CL** compose de tous les chemins LSP dont le UB est supérieur ou égal à ɣ (UB> = ɣ).

CL = {ER-LSP1, ER-LSP2, ER LSP3…}

Selected-LSP est le ER-LSP avec le plus grand UB en CL. Chemin le plus encombré.

- Nous construisons une autre structure de données appelée **ST (trafic sélectionné)** qui contient tous les flux de trafic contenus dans le SelectedLSP, tous les identifiants de trafic attribués à Selected-LSP

ST = {Trafic-ID1, Trafic-ID2…}

Nous supposons maintenant que Selected-LSP est un chemin partant du noeud ni au noeud nj, Selected-LSP = ni, n1, n2… nk… nj. Et un autre chemin du noeud ni au noeud nj appelé P existe aussi et P = ni, n1, n2… nk… nj. Si ni, n1, n2… nk = n'i, n'1, n'2… n'k. Et nk + 1 ≠ n'k + 1, le nœud nk + 1 est le nœud de déviation et P le chemin de déviation de Selected-Path par rapport au nœud nk + 1.

- Après cela, nous construisons une collection **NDC** de chemins de déviation négociée, où **NDC** compose tous les chemins de déviation de LSP sélectionné.

NDC = {P1, P2, P3…}

RB: La bande passante demandée d'un flux de trafic.

FC: capacité libre.

3.3 L'algorithme LBMPLS :

L'ensemble des itinéraires trouvés correspond aux conditions indispensables à la réalisation du passage suivant : Lorsqu'un nœud d'entrée reçoit un nouveau flux de trafic, il va:

UB: Bande passante utilisée.

FC: capacité libre.

γ : seuil déterminé.

CL : ensemble de chemins encombrés.

Selected-LSP: MOST! Chemin encombré.

ST : ensemble des flux passant sur Selected-LSP

En attente de trafic : trafic qui sera redirigé sur le chemin de déviation

NDC : ensemble de chemins de déviation.

1. Vérifiez périodiquement les taux **UB** des liaisons adjacentes, s'il existe un ER-LSP avec (FC < ɣ), puis établissez **CL**, passez à **(3),** sinon passez à **(2).** Nous vérifions UB et non FC car comme cela le réseau MPLS fonctionne

2. CL est NULL → aucune congestion n'est sur le point de passer, allez à **(1).**

3. ** // nous choisissons le chemin le plus congestionné …… Sélectionnez ERLSP dont le FC est le plus petit de CL et attribuez sa valeur à Selected-LSP aller à (4)

4. Établir ST.

5. Sélectionnez le flux de trafic de ST dont la priorité est inférieure avec un minimum de contrôle, allez à (6)

6. Attribuez l'ID de trafic du flux de trafic au trafic en attente, allez à (7)

7. Etablissez NDC: (noeud Trouver que les chemins de déviation ont existé): ** // Trouvez des successeurs que le noeud de déviation, après avoir sélectionné les chemins de déviation à partir de ce noeud et allant à la même destination ** //, allez à (8)

8. Sélectionnez P dont le FC est le plus grand dans NDC, si (RB < FC et UB + RB > ɣ), puis passez à (9), sinon passez à (10).

9. Affectez l'ID de trafic de P au trafic en attente et basculez le trafic en attente à l'adresse (13).

10. Supprimez P de NDC. Allez à (11).

11. La capacité est insuffisante pour satisfaire la demande de bande passante du flux de trafic sélectionné, le transport échoue et l'algorithme se termine.

12. Supprimer le trafic en attente de ST. allez à (14).

13. Si **FC** de LSP sélectionné ≤ **PB** - **ɣ** , passez à (5), sinon passez à (14)

14. Supprimer Selected-LSP de CL, l'algorithme se termine avec succès, aller à (1)

- Cet algorithme vise à réduire la congestion sur le réseau et à éviter que la congestion ne se reproduise.

3.4 Modèle LBMPLS :

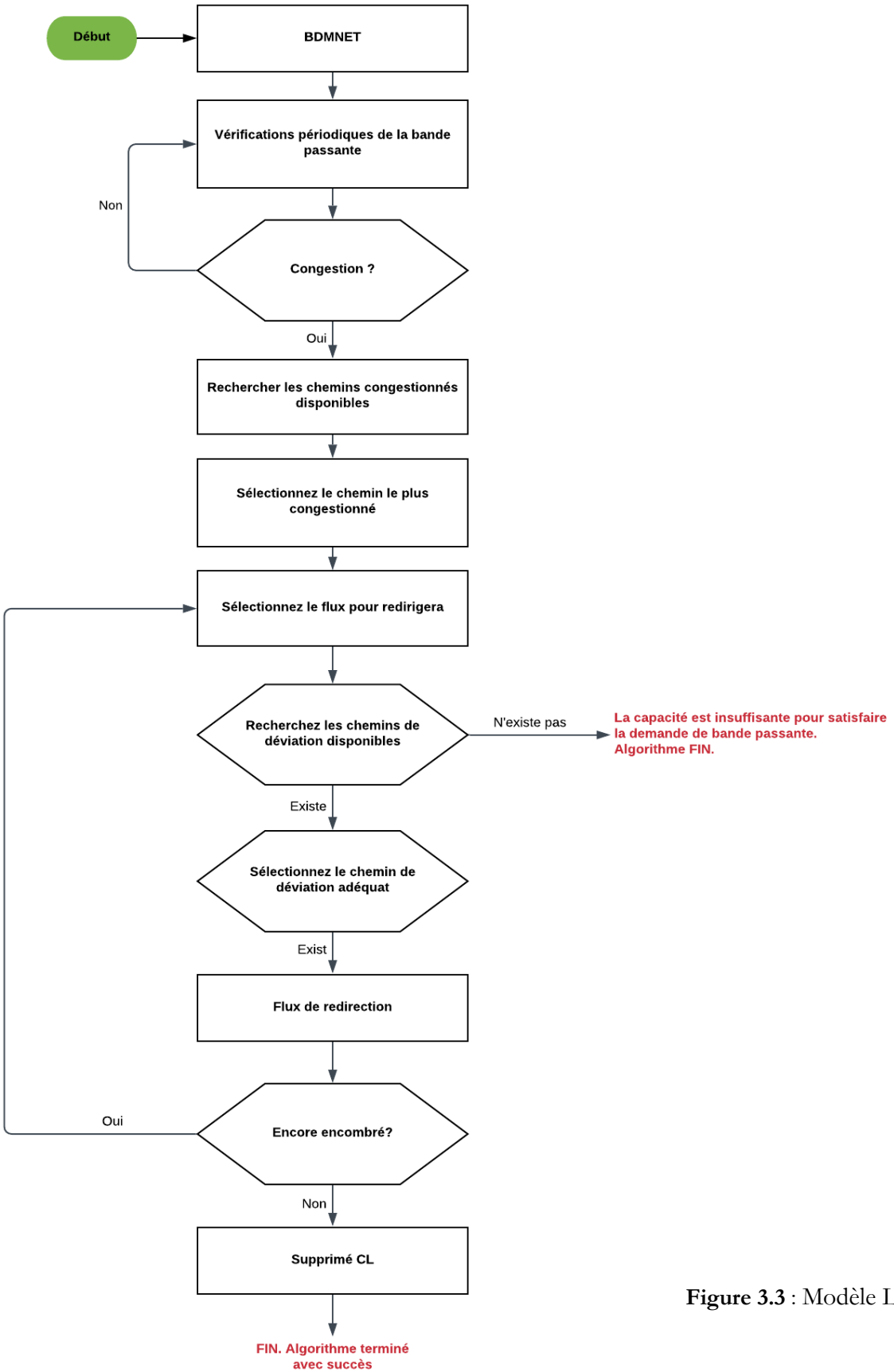

Figure 3.3 : Modèle LBMPLS

3.5 Conclusion :

Plusieurs approches et algorithmes ont été proposés pour tenter de réduire le problème de congestion des réseaux MPLS. Chacune avec des points positifs et des inconvénients en fonction des caractéristiques du réseau et de ce qui est le plus important aux yeux de chaque cas.

Dans ce chapitre, nous avons discuté de la proposition et de la description d'un nouvel algorithme d'équilibrage de charge basé sur la déviation du chemin encombré par rapport à un chemin inexploité. En fonction du concept de seuil pour la détermination de la congestion, en plus de considérer la priorité du flux pour accélérer le temps de transfert.

Afin de vérifier la validité de cette proposition et de connaître l'efficacité de cet algorithme et de ce qui le distingue des autres, il doit être appliqué, imité et comparé à d'autres méthodes jusqu'à ce que nous le trouvions. C'est ce que nous allons mentionner dans le prochain chapitre.

Chapitre 4

L'implémentation et Simulation

Contenu

4.1 Introduction

À travers ce que nous avons discuté dans le chapitre précédent, nous avons mentionné la manière dont notre algorithme et ses caractéristiques les plus importantes dépendent. Pour être efficaces dans la réduction du surpeuplement et dans l'accélération du transport, ils doivent être testés, étudiés, analysés et comparés.

Par conséquent, dans ce chapitre, nous allons essayer d'implémenter et d'exécuter notre algorithme et nos simulations sur l'OMNET.

Nous commencerons ce chapitre en définissant OMNET, puis en nous souvenant d'INET, de ses propriétés et de ses composants, après nous appliquerons l'algorithme à OMNET++, puis nous essaierons de l'émuler et de l'essayer, pour conclure par une conclusion finale.

4.2 Simulateur de réseau OMNeT++ : [1]

OMNeT++ (Objective Modular Network) est un outil de simulation à événements discrets, open source écrit en C++. OMNeT++ est un espace de simulation modulaire à base de composants, avec hiérarchique, et des modules personnalisables. Son domaine d'application principal est celui des réseaux de communication. OMNeT++ basée sur le concept orientée objet qui permet de développer des scénarios de simulations en utilisant l'ensemble d'outils et de modules qu'il offre.

L'OMNeT++ utilise un langage modulaire nommé NEtwork Descrption (NED), qui permet la modélisation de systèmes complexes par assemblage de plusieurs modules élémentaires en ensemble. L'architecture du réseau est décrite ci-dessous de façon d'organisation hiérarchique, avec les composants suivants (figure 4.1):

- **Module simple**: est un composant unique qui nous permet de définir comportement algorithmique.

- Canal: est le lien qui assure la liaison entre les modules
- Module composé: une structure composée par des modules simples

Les Modules actifs, également nommés des modules simples sont dérivés d'une classe de base, cSimpleModule. Ces modules actifs, peuvent être regroupés de façon hiérarchie afin de former des modules composés. Illimité différentes hiérarchiques couches du module composé peuvent être créés afin de faire un module de système. Les modules communiquent entre eux en utilisant des messages ; ceux qui sont envoyés soit par connexion directe entre les modules (input/output gates) ou directement aux modules de destination. Pour chaque module simple correspond un fichier .cc et un fichier .h.

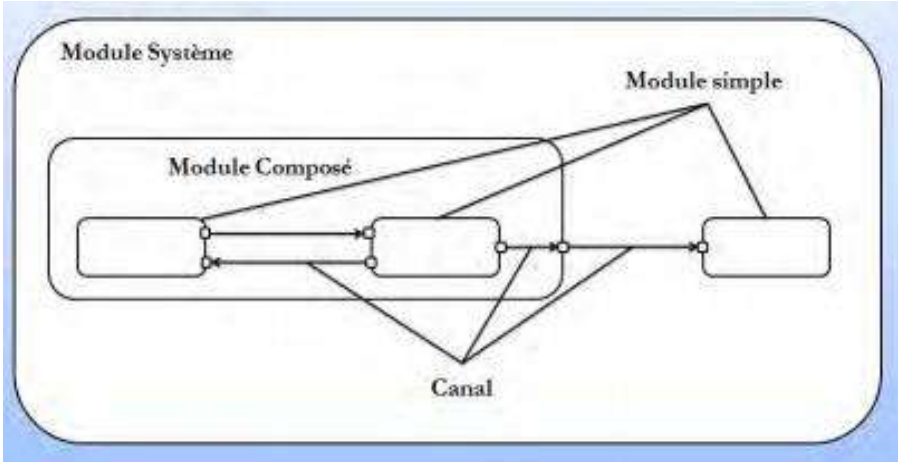

Figure 4.1 : Structure modulaires d'OMNeT++.

Une fois que des modules de différents modules sont construits, le modèle de réseau peut être défini à l'aide langage Description de réseau (NED) particulière à OMNeT++. Langage NED est utilisé pour décrire quel module sont utilisés pour définir la topologie du réseau, les interconnexions entre les modules, et les paramètres des modules composés. La Topologie du réseau peut également être définie à l'aide de l'interface utilisateur d'OMNeT++, qui se traduit automatiquement en code les différents modules. La figure 4.2 montrent comme par exemple la façon dont deux hôtes sont connectés ensemble, en utilisant l'interface graphique, ou la langue NED.

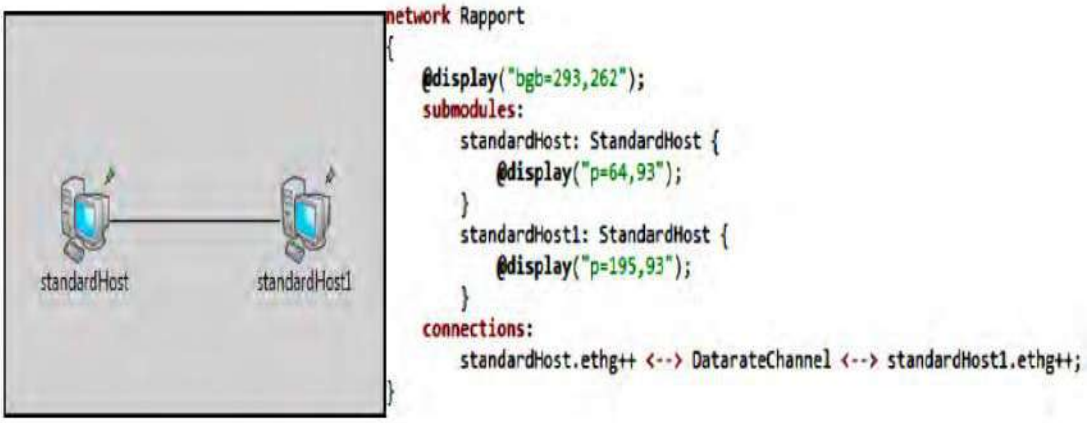

Figure 4.2 : Description du réseau (Graphique (à gauche) et code NED (à droite)).

Une fois que le modèle de simulation décrit dans le NED et le comportement du module écrit en C++ sont terminés, les paramètres de simulation sont stockés en utilisant les fichiers .INI, tandis que le «backbone» de la simulation (fichiers C ++ et NED) est supposé rester inchangé, certains paramètres varient en fonction de l'expérience. Par conséquent, tous les paramètres sont enregistrés séparément dans un fichier .ini L'utilisation de ce différentiation nous permet d'avoir plusieurs scénarios, d'observer comment la simulation se comporter selon les différentes entrées sans avoir à modifier les fichiers de modèles.

OMNeT++ est un environnement complet de simulation, lorsque les modules standards sont génériques et souple pour tenir dans un scénario de base. Cependant, du fait de sa modularité chaque module peut être facilement réutilisé dans un autre module, ce qui a conduit à la diversité de développement au l'environnement de simulation.

4.3 INET Framework : [2]

4.3.1 Définition :

INET Framework est une bibliothèque de modèles à code source ouvert pour l'environnement de simulation OMNeT ++. Il fournit des protocoles, des agents et d'autres modèles aux chercheurs et aux étudiants travaillant avec des réseaux de communication. INET est particulièrement utile lors de la conception et de la validation de nouveaux protocoles, ou lors de l'exploration de scénarios nouveaux ou exotiques.

INET prend en charge une vaste gamme de réseaux de communication, notamment les réseaux câblés, sans fil, mobiles, ad hoc et à capteurs. Il contient des modèles pour la pile Internet (TCP, UDP, IPv4, IPv6, OSPF, BGP, etc.), des protocoles de couche liaison (Ethernet, PPP, IEEE 802.11, divers protocoles MAC de capteurs, etc.), une prise en charge améliorée de la couche physique sans fil. , Protocoles de routage MANET, DiffServ, MPLS avec signalisation LDP et RSVP-TE, plusieurs modèles d'application et de nombreux autres protocoles et composants. Il prend également en charge la mobilité des nœuds, la visualisation avancée, l'émulation de réseau, etc.

4.3.2 conçu pour l'expérimentation :

INET est construit autour du concept de modules qui communiquent par la transmission de messages. Les agents et les protocoles réseau sont représentés par des composants, qui peuvent être librement combinés pour former des hôtes, des routeurs, des commutateurs et d'autres périphériques réseau. Les nouveaux composants peuvent être programmés par l'utilisateur et les composants existants ont été écrits de manière à être faciles à comprendre et à modifier. INET bénéficie de l'infrastructure fournie par OMNeT ++. Outre l'utilisation des services fournis par le noyau et la bibliothèque de simulation OMNeT ++ (modèle de composant, paramétrage, enregistrement des résultats, etc.), cela signifie également que les modèles peuvent être développés, assemblés, paramétrés, exécutés et leurs résultats valorisés dans le confort de l'utilisateur. OMNeT ++ Simulation IDE ou à partir de la ligne de commande. INET Framework est géré par l'équipe OMNeT ++ pour la communauté, à l'aide de correctifs et de nouveaux modèles fournis par les membres de la communauté.

4.3.3 INET et MPLS :

INET fournit une assistance de base pour la création de simulations MPLS. Il fournit des modèles pour les protocoles MPLS, LDP et RSVP-TE et leurs structures de données associées, ainsi que des modèles de routeur compatibles MPLS préassemblés. MPLS requiert la présence d'un module LIB (Label Information Base)

dans le routeur auquel il accède via des appels de fonction C ++. Les implémentations du protocole de contrôle MPLS (par exemple, le module RSVP) gèrent les informations en LIB via des appels C ++. Les modules de base sont:

- **Mpls** : implémente le protocole MPLS.
- **LibTable** : contient le LIB (Label Information Base).
- **Ldp** : implémente le protocole de signalisation LDP pour MPLS.
- **RsvpTe** : implémente le protocole de signalisation RSVP-TE pour MPLS.
- **Ted** : contient la base de données d'ingénierie du trafic.
- **LinkStateRouting** : est un simple protocole de routage à état de liens.
- **RsvpClassifier** : est un classificateur d'entrée configurable pour MPLS.

4.4 L'implémentation de l'algorithme:

Dans le chapitre précédent, nous avons présenté la nouvelle proposition et le nouvel algorithme d'équilibrage de charge basés sur la déviation du chemin encombré dans un chemin inexploité. Selon le concept de seuil pour déterminer la congestion, en plus d'examiner la priorité de flux pour accélérer le temps de transfert.

L'algorithme est résumé comme suit:

Si FC (P) ≤ (PB(P) - ɣ) alors:

- **Mettez P dans CL.**
- **Rechercher des chemins déviés de P, établir NDC (P).**
- **Établir ST.**
- **Sélectionnez Flux ∈ ST et RB (Flux) = min RB et priorité (Flux) = priorité inférieure**

  S'il existe, D ∈ NDC, RB (Flux) ≤ FC (D) et UB (D) + RB (Flux) ≤ ɣ alors

  - **Activez le flux sélectionné dans D.**

Sinon, il n'y a pas de congestion.

L'OMNeT++ est basé sur le langage C, alors nous allons implémenter notre algorithme dans le langage C. Nous établissons une classe au nom de LBMNET comme suit : premièrement, nous allons créer des sous-classes :

1- Classe ER_LSP :

```
//.h file code:

#include <string>
class ER_LSP
{
private:
    std::wstring Path_ID;
    std::wstring Traffi_ID;
    int UB = 0;
public:
    ER_LSP(const std::wstring &Path_ID,
           const std::wstring &Traffic_ID, int UB);
    virtual std::wstring getPath_ID();
    virtual void setPath_ID(const std::wstring &path_ID);
    virtual std::wstring getTraffi_ID();
    virtual void setTraffi_ID(const std::wstring &traffi_ID);
    virtual int getUB();
    virtual void setUB(int uB);
};
```

Figure 4.3 : Classe ER_LSP.h

```
//.cpp file code:

ER_LSP::ER_LSP(const std::wstring &Path_ID,
               const std::wstring &Traffic_ID, int UB)
{
    this->Path_ID = Path_ID;
    this->Traffi_ID = Traffic_ID;
    this->UB = UB;
}
std::wstring ER_LSP::getPath_ID(){ return Path_ID;}
void ER_LSP::setPath_ID(const std::wstring &path_ID)
        { Path_ID = path_ID;}
std::wstring ER_LSP::getTraffi_ID(){ return Traffi_ID;}
void ER_LSP::setTraffi_ID(const std::wstring &traffi_ID)
        {  Traffi_ID = traffi_ID;}
int ER_LSP::getUB() {   return UB;}
void ER_LSP::setUB(int uB){ UB = uB;}
```

Figure 4.4 : Classe ER_LSP.cc

2- **Classe Trafic** :

```
//.h file code:

#include <string>
class Traffics
{
private:
    std::wstring Traffic_ID;
    int Traffic_UB = 0;
    int Priority = 0;
    int Traffic_langth = 0;
    // Constructor
public:
    Traffics(const std::wstring &Traffic_ID, int Traffic_UB,
             int Priority, int Traffic_langth);
    //   Get & Set
    virtual std::wstring getTraffic_ID();
    virtual void setTraffic_ID(const std::wstring &traffic_ID);
    virtual int getTraffic_UB();
    virtual void setTraffic_UB(int traffic_UB);
    virtual int getTraffic_langth();
    virtual void setTraffic_langth(int traffic_langth);
    virtual int getPriority();
    virtual void setPriority(int priority);
};
```

Figure 4.5 : Classe Trafic.h

```
//.cpp file code:

Traffics::Traffics(const std::wstring &Traffic_ID,
                   int Traffic_UB, int Priority, int Traffic_langth)
{
    this->Traffic_ID = Traffic_ID;
    this->Traffic_UB = Traffic_UB;
    this->setPriority(Priority);
    this->Traffic_langth = Traffic_langth;
}
std::wstring Traffics::getTraffic_ID() {    return Traffic_ID;}
void Traffics::setTraffic_ID(const std::wstring &traffic_ID)
          {  Traffic_ID = traffic_ID;}
int Traffics::getTraffic_UB() { return Traffic_UB;}
void Traffics::setTraffic_UB(int traffic_UB){Traffic_UB = traffic_UB;}
int Traffics::getTraffic_langth() { return Traffic_langth;}
void Traffics::setTraffic_langth(int traffic_langth)
          {  Traffic_langth = traffic_langth;}
int Traffics::getPriority() {   return Priority;}
void Traffics::setPriority(int priority){ Priority = priority;}
```

Figure 4.6 : Classe Trafic.cc

3- Classe Path :

```
//.h file code:
#include <vector>
class Paths
{ private:
    int Path_ID = 0;
    int Path_UB = 0;
    int FC = 0;
    std::vector<int> Path_nodes;
    std::vector<Traffics*> ST_Trafics;
    // Constructor
public:
    Paths();
    Paths(std::vector Path_nodes);
    Paths(int Path_ID, int Path_UB, int FC, std::vector<int>
          &Path_nodes, std::vector<Traffics*> &ST_Trafics);
    //    Get & Set
    virtual int getPath_ID();
    virtual void setPath_ID(int path_ID);
    virtual int getPath_UB();
    virtual void setPath_UB(int path_UB);
    virtual std::vector<int> getPath_nodes();
    virtual void setPath_nodes(std::vector<int> &path_nodes);
    virtual std::vector<Traffics*> getPath_ST();
    virtual void setPath_ST(std::vector<Traffics*> &path_ST);
    virtual int getFC();
    virtual void setFC(int fC);
};
```

Figure 4.7 : Classe Path.h

```
//.cpp file code:

Paths::Paths(){}
Paths::Paths(std::vector Path_nodes)
            {  this->Path_nodes = Path_nodes; }
Paths::Paths(int Path_ID, int Path_UB, int FC, std::vector<int>
              &Path_nodes, std::vector<Traffics*> &ST_Trafics)
{
    this->Path_ID = Path_ID;
    this->Path_UB = Path_UB;
    this->FC = FC;
    this->Path_nodes = Path_nodes;
    this->ST_Trafics = ST_Trafics;
}
int Paths::getPath_ID() {   return Path_ID;}
void Paths::setPath_ID(int path_ID) {   Path_ID = path_ID;}
int Paths::getPath_UB() {   return Path_UB;}
void Paths::setPath_UB(int path_UB) {   Path_UB = path_UB;}
std::vector<int> Paths::getPath_nodes() {   return Path_nodes;}
void Paths::setPath_nodes(std::vector<int> &path_nodes)
             { Path_nodes = path_nodes;}
std::vector<Traffics*> Paths::getPath_ST(){ return ST_Trafics;}
void Paths::setPath_ST(std::vector<Traffics*> &path_ST)
          { ST_Trafics = path_ST;}
int Paths::getFC(){ return FC;}
void Paths::setFC(int fC) { FC = fC;}
```

Figure 4.8 : Classe Path.cc

4- **Classe LBMPLS** :

```
//.h file code:

#include <string>
#include <vector>
class LBMNET
{
private:
    static Paths *SRP_solution;
    int y = 0;
    std::vector BD;
    Paths *Path;

public:
    LBMNET(int y, std::vector BD, Paths *path);
    static void Change_BD(std::vector path,
           std::vector<std::vector> &BD, int node);
    // ----  Most SRP Congestion ----
    static int Most_Path(std::vector<Paths*> &l);
    static int min_List(std::vector<int> &l);
    static void main(std::vector<std::wstring> &args);
    virtual Paths *getSRP_solution();
    virtual void setSRP_solution(Paths *sRP_solution);
};
```

```
//.cpp file code:

Paths *LBMNET::SRP_solution = new Paths();
LBMNET::LBMNET(int y, std::vector BD, Paths *path)
{
    this->y = y;
    this->BD = BD;
    this->Path = path;
}
void LBMNET::Change_BD(std::vector path, std::vector<std::vector> &BD, int node)
{

int LBMNET::Most_Path(std::vector<Paths*> &l)
{

int LBMNET::min_List(std::vector<int> &l)
{

void LBMNET::main(std::vector<std::wstring> &args)
{
```

```
std::wstring GraphPrintAllPaths::print(Graph *graph, int start, int end,
                                       const std::wstring &path, std::vector<bool> &visited){
    std::wstring newPath = path + L"->" + std::to_wstring(start);
    visited[start] = true;
    std::list<Node*> list = graph->adjacencyList[start];
    for (int i = 0; i < list.size() ; i++)
    {   Node *node = list.get(i);
        if (node->destination != end && visited[node->destination] == false){
            print(graph,node->destination,end,newPath,visited);}
        else if (node->destination == end){
            d = d + newPath + L"->" + node->destination;
        }
    }
    //remove from path
    visited[start] = false; return d;
}
std::vector GraphPrintAllPaths::printAllPaths(Graph *graph, int start, int end){
    std::vector<bool> visited(graph->vertices);
    visited[start] = true;
    std::wstring p = print(graph, start, end, L"", visited);
    std::wstring r = L"->";
    p = p.replaceAll(r, L".");
    std::vector l = ALLPaths(p, 0);
    std::wcout << l << std::endl;
    return l;
}
```

Figure 4.9: Classe LBMPLS -1-

La fonction GraphPrintAllPaths : Cette fonction retourne un chemin possible à partir du nœud où le problème est survenu Congestion au noeud final, formant ainsi notre liste de NDC.

```
void LBMNET::main(std::vector<std::wstring> &args)
{
    int y = 70;
    std::vector BD = std::vector<std::vector>();
    std::vector CL = std::vector<Paths*>();
    std::vector<std::vector> ND;
    Paths *path = new Paths();
    LBMNET *lbmnet = new LBMNET(y,BD,path);

    //----- Test threshold -----------
    if (path->getFC() <= path->getFC() - y){
        CL.push_back(path); }
```

```
//------- Find to solution
while (!CL.empty())
{
    Paths *SRP = static_cast<Paths*>(CL[Most_Path(CL)]);
    std::vector<std::vector> db = static_cast<std::vector<std::vector>>(BD);
    Change_BD(SRP->getPath_nodes(), db,SRP->getPath_ID());
    Graph *G = new Graph(db.size());
    GraphPrintAllPaths *m = new GraphPrintAllPaths(db.size(), G, db);
    ND = m->printAllPaths(G, SRP->getPath_ID(),SRP->getPath_nodes()->get(SRP->getPath_nodes()->size() - 1));
    std::vector<Paths*> NDC;
    for (int i = 0;i < ND.size();i++)
    {   Paths tempVar(ND[i]);
        NDC.push_back(&tempVar);
    }
    std::vector<int> D = std::vector();
    for (int i = 0; i < SRP->getPath_ST()->size();i++)
    {  D.push_back(SRP->getPath_ST()->get(i).getPriority() - SRP->getPath_ST()->get(i).getTraffic_UB());}
    int index_STi = min_List(D);
    for (int i = 0; i < NDC.size();i++)
    {  if ((NDC[i]->getFC() >= D[index_STi]) && (NDC[i]->getFC() + SRP->getPath_ST()->get(i).getTraffic_UB() <= y)){
            lbmnet->setSRP_solution(NDC[i]);}
    }
}
```

Figure 4.10: Classe LBMPLS -2-

4.5 Simulation :

Nous incluons la simulation d'une allocation de bande passante statique en utilisant le protocole RSVP-TE pour la réservation de bande passante et MPLS en utilisant OMNET ++ 5.4.1 IDE intégré au cadre INET. Le simulateur de réseau OMNET ++ 5.4.1 en lui-même ne prend pas en charge la mise en réseau MPLS. Le package supplémentaire d'INET est nécessaire pour simuler les réseaux MPLS.

Les étapes suivies dans la simulation :

1- Nous créons un nouveau projet OMNET ++ dans le dossier souhaité

2- Dans la barre de propriétés du projet, sélectionnez l'onglet Références et cochez la case INET pour indiquer que le projet utilisera les fonctions du package INET.

3- Ensuite, nous créons un fichier NED (Network Definition File) sous ce dossier pour définir la structure du réseau à l'aide des outils d'interface graphique du framework. Le fichier NED dans l'exemple de réseau RSVPTE4.ned donné ci-dessous, que nous avons extrait du tunnel de dossiers inet / examples / mpls / testte à des fins d'analyse, ressemble à ce qui suit.

Chaque LSR est un LSR RSVP qui comprend les modules rsvp, linkStateRouting, couche réseau, mpls, ppp, table de routage, table d'interface, libTable, etc.
Chaque hôte est un hôte standard avec une adresse IP et capable d'émettre et de recevoir des paquets.

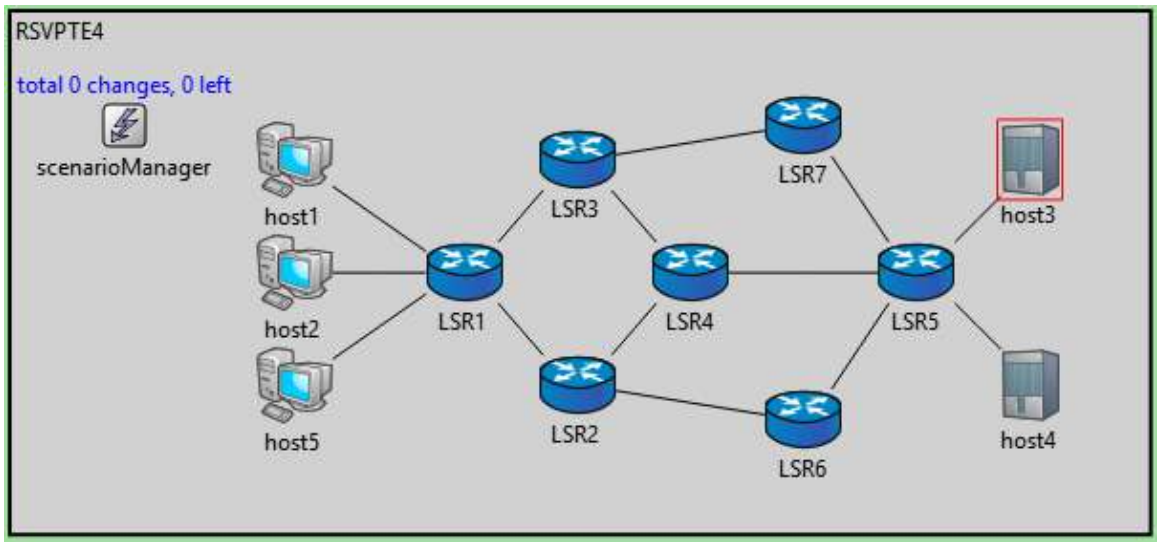

Figure 4.11 : Instantané de RSVETE4.ned

4- Nous créons un fichier 'rt' pour chacun des hôtes et des LSR du réseau, qui représente les informations de la table de routage pour les nœuds respectifs. Pour l'exemple de réseau, nous avons sept fichiers «rt» définis pour les LSR et cinq pour les hôtes standard. Pour comprendre les résultats et l'analyse présentés dans les chapitres suivants, nous devons montrer la structure des fichiers 'rt'.

Chaque hôte a une interface, définie par ppp0 et un inet addr.

host1.rt:

```
ifconfig:
name: ppp0  inet_addr: 10.0.1.1     MTU: 1500   Metric: 1
ifconfigend.

route:
10.1.1.1    *           255.255.255.255   H     0     ppp0
default:    10.1.1.1    0.0.0.0           G     0     ppp0
routeend.
```

host2.rt:

```
ifconfig:
name: ppp0  inet_addr: 10.0.2.1     MTU: 1500   Metric: 1
ifconfigend.

route:
10.1.1.1    *           255.255.255.255   H     0     ppp0
default:    10.1.1.1    0.0.0.0           G     0     ppp0
routeend.
```

host3.rt:

```
ifconfig:
name: ppp0  inet_addr: 10.2.1.1     MTU: 1500   Metric: 1
ifconfigend.

route:
10.1.5.1    *           255.255.255.255   H     0     ppp0
default:    10.1.5.1    0.0.0.0           G     0     ppp0
routeend.
```

host4.rt:

```
ifconfig:
name: ppp0  inet_addr: 10.2.2.1     MTU: 1500   Metric: 1
ifconfigend.

route:
10.1.5.1    *           255.255.255.255   H     0     ppp0
default:    10.1.5.1    0.0.0.0           G     0     ppp0
routeend.
```

host5.rt:

```
ifconfig:
name: ppp0  inet_addr: 10.0.3.1     MTU: 1500   Metric: 1
ifconfigend.

route:
10.1.1.1    *           255.255.255.255   H     0     ppp0
default:    10.1.1.1    0.0.0.0           G     0     ppp0
routeend.
```

LSR1.rt:

Puisqu'il y a cinq interfaces, cinq adresses inet sont définies. L'adresse d'entrée du routeur serait 10.1.1. *. Les itinéraires sont définis pour chacun de ses voisins et les interfaces auxquelles ils font référence.

```
ifconfig:
name: ppp0   inet_addr: 10.1.1.1 MTU: 1500    Metric: 1
name: ppp1   inet_addr: 10.1.1.2 MTU: 1500    Metric: 1
name: ppp2   inet_addr: 10.1.1.3 MTU: 1500    Metric: 1
name: ppp3   inet_addr: 10.1.1.4 MTU: 1500    Metric: 1
name: ppp4   inet_addr: 10.1.1.5 MTU: 1500    Metric: 1
ifconfigend.

route:
10.1.2.1     10.1.2.1     255.255.255.255 H   0     ppp0
10.1.3.1     10.1.3.1     255.255.255.255 H   0     ppp1
10.0.2.1     10.0.2.1     255.255.255.255     H     0     ppp2
10.0.1.1     10.0.1.1     255.255.255.255     H     0     ppp3
10.0.3.1     10.0.3.1     255.255.255.255     H     0     ppp4
routeend.
```

LSR2.rt:

LSR2 est connecté directement à trois nœuds de réseau via les interfaces ppp0, ppp1 et ppp2 et peut être désigné par inet addr, 10.1.2. *.

```
ifconfig:
name: ppp0  inet_addr: 10.1.2.1     MTU: 1500   Metric: 1
name: ppp1  inet_addr: 10.1.2.2     MTU: 1500   Metric: 1
name: ppp2  inet_addr: 10.1.2.3     MTU: 1500   Metric: 1
ifconfigend.

route:
10.1.1.1    10.1.1.1    255.255.255.255 H 0     ppp0
10.1.4.1    10.1.4.1    255.255.255.255 H 0     ppp1
10.1.6.1    10.1.6.1    255.255.255.255 H 0     ppp2
routeend.
```

De même, d'autres nœuds de réseau sont définis dans les fichiers rt.

- L'adresse d'inet pour :

  LSR3: 10.1.3. *, 3 interfaces ppp0, ppp1, ppp2

  LSR4: 10.1.4. *, 3 interfaces ppp0, ppp1, ppp2

  LSR5: 10.1.5. *, 5 interfaces ppp0, ppp1, ppp2, ppp3, ppp4

  LSR6: 10.1.6. *, 2 interfaces ppp0, ppp1

  LSR7: 10.1.7. *, 2 interfaces ppp0, ppp1

5- Nous définissons un fichier XML pour chaque LSR décrivant les chemins d'accès d'un hôte à l'autre, comme décrit ci-dessous.

LSR1 fec.xml:

```
<?xml version="1.0"?>
<fectable>
      <fecentry>
            <id>1</id>
            <destination>host3</destination>
            <label>1</label>
      </fecentry>
      <fecentry>
            <id>2</id>
            <destination>host4</destination>
            <label>2</label>
      </fecentry>
      <fecentry>
            <id>1</id>
            <destination>host3</destination>
            <label>3</label>
      </fecentry>
      <fecentry>
            <id>2</id>
            <destination>host4</destination>
            <label>4</label>
      </fecentry>
</fectable>
```

Il existe deux FEC définies ici pour les paquets sortant de LSR1. Pour tous les paquets destinés à l'hôte 3, l'ID FEC est 1 et Label insérée est : 1(Orienté vers LSR1), et 3 (Orienté vers LSR3). Pour les paquets destinés à l'hôte 4, le chemin suivi est défini par l'ID FEC, 2 et Label de type : 2 (Orienté vers LSR2), 4 (Orienté vers LSR3).

LSR1 lib.xml:

```
<?xml version="1.0"?>
<libtable>
      <libentry>
            <inLabel>1</inLabel>
            <inInterface>any</inInterface>
            <outInterface>ppp0</outInterface>
            <outLabel>
                  <op code="push" value="101"/>
            </outLabel>
            <color>100</color>
      </libentry>
      <libentry>
            <inLabel>2</inLabel>
            <inInterface>any</inInterface>
            <outInterface>ppp0</outInterface>
            <outLabel>
                  <op code="push" value="301"/>
            </outLabel>
            <color>300</color>
      </libentry>
      <libentry>
            <inLabel>3</inLabel>
            <inInterface>any</inInterface>
            <outInterface>ppp1</outInterface>
            <outLabel>
                  <op code="push" value="302"/>
            </outLabel>
            <color>400</color>
      </libentry>
      <libentry>
            <inLabel>4</inLabel>
            <inInterface>any</inInterface>
            <outInterface>ppp1</outInterface>
            <outLabel>
                  <op code="push" value="303"/>
            </outLabel>
            <color>500</color>
      </libentry>
</libtable>
```

Ce fichier définit les paramètres de la LibTable. Les paramètres sont lus à partir de ce fichier et définis dans l'entrée correspondante de la table de transfert. La première libentry dit que "si un paquet vient avec inlabel comme 1 depuis n'importe quelle interface, envoyez ce paquet sur le chemin représenté par ppp0 après avoir inséré une étiquette 101 dans la pile d'étiquettes" Lorsque ce paquet MPLS encapsulé atteint

LSR2) avec LabelStack en tant que 1 **:** 101, l'entrée correspondante est recherchée dans la table de transmission LSR2.

- Le routeur LSR1 fait le processus **Push (**Label sortante est ajouté**)** parce que c'est le nœud d'entrée **LER (Ingress).**

- Les routeurs LSR2, LSR3, LSR4, LSR6 et LSR7 font le processus **Swap (**Label entrante est remplacé par sortante**)**.

LSR2 lib.xml:

```
<?xml version="1.0"?>
<libtable>
	<libentry>
		<inLabel>101</inLabel>
		<inInterface>ppp0</inInterface>
		<outInterface>ppp1</outInterface>
		<outLabel>
			<op code="swap" value="201"/>
		</outLabel>
		<color>200</color>
	</libentry>
	<libentry>
		<inLabel>301</inLabel>
		<inInterface>ppp0</inInterface>
		<outInterface>ppp1</outInterface>
		<outLabel>
			<op code="swap" value="202"/>
		</outLabel>
		<color>200</color>
	</libentry>
	<libentry>
		<inLabel>101</inLabel>
		<inInterface>ppp0</inInterface>
		<outInterface>ppp2</outInterface>
		<outLabel>
			<op code="swap" value="601"/>
		</outLabel>
		<color>600</color>
	</libentry>
	<libentry>
		<inLabel>301</inLabel>
		<inInterface>ppp0</inInterface>
		<outInterface>ppp2</outInterface>
		<outLabel>
			<op code="swap" value="602"/>
		</outLabel>
		<color>600</color>
	</libentry>
</libtable>
```

LSR3 lib.xml:

```
<?xml version="1.0"?>
<libtable>
   <libentry>
      <inLabel>302</inLabel>
      <inInterface>ppp0</inInterface>
      <outInterface>ppp2</outInterface>
      <outLabel>
          <op code="swap" value="701"/>
      </outLabel>
      <color>200</color>
   </libentry>
   <libentry>
      <inLabel>303</inLabel>
      <inInterface>ppp0</inInterface>
      <outInterface>ppp2</outInterface>
      <outLabel>
         <op code="swap" value="702"/>
      </outLabel>
      <color>400</color>
   </libentry>
   <libentry>
      <inLabel>203</inLabel>
      <inInterface>ppp1</inInterface>
      <outInterface>ppp2</outInterface>
      <outLabel>
         <op code="swap" value="701"/>
      </outLabel>
      <color>200</color>
   </libentry>
   <libentry>
      <inLabel>204</inLabel>
      <inInterface>ppp1</inInterface>
      <outInterface>ppp2</outInterface>
      <outLabel>
         <op code="swap" value="702"/>
      </outLabel>
      <color>400</color>
   </libentry>
</libtable>
```

LSR4 lib.xml

```
<?xml version="1.0"?>
<libtable>
   <libentry>
      <inLabel>201</inLabel>
      <inInterface>ppp0</inInterface>
      <outInterface>ppp2</outInterface>
      <outLabel>
          <op code="swap" value="203"/>
      </outLabel>
      <color>200</color>
   </libentry>
   <libentry>
      <inLabel>202</inLabel>
      <inInterface>ppp0</inInterface>
      <outInterface>ppp2</outInterface>
      <outLabel>
         <op code="swap" value="204"/>
      </outLabel>
      <color>400</color>
   </libentry>
   <libentry>
      <inLabel>201</inLabel>
      <inInterface>ppp1</inInterface>
      <outInterface>ppp2</outInterface>
      <outLabel>
         <op code="swap" value="401"/>
      </outLabel>
      <color>200</color>
   </libentry>
   <libentry>
      <inLabel>202</inLabel>
      <inInterface>ppp1</inInterface>
      <outInterface>ppp2</outInterface>
      <outLabel>
         <op code="swap" value="402"/>
      </outLabel>
      <color>400</color>
   </libentry>
</libtable>
```

LSR6 lib.xml:

```
<?xml version="1.0"?>
<libtable>
      <libentry>
            <inLabel>601</inLabel>
            <inInterface>ppp0</inInterface>
            <outInterface>ppp1</outInterface>
            <outLabel>
                  <op code="swap" value="102"/>
            </outLabel>
            <color>100</color>
      </libentry>
      <libentry>
            <inLabel>602</inLabel>
            <inInterface>ppp0</inInterface>
            <outInterface>ppp1</outInterface>
            <outLabel>
                  <op code="swap" value="302"/>
            </outLabel>
            <color>300</color>
      </libentry>
</libtable>
```

LSR7 lib.xml:

```
<?xml version="1.0"?>
<libtable>
      <libentry>
            <inLabel>701</inLabel>
            <inInterface>ppp0</inInterface>
            <outInterface>ppp1</outInterface>
            <outLabel>
                  <op code="swap" value="102"/>
            </outLabel>
            <color>100</color>
      </libentry>
      <libentry>
            <inLabel>702</inLabel>
            <inInterface>ppp0</inInterface>
            <outInterface>ppp1</outInterface>
            <outLabel>
                  <op code="swap" value="302"/>
            </outLabel>
            <color>300</color>
      </libentry>
</libtable>
```

- Le routeur LSR5 fait le processus **Pop (**enlever Label sortante**)** parce que c'est le nœud d'entrée **LER (Egress).**

LSR5 lib.xml:

```
<?xml version="1.0"?>
<libtable>
      <libentry>
            <inLabel>102</inLabel>
            <inInterface>ppp4</inInterface>
            <outInterface>ppp1</outInterface>
            <outLabel>
                  <op code="pop"/>
            </outLabel>
      </libentry>
      <libentry>
            <inLabel>302</inLabel>
            <inInterface>ppp4</inInterface>
            <outInterface>ppp2</outInterface>
            <outLabel>
                  <op code="pop"/>
            </outLabel>
      </libentry>
</libtable>
```

6- La simulation peut ensuite être exécutée à l'aide du fichier omnetpp.ini.

- ***Problème :***

Nous ne pouvons pas exécuter cet algorithme car nous n'avons pas pu calculer le reste de la bande passante du chemin, faute de ressources et de références parlant de la simulation de réseaux MPLS dans OMNeT ++, cette variable étant nécessaire pour déterminer le seuil de congestion. Ainsi, nous ne pouvons pas dire que la piste est occupée ou non.

Par conséquent, pour démontrer la méthode et le mécanisme dépendent de notre algorithme, nous en supposant manuellement l'apparition de la congestion du réseau en deux endroits (en LSR2 et LSR4) afin d'en extraire le résultat.

Pour créer une congestion manuelle et la gérer de manière classique, nous utilisons les éléments suivants:

LSR1_rsvp.xml:

```
<?xml version="1.0"?>
<sessions>
      <session>
            <endpoint>host3</endpoint>
            <tunnel_id>1</tunnel_id>

            <setup_pri>1</setup_pri>
            <holding_pri>1</holding_pri>

            <paths>
                  <path>
                        <lspid>100</lspid>

                        <bandwidth>400000</bandwidth>
                        <route>
                              <node>LSR1%routerId</node>
                              <node>LSR2%routerId</node>
                              <node>LSR4%routerId</node>
                              <node>LSR5%routerId</node>
                        </route>

                        <permanent>true</permanent>
                        <color>100</color>
                  </path>
            </paths>
      </session>
      <session>
            <endpoint>host4</endpoint>
            <tunnel_id>2</tunnel_id>

            <paths>
                  <path>
                        <lspid>200</lspid>

                        <bandwidth>400000</bandwidth>
                        <route>
                              <node>LSR1%routerId</node>
                              <node>LSR2%routerId</node>
                              <node>LSR4%routerId</node>
                              <node>LSR3%routerId</node>
                              <node>LSR7%routerId</node>
                              <node>LSR5%routerId</node>
                        </route>

                        <permanent>true</permanent>
                        <color>200</color>
                  </path>
            </paths>
      </session>
</sessions>
```

Scenario.xml :

```
<?xml version="1.0"?>
<scenario>
      <at t="2s">
            <add-session module="LSR1.rsvp">
                  <endpoint>host3</endpoint>
                  <tunnel_id>1</tunnel_id>
                  <paths>
                        <path>
                              <lspid>101</lspid>
                              <bandwidth>400000</bandwidth>
                              <route>
                                    <node>LSR1%routerId</node>
                                    <node>LSR2%routerId</node>
                                    <node>LSR6%routerId</node>
                                    <node>LSR5%routerId</node>
                              </route>
                              <color>100</color>
                        </path>
                  </paths>
            </add-session>
      </at>
      <at t="2.2s">
            <bind-fec module="LSR1.classifier">
                  <id>1</id>
                  <endpoint>host3</endpoint>
                  <tunnel_id>1</tunnel_id>
                  <destination>host3</destination>
                  <lspid>101</lspid>
            </bind-fec>
      </at>
      <at t="2.4s">
            <bind-fec module="LSR1.classifier">
                  <id>1</id>
                  <endpoint>host3</endpoint>
                  <tunnel_id>1</tunnel_id>
                  <destination>host3</destination>
                  <lspid>200</lspid>
            </bind-fec>
      </at>
            <at t="2.6s">
            <add-session module="LSR1.rsvp">
                  <endpoint>host3</endpoint>
                  <tunnel_id>1</tunnel_id>
                  <paths>
                        <path>
                              <lspid>200</lspid>
                              <bandwidth>400000</bandwidth>
                              <route>
                                    <node>LSR1%routerId</node>
                                    <node>LSR2%routerId</node>
                                    <node>LSR4%routerId</node>
                                    <node>LSR3%routerId</node>
                                    <node>LSR7%routerId</node>
                                    <node>LSR5%routerId</node>
                              </route>
```

```
                              <color>200</color>
                        </path>
                  </paths>
            </add-session>
      </at>
      <at t="2.8s">
            <bind-fec module="LSR1.classifier">
                  <id>1</id>
                  <endpoint>host3</endpoint>
                  <tunnel_id>1</tunnel_id>
                  <destination>host3</destination>
                  <lspid>200</lspid>
            </bind-fec>
      </at>
</scenario>
```

Le type de module Scenario Manager sert à configurer et à contrôler des expériences de simulation. Scenario Manager exécute un script spécifié en XML. Il comporte quelques commandes intégrées, tandis que d'autres commandes sont envoyées (en C ++) pour être exécutées par d'autres modules simples.

Après, Ensuite, nous appliquons Run :

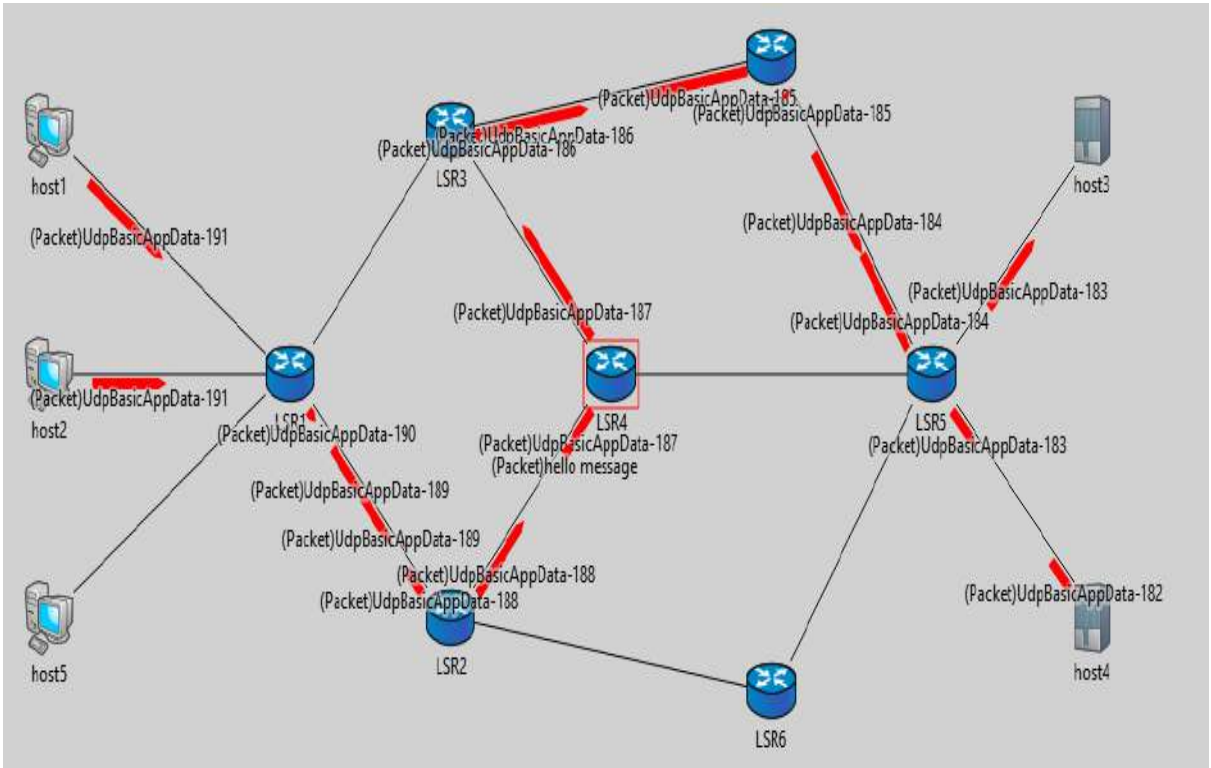

Figure 4.12 : Chemin de transmission de données avant la congestion.

La figure 4.12 indique que les données sont transmises des clients (host1 et host 2) aux serveurs (host3 et host4) le long du chemin : LSR1 – LSR2 – LSR4 – LSR3 – LSR7 – LSR5.

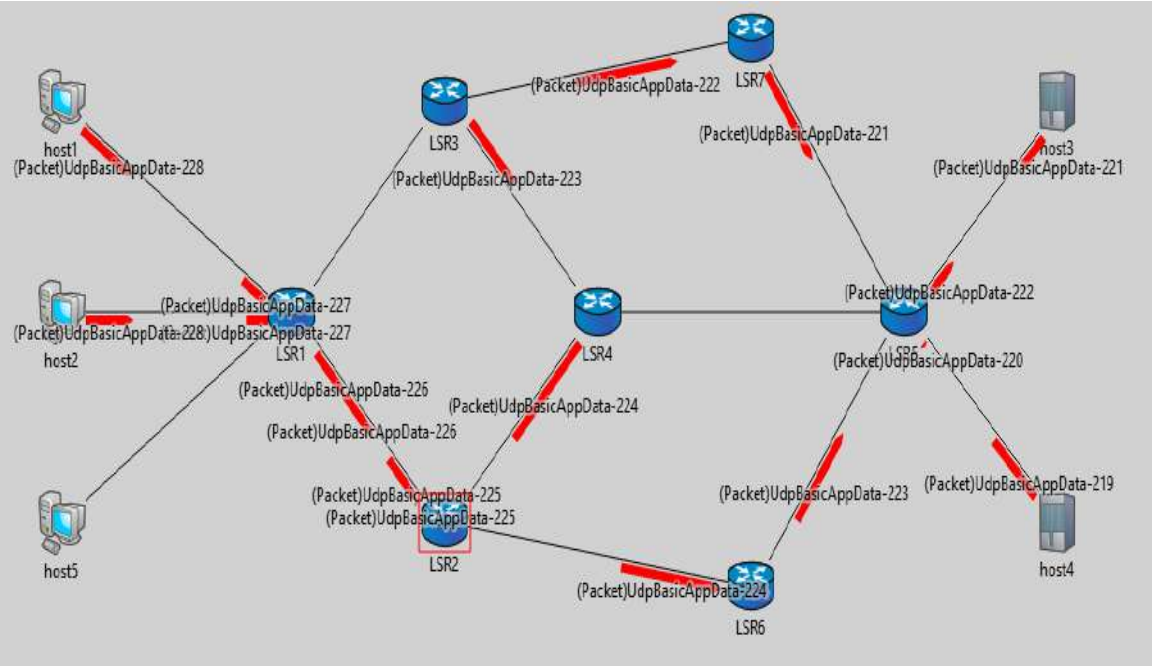

Figure 4.13 : Chemin de transmission de données après la congestion dans LSR2

Donc, Notez que sur la figure 4.13, La congestion se produit dans le noeud LSR2, il existe deux chemins de transfert de données après l'application de l'algorithme de déviation, le chemin d'origine (LSR1 – LSR2 – LSR4 – LSR3 – LSR7 – LSR5) et nouvel chemin (le chemin de déviation (LSR1 – LSR2 – LSR6 – LSR5) étant utilisés pour éviter et résoudre le problème d'encombrement.

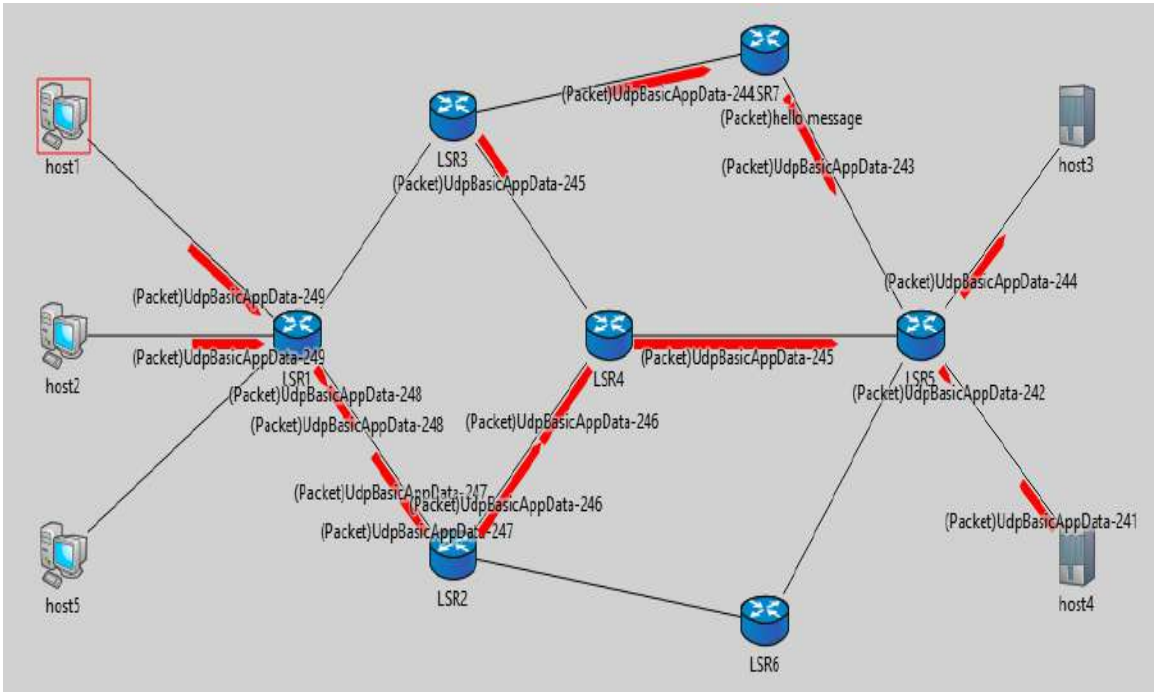

Figure 4.14 : Chemin de transmission de données après la congestion dans LSR4

Donc, Notez que sur la figure 4.14, La congestion se produit dans le noeud LSR4, il existe deux chemins de transfert de données après l'application de l'algorithme de déviation, le chemin d'origine (LSR1 – LSR2 – LSR4 – LSR3 – LSR7 – LSR5) et nouvel chemin (le chemin de déviation (LSR1 – LSR2 – LSR4 – LSR5) étant utilisés pour éviter et résoudre le problème d'encombrement.

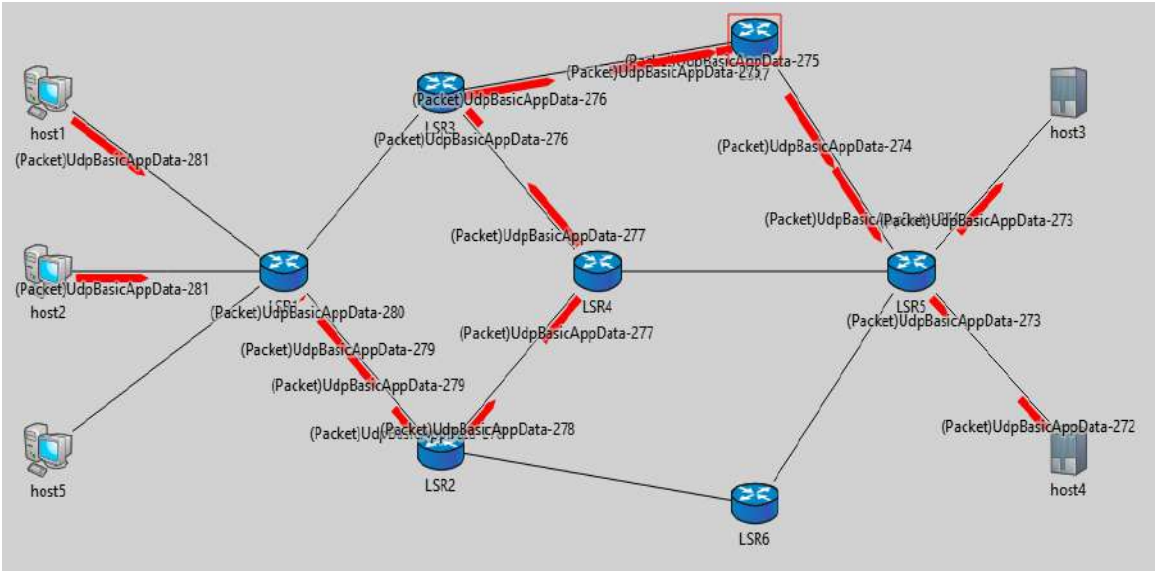

Figure 4.15 : Chemin de transmission de données Après la fin de la congestion.

Donc, Notez que sur la figure 4.15, Une fois l'encombrement terminé, renvoyez les données par le chemin d'origine (LSR1 – LSR2 – LSR4 – LSR3 – LSR7 – LSR5).

7- Pour l'analyse :

Nous générons un fichier d'analyse sous ce projet. La simulation produit des fichiers vectoriels (.vec) et scalaires (.sca) qui sont chargés dans le fichier d'analyse. Chacun des vecteurs et des scalaires peut être ouvert pour afficher les graphiques correspondants.

4.6 Résultats de la simulation et analyses :

- **CAS 1: Pour bandwidth (la bande passante)**

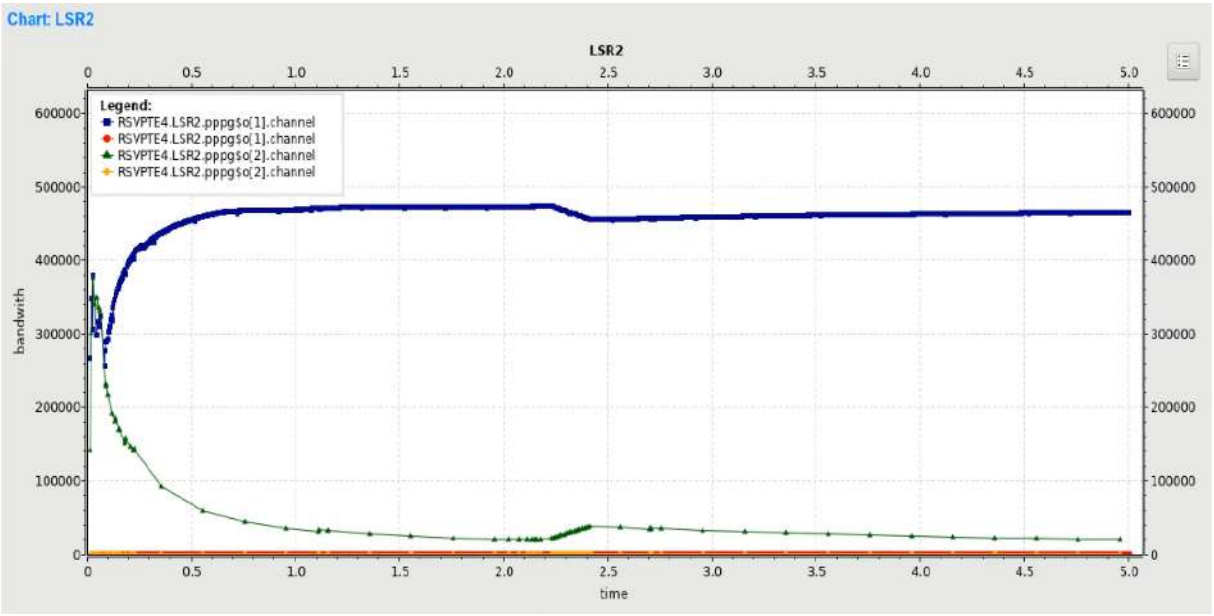

Figure 4.16 : Graph de la Changement de bande passante au niveau du LSR2.

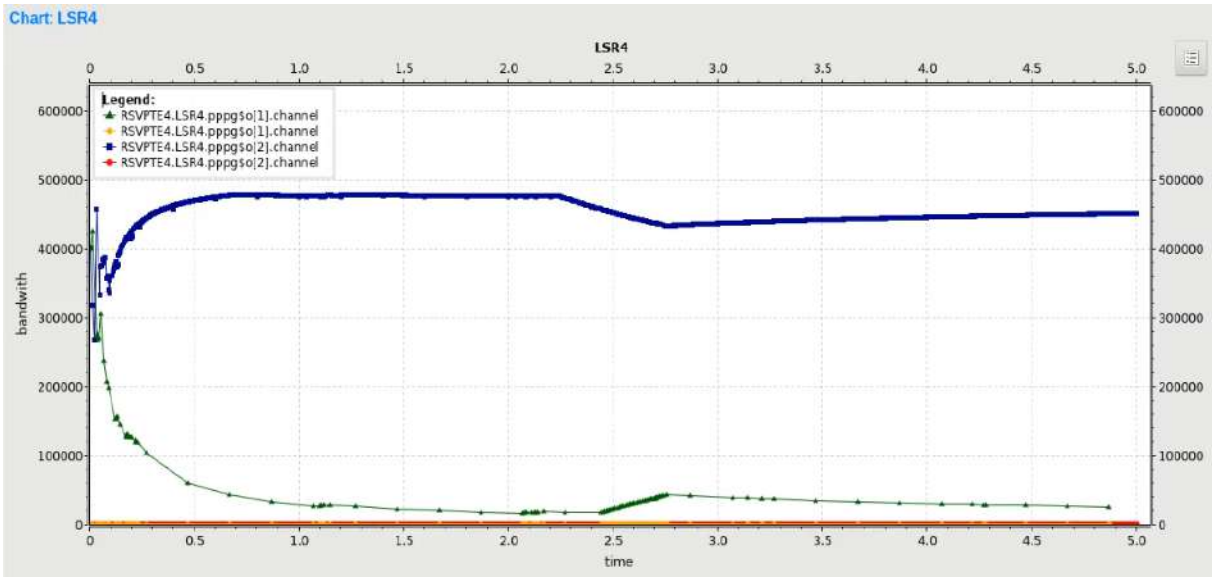

Figure 4.17: graph de la Changement de bande passante au niveau du LSR4.

- **<u>Cas 2</u> : Pour delay (retard)**

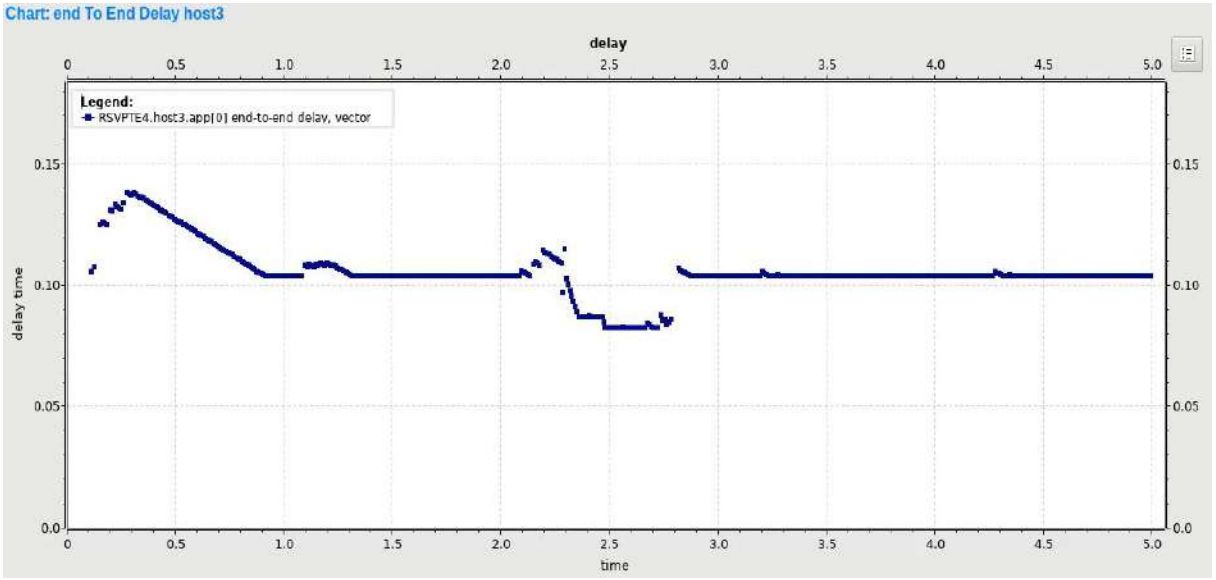

Figure 4.18 : graph de la Changement delay au niveau du Host3.

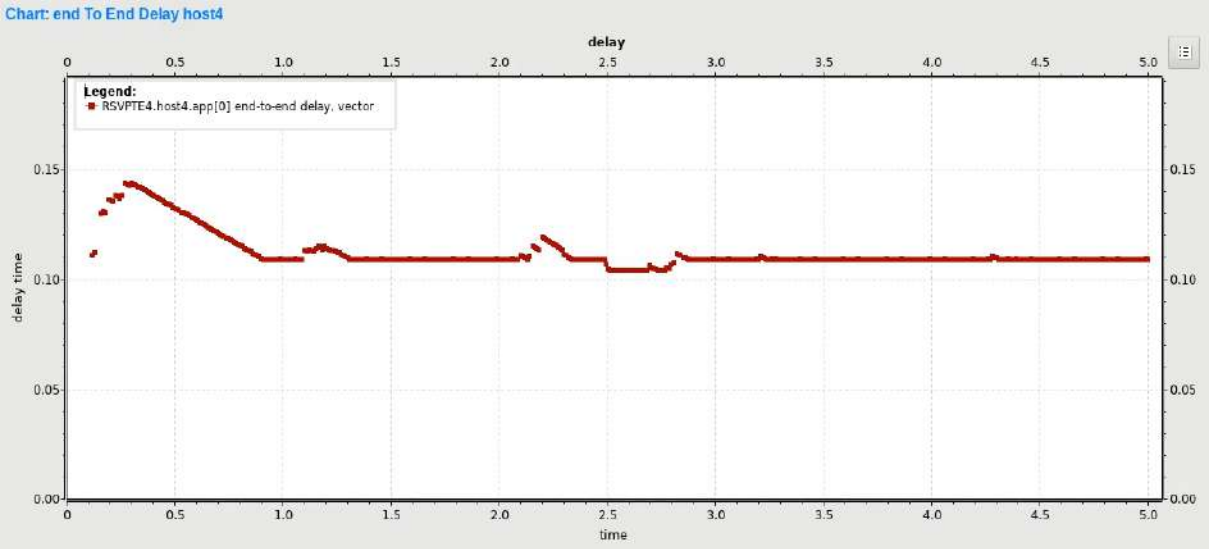

Figure 4.19 : graph de la Changement delay au niveau du Host4.

- **Observation et inférence:**

  - Dans le cas 1 pour la bande passante, on remarque sur le graph du LSR2 à partir du moment 2.2s **(congestion)** une diminution de la valeur de la bande passante au niveau de l'interface ppp1 connectée au LSR4 (qui est le chemin encombré), tandis que la valeur de la bande passante augmente au niveau de l'interface ppp2 connecté avec le LSR6 (qui est un chemin déviant). La même chose pour LSR4.

  - Dans le cas 2 pour delay, on remarque sur le graph du LSR2 et LSR4 à partir du moment 2.2s **(congestion)** une diminution de la valeur de delay au niveau de host 3 et host 4, Après la fin de la congestion à partir du moment 2.8s, nous voyons le retour de la valeur de delay à ce qu'elle était.

4.7 Conclusion :

Ce à quoi nous avons fait référence dans la description de l'algorithme du chapitre précédent, que nous avons appelé LBMPLS, devait être appliqué, implémenté et simulé afin de les identifier et de déterminer son efficacité en terme de réduction des encombrements.

Dans ce chapitre, nous avons appliqué et écrit en C ++ pour émuler OMNeT ++, et nous les avons divisés en plusieurs classes.

Au cours de la simulation, nous avons utilisé INET et apporté plusieurs modifications aux fichiers MPLS afin d'adapter le fonctionnement de l'algorithme LBMPLS.

Mais faute de ressources et de références pour OMNeT ++ et MPLS, nous n'avons pas pu implémenter cet algorithme et l'émuler sur OMNeT ++ car nous ne savions pas comment calculer le reste de la bande passante du chemin dans le réseau, elle est considérée comme la variable nécessaire pour connaître le seuil de congestion, sans qu'elle puisse savoir si le chemin est occupé ou non, nous ne pouvons donc pas implémenter LBMPLS.

Cependant, nous avons créé la congestion manuelle, en supposant qu'il soit produit à deux endroits du réseau et traité de manière statique. Le but de cette hypothèse était de démontrer le mécanisme de l'algorithme LBMPLS et son efficacité pour réduire la congestion et analyser ses résultats.

L'analyse des résultats de l'algorithme montre qu'il est efficace pour réduire efficacement la congestion, et améliorer les performances du réseau en réduisant le temps de transfert des données et les délais.

Chapitre

5 Conclusion Générale

Contenu

5. Conclusion Générale :

Pour terminer ce mémoire, nous faisons ici le bilan de l'étude réalisée. Nous résumons les apports, les limitations de ce travail et nous dégageons certaines perspectives de recherche pour étendre notre algorithme.

5.1 Bilan de l'étude réalisé :

La croissance explosive d'Internet a grandement affecté la gestion et le contrôle du réseau. L'ajout de ressources supplémentaires au réseau peut temporairement réduire les encombrements, mais ce n'est pas une solution rentable pour résoudre les conflits de ressources à long terme. De plus, la tendance à fournir des différentes classes de services ajoute une couche supplémentaire de complexité à la gestion et aux opérations du réseau. Les fournisseurs ont besoin de mécanismes pour coordonner, contrôler et utiliser efficacement les ressources existantes afin de répondre à la demande des clients.

Aujourd'hui, MPLS est devenu la technologie de transport dominante dans les réseaux centraux à base de paquets. MPLS a mis au point un certain nombre de mécanismes pour améliorer la disponibilité, la fiabilité et les améliorations du réseau. La réduction de la congestion est l'un des principaux objectifs de performance axés sur le trafic et les ressources.

MPLS utilise également l'ingénierie du trafic pour contrôler les flux de trafic réseau, optimiser l'utilisation des ressources et les performances du réseau, et réduire les encombrements.

L'ingénierie du trafic (TE) est devenue une exigence de base pour les FAI afin d'optimiser l'utilisation des ressources du réseau existantes et de maintenir la qualité de service avec des ressources du réseau réduites. L'un de ses principaux objectifs est d'équilibrer les charges sur le réseau, l'équilibrage de la charge étant un aspect important de l'ingénierie du trafic. Il peut utiliser certains mécanismes pour tracer une partie du trafic sur des routes très utilisées, mais également sur des voies qui ne le sont pas assez pour éviter les embouteillages sur un chemin plus court et pour améliorer la productivité totale du réseau et l'utilisation des ressources du réseau MPLS. MPLS permet un routage très efficace des chemins de commutation d'étiquettes, ce qui peut être utilisé pour optimiser l'utilisation des ressources du réseau et améliorer les caractéristiques de performance orientées.

L'objectif principal de notre recherche était de proposer un nouvel algorithme idéal pour la gestion de la congestion dans les réseaux MPLS. , Sur la base des méthodes étudiées dans la

littérature et en prenant en compte les problèmes de recherche et les conclusions apparues dans nos enquêtes.

Nous avons donc proposé une nouvelle approche de réduction de l'encombrement dans MPLS utilisant le mécanisme d'équilibrage de charge, appelé LBMPLS, qui vise à déformer efficacement les LSP des liaisons les plus utilisées du réseau en liaisons insuffisamment utilisées. Nous utilisons le concept de seuil pour déterminer la congestion, considérons la priorité de flux pour accélérer le temps de transmission élevé du flux prioritaire, réduisons le délai de bout en bout et la capacité libre pour sélectionner le chemin de déviation adéquat.

5.2 Limitations:

Au cours de ce travail, nous avons dû faire face à plusieurs difficultés inhérentes au contexte dans lequel nous évoluons, les limitations dont nous faisons état ici en sont la conséquence directe. La nature de ce mémoire nous a conduit à explorer un nouveau domaine au cours de l'expérimentation.

Afin de vérifier la validité de cette proposition et de connaître l'efficacité de cet algorithme et ce qui le distingue des autres, vous devez appliquer, implémenter, simuler et comparer d'autres moyens, mais nous n'avons pas pu émuler OMNeT ++ pour plusieurs raisons:

- La difficile d'utiliser le programme OMNET.
- Le manque de ressources et de références appartenant à OMNET, notamment en ce qui concerne la simulation des réseaux MPLS.
- Le temps: notre recherche a demandé beaucoup d'efforts pour rassembler les informations disponibles dans ce domaine.
- La raison principale est que nous ne pouvons pas calculer le reste de la bande passante du réseau MPLS, une variable très importante dans le travail de l'algorithme, qui est responsable de la détection de l'encombrement à l'aide du seuil.

5.3 Solution :

Les raisons que nous avons rencontrées en simulation ne nous ont pas empêché de trouver une solution qui nous permettant de montrer le fonctionnement de notre algorithme, même s'il ne nous donne pas le résultat attendu.

La solution que nous avons expliquée était manuelle et représentait l'hypothèse d'encombrement en deux points du réseau. Notre encombrement a été résolu de la même manière que notre algorithme, mais de manière fixe, car le problème n'est qu'une hypothèse.

L'analyse des résultats de l'algorithme montre qu'il est efficace pour réduire efficacement la congestion, et améliorer les performances du réseau en réduisant le temps de transfert des données et les délais.

5.4 Perspectives :

Comme perspectives pour la poursuite de nos travaux, nous espérons dans le proche avenir que cette recherche sera l'une des sujets de la thèse et qu'elle se développera autant que possible avec suffisamment de temps.

Nous travaillerons pour trouver des solutions à toutes les difficultés rencontrées dans notre recherche, à la fois dans la maîtrise de l'utilisation d'OMNET et de tout ce qui l'entoure, et dans le cadre de la mise en œuvre de notre algorithme et de l'analyse de ses résultats,

Et connaître l'efficacité, distinguer les avantages et les inconvénients, les comparer à d'autres algorithmes et essayer de les améliorer et de les intégrer à de nouvelles méthodes et à des méthodes adaptées au développement technologique.

Références Bibliographiques

1.1 D.Awduche, J.Malcolm, J.Agogbua, M.O'Dell et J.McManus, "Requirements for Traffic Engineering Over MPLS", RFC2702 IETF, September 1999.

1.2 https://itel.com/what-is-mpls-and-how-does-it-work/, Accédé février /03 /2019.

1.3 https://www.techopedia.com/definition/527/multiprotocol-label-switching-mpls; Accédé février /03 /2019.

1.4 https://searchnetworking.techtarget.com/definition/Multiprotocol-Label-Switching-MPLS; Accédé février /03 /2019.

1.5 Felicia Marie Holness, "Congestion Control Mechanisms with in MPLS Networks", Septembre 2000.

1.6 Laurent Charbonnier, "Evaluation de la sécurité des réseaux privés virtuels sur MPLS", Montréal, Accédé Décembre /10/2007.

1.7 Sanu Mathews, "Multi-Protocol Label Switching (MPLS)", https://thinkpalm.com/blogs/multi-protocol-label- switching/, Accédé fivrier /15/2019.

1.8 Petr Grygarek, "Multi-Protocol Label Switching (MPLS)", 2005.

1.9 Richard A Steenbergen <ras@nlayer.net> nLayer Communications, Inc. "MPLS for Dummies ".

1.10 Ghefir Mohamed El Amine, Mémoire de Magister en Systèmes et Réseaux de Télécommunication, « planification, ingénierie des réseaux de nouvelle génération – NGN », Décembre 2013.

1.11 Steve Smith, Systems Engineer, « Introduction to MPLS », 2003.

1.12 https://www.cisco.com/c/en/us/td/docs/ios/12_0s/feature/guide/TE_1208S.pdf

1.13 https://www.mplsinfo.org/traffic-engineering.html, Accédé février /08 /2019.

2.1 Prabhjot Kaur, Satish Devane, "LOAD BALANCING IN MPLS"**,** International Journal of Computer Engineering & Science, ov. 2013.

2.2 **[Keping Long *et al*, 2001**], Keping Long , Zhongshan Zhang and Shiduan Cheng, "Load Balancing Algorithms in MPLS Traffic Engineering" , IEEE, 2001

2.3 **[Gustavo B *et al*, 2004],** Gustavo B. Figueiredo and Nelson L. S. da Fonseca, "A Minimum Interference Routing Algorithm" , IEEE Communications Society, 2004

2.4 **[Elio Salvadori *et al*, 2003],** Elio Salvadori, Roberto Battiti, " A Load Balancing Scheme for Congestion Control in MPLS Networks " , IEEE International Symposium on Computers and Communication (ISCC'03), 2003.

2.5 [Tijani Mohammed *et al*, 2007], Chengcheng, Li Peng Li and Tijjani Mohammed, "An Optimal MPLS-TE Solution to Route Selection and Redistribution on Congested Networks", IEEE International Conference on Networking, Architecture, and Storage (NAS 2007), 2007.

2.6 [Anwar Elwalid *et al*, 2001], Anwar Elwalid, Cheng Jin, Steven Low and Indra Widjaja, "MATE: MPLS Adaptive Traffic Engineering", IEEE INFOCOM 2001.

2.7 [R. Boutaba *et al*, 2002], R. Boutaba, W. Szeto and Y. Iraqi, « DORA: Efficient Routing for MPLS Traffic Engineering" , Journal of Network and Systems Management, Vol. 10, No. 3, September 2002.

2.8 [Sherif Ibrahim Mohamed *et al*, 2006], Sherif Ibrahim Mohamed and Khaled M. F. Elsayed, senior member IEEE, "Distributed Explicit Partial Rerouting (DEPR) Scheme for Load Balancing in MPLS Networks" , IEEE Symposium on Computers and Communications (ISCC'06),2006.

2.9 [A.B. Bagula *et al*, 2004], A.B. Bagula, M. Botha, and A.E Krzesinski, "Online Traffic Engineering: The Least Interference Optimization Algorithm" , IEEE Communications Society, 2004.

2.10 [Bing-feng Cui *et al*, 2004], Bing-feng Cui, Zhen Yang, Wei Ding, "A Load Balancing Algorithm Supporting QoS for Traffic Engineering in MPLS Networks" , IEEE International Conference on Computer and Information Technology (CIT'04), 2004.

2.11 [J. Tang *et al*, 2005], J. Tang, C.K. Siew and G. Feng, "Parallel LSPs for constraint-based routing and load balancing in MPLS networks" , IEE Proc.-Commun., Vol. 152, No. 1, February 2005.

2.12 [K. Abboud *et al*, 2008], K. Abboud, A. Toguyeni and A. Rahmani , " PERFORMANCE AND COMPLEXITY EVALUATION OF MULTI-PATH ROUTING ALGORITHMS FOR MPLS-TE" , International Conference on Software and Data Technologies - ICSOFT 2008.

2.13 [Fenglin Li *et al*, 2012], Fenglin Li and Jianxun Chen, " MPLS Traffic Engineering Load Balance Algorithm Using Deviation Path" , IEEE International Conference on Computer Science and Service System, 2012.

4.1 [M. E. Fekair, 2017], Fekair, Mohammed El Amine, " Etude et modélisation des techniques de qualité de service (QoS) dans les réseaux véhiculaires", thèse de doctorat, université KASDI-MERBAH OUARGLA, 2017.

4.2 INET Framework User's Guide, OpenSim Ltd, Jan 28, 2019.