
DIPLÔME EN TRANSMISSION DE DONNÉES RAS & PROTOCOLE DE ROUTAGE MPLS

Session 2001

Candidat : Philippe Logean

phlogean1@geneva-link.ch

Filière télécommunication

Professeur : Eric Jenny

Résumé *Ce document traite deux sujets. Le premier explique la manière de configurer un routeur Cisco comme serveur d'accès distant. Dans cette partie, le fonctionnement du protocole PPP est développé. La deuxième sujet présente les protocoles MPLS et LDP. Des applications pratiques sont réalisées, sur des machines Linux et des routeurs Cisco, afin d'illustrer le fonctionnement de ces protocoles. Quelques applications du protocole MPLS pour le trafic engineering sont aussi présentées.*

Mots clef *Routeur / RAS / PPP / MPLS / LDP / Trafic Engineering / VPN*

Table des matières

1. Présentation du travail de diplôme	4
2. Routeur CISCO 2621	5
2.1. Introduction.....	5
2.2. Philosophie du routeur	5
2.2.1. Configuration du routeur de l'extérieur	5
2.2.2. Composant de configuration interne	6
2.2.3. Accès à l'interface utilisateur	7
2.2.4. Quelques commandes utiles.....	8
3. Configuration du routeur comme serveur d'accès distant	9
3.1. Objectifs	9
3.2. Configuration « routeur à routeur ».....	9
3.2.1. Topologie	9
3.2.2. Configuration des PCs (sur Windows 2000).....	10
3.2.3. Configuration et test des routeurs (HDLC).....	11
3.2.4. Point to Point Protocol (PPP).....	15
3.2.5. Configuration des routeur pour PPP.....	21
3.3. Configuration « TA à routeur » avec adresse statique	23
3.3.1. Topologie	23
3.3.2. Configuration des PCs.....	23
3.3.3. Configuration du routeur	27
3.3.4. Remarques sur le « dialer map »	29
3.3.5. Remarques sur « idle-timeout »	29
3.4. Configuration « unnumbered »	30
3.5. Configuration dynamique de l'adresse IP du client	31
3.5.1. Principe.....	31
3.5.2. Configuration du routeur	31
3.5.3. Configuration du PC client.....	32
3.6. Configuration pour accès distant au réseau du laboratoire	34
3.6.1. Topologie	34
3.6.2. Configuration du routeur pour un client	34
3.6.3. Configuration du routeur pour l'accès de plusieurs clients.....	36
3.6.4. Capture d'une phase d'établissement avec le débogueur du routeur Cisco	37
3.7. Conclusion.....	38
4. Introduction au protocole <i>MPLS</i>	39
4.1. <i>MPLS</i> , c'est quoi ?	39
4.2. Concept.....	39
4.3. Élément du réseau <i>MPLS</i>	40
4.4. Encapsulation des paquets.....	41
4.5. <i>Forwarding</i> des paquets.....	42
4.6. Routage des paquets.....	43
4.7. Distribution des labels avec le protocole <i>LDP</i>	43
4.7.1. Introduction.....	43
4.7.2. Types de messages <i>LDP</i>	43
4.7.3. Modes de distribution des labels	45
4.7.4. Exemple d'échange <i>LDP</i>	46
4.7.5. Format des paquets.....	47
5. Etude du mécanisme de <i>forwarding</i> <i>MPLS</i> sur des machines Linux	50
5.1. Introduction.....	50

5.2. Configuration des interfaces des PCs.....	51
5.3. Encapsulation des paquets avec les labels <i>MPLS</i>	52
5.3.1. Objectif.....	52
5.3.2. Configuration des LER.....	52
5.3.3. Configuration des LSR.....	55
5.3.4. Tests et analyse des paquets.....	56
5.4. Création d'un <i>stack</i> de labels	58
5.4.1. Objectif.....	58
5.4.2. Configuration des LSR.....	59
5.4.3. Tests et analyse des paquets.....	60
5.5. Réseau IP et <i>MPLS</i>	62
5.5.1. Objectif.....	62
5.5.2. Configuration des LER.....	62
5.5.3. Configuration des autres machines	64
5.5.4. Tests et analyse des paquets.....	64
5.6. Conclusion.....	65
6. Etude du mécanisme de distribution des labels avec <i>LDP</i> sur Linux	66
6.1. Introduction.....	66
6.2. Transfert <i>LDP</i> entre deux <i>LER</i>	66
6.2.1. Objectifs	66
6.2.2. Configuration des machines.....	66
6.2.3. Tests et analyse des paquets.....	69
6.3. Réseau <i>MPLS</i> utilisant <i>LDP</i>	73
6.3.1. Objectifs	73
6.3.2. Configuration des machines.....	73
6.3.3. Teste et analyse des paquets	75
6.4. Conclusion.....	76
7. Mise en place d'un réseau utilisant MPLS avec des routeur Cisco	77
7.1. Introduction.....	77
7.2. Mise en œuvre d' <i>OSPF</i>	77
7.3. Activation d' <i>MPLS</i> sur les routeurs	79
7.4. Tests du réseau.....	80
8. Traffic engineering sur le réseau MPLS	82
8.1. Introduction.....	82
8.2. Création d'un <i>path</i> statique	82
8.2.1. Objectif.....	82
8.2.2. Autoriser le traffic engineering sur les routeurs.....	82
8.2.3. Création des paths	83
8.2.4. Test et analyse	85
8.3. Création d'un <i>backup path</i>	89
8.3.1. Objectif.....	89
8.3.2. Création des path.....	89
8.3.3. Création des backup-path.....	90
8.3.4. Test et analyse	91
8.4. Création d'une route dynamique avec contrainte.....	93
8.4.1. Objectif.....	93
8.4.2. Préparation des routeurs pour RSVP	93
8.4.3. Création dynamique de tunnels avec contrainte.....	94
8.4.4. Test et analyse	95
8.5. Conclusion.....	96

9. Utilité de <i>MPLS</i> pour les <i>VPN</i>	97
9.1. Introduction.....	97
9.2. Configuration des routeurs.....	98
9.3. Fonctionnement du <i>VPN</i> grâce à <i>MPLS</i>	98
9.4. Conclusion.....	99
10. Remarques et conclusion finale	100
11. Remerciements.....	101
12. Planning.....	102
13. Annexes	103
14. Bibliographie	104
14.1. Livres.....	104
14.2. RFC	104
14.3. Web	105

1. Présentation du travail de diplôme

Dans ce travail de diplôme, deux sujets sont traités. Le premier, est la réalisation d'un *RAS (Remote Access)* utilisant un routeur Cisco. Le deuxième, est l'étude du protocole de routage *MPLS*.

Le but de la partie *RAS* est de réaliser une topologie de réseau permettant à un client (PC) d'accéder, via *ISDN*, à un réseau Internet ou intranet. Pour cela, un routeur Cisco 2600 est utilisé comme serveur d'accès distant. L'étude de cette topologie est faite en plusieurs étapes, permettant de traiter les différents problèmes un à un. Les objectifs de cette première partie du travail diplôme sont de se familiariser avec les routeurs Cisco et de réaliser une configuration pouvant être utilisée pour les travaux pratiques du laboratoire. Pour cela, chaque configuration testée est décrite en détail. Une description du protocole *PPP* est aussi développée dans ce document. En effet, c'est ce protocole que nous utilisons pour les phases d'établissement et de transmission du *RAS*.

Le but de la partie *MPLS* est de réaliser une première étude de ce protocole, pour permettre au laboratoire d'acquérir des compétences dans ce domaine. En effet, le protocole *MPLS* est susceptible d'être de plus en plus utilisé par les *providers*, car il offre des solutions aux problèmes de transport des paquets à travers le *core network*. Certains de ces problèmes sont étudiés dans ce travail. Les mécanismes utilisés, par *MPLS*, pour le *forwarding* des paquets, sont développés dans ce document. Le protocole de distribution de label *LDP* est aussi décrit. Une étude pratique de ces protocoles a été réalisée, au laboratoire, sur des machines Linux. Nous avons eu la chance de pouvoir mettre en oeuvre des scénarios utilisant *MPLS* pour effectuer du *traffic engineering*, sur des routeurs Cisco 7200, au laboratoire IPSS de Swisscom à Bümplitz.

2. Routeur CISCO 2621

2.1. Introduction

Dans le cadre de ce travail de diplôme, les éléments principaux utilisés, pour la réalisation pratique de l'étude, sont les routeurs Cisco. En effet, la configuration d'un serveur d'accès distant ou la mise en place d'un réseau utilisant le protocole MPLS nécessite de connaître un minimum sur la façon dont ils fonctionnent. Ce chapitre n'a pour but que de présenter les principes de base de ces routeurs. Pour plus de détails, le lecteur peut se référer aux chapitres 5 et 6 du livre «*Introduction to Cisco router configuration*».

2.2. Philosophie du routeur

2.2.1. Configuration du routeur de l'extérieur

Les routeurs Cisco peuvent être configurés de plusieurs manières (FIGURE 2.1).

- *Console Port* : Permet de configurer le routeur depuis un terminal directement connecté sur celui-ci.
- *Auxiliary Port* : Le routeur peut aussi être configuré en utilisant le port auxiliaire.
- *Virtual Terminals* : Une fois le routeur connecté au réseau, la configuration peut s'effectuer depuis un terminal virtuel, typiquement par Telnet.
- *TFTP Server* : La configuration du routeur peut être téléchargée, par le réseau, depuis un serveur TFTP. L'opération inverse peut aussi être effectuée afin de sauvegarder la configuration du routeur sur un serveur TFTP.
- *Network Management Station* : Permet de configurer le routeur depuis un système distant où est installé un programme de *management* réseau comme CiscoWorks ou HP OpenView (SNMP).

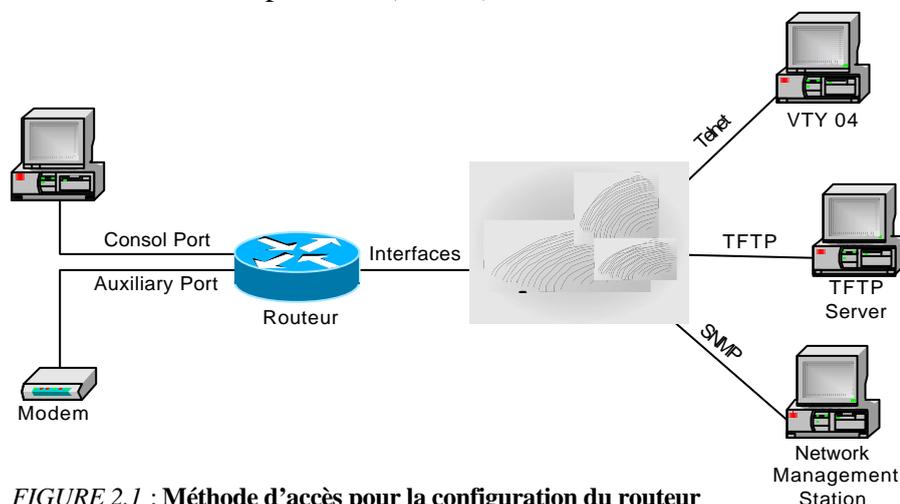


FIGURE 2.1 : Méthode d'accès pour la configuration du routeur

Pour ce travail, la configuration s'effectue en utilisant le « *Consol Port* » du routeur et le logiciel HyperTerminal avec, comme paramètres, 8 bits de données à 9600 bits/s, sans parité et 1 bit de stop. Le routeur est relié à un ordinateur grâce à un câble plat RJ-45 et à un adaptateur RJ45-DB9 (fourni avec le routeur) connecté sur le port série du PC. Afin de sauvegarder les différentes configurations du routeur, un serveur TFTP est aussi utilisé. Cisco fournit un serveur TFTP qui peut être chargé depuis le site <http://www.cisco.com/cgi-bin/tablebuild.pl/tftp>.

2.2.2. Composant de configuration interne

L'architecture interne du routeur comporte divers éléments qui jouent un rôle important dans le *startup process*. La figure ci-dessous les illustre :

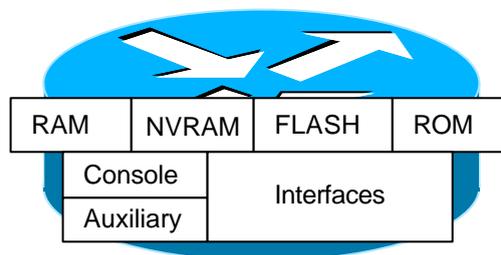


FIGURE 2.2 : Composant de configuration interne

- **RAM/DRAM** : La RAM/DRAM est la mémoire principale du routeur. Appelée aussi *working storage*, elle contient les informations de la configuration en cours.
- **NVRAM** : (*nonvolatile RAM*) Elle contient une copie de sauvegarde de la configuration du routeur. Si le routeur est éteint, la copie de sauvegarde permet au routeur de fonctionner au prochain démarrage, sans qu'il soit nécessaire de le reconfigurer.
- **Flash Memory** : Cette mémoire contient une copie du programme *Cisco Internetwork Operating System* (Cisco IOS). La structure de cette mémoire permet de garder plusieurs versions du programme Cisco IOS. Lorsque le routeur est éteint ou redémarré, le contenu de cette mémoire est conservé.
- **ROM** : Cette dernière contient un programme d'initialisation du démarrage et un petit système de monitoring qui peut être utilisé pour réparer en cas de problème durant la phase d'initialisation.
- **Interfaces** : Les interfaces sont les connexions réseau du routeur où entrent et sortent les paquets. Le routeur Cisco 2621 possède deux interfaces Fast Ethernet 10/100BaseT.
- **Auxiliary Ports** : Le programme Cisco IOS autorise l'utilisation du port auxiliaire pour le routage asynchrone comme interface de réseau.

2.2.3. Accès à l'interface utilisateur

La console du routeur peut être obtenue par une émulation de terminal sur une station de travail, comme HyperTerminal ou par Telnet sur une station distante. Les deux méthodes donnent accès à l'interface utilisateur du programme Cisco IOS.

L'interface utilisateur du programme Cisco IOS offre plusieurs modes de commandes comme le montre la figure ci-dessous.

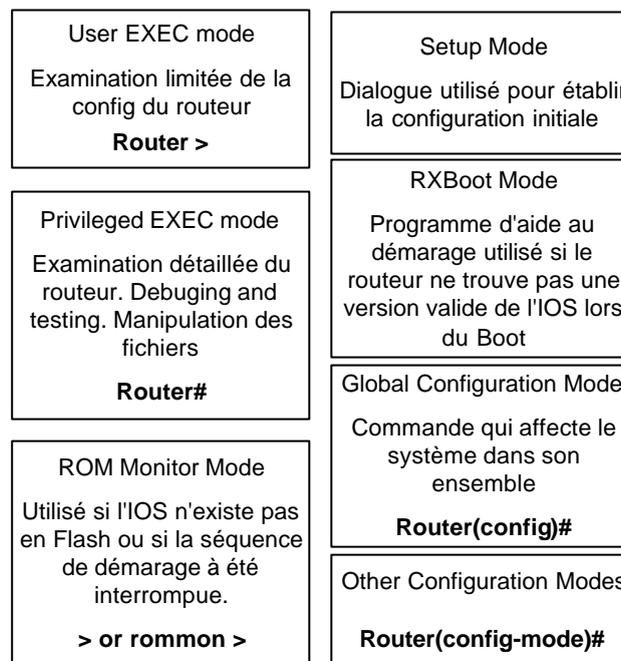


FIGURE 2.3 : Modes de commandes

Dans la console, les commandes suivantes permettent de passer d'un mode à un autre.

Passer en mode privilégié :

- Router> **enable**

Configuration globale du routeur :

- Router# **configure terminal**

Configurer une interface :

- Router(config)# **interface fastethernet0/0**

La commande «exit» permet de revenir à l'état précédent. Les touches Ctrl-Z permettent de sortir du mode configuration.

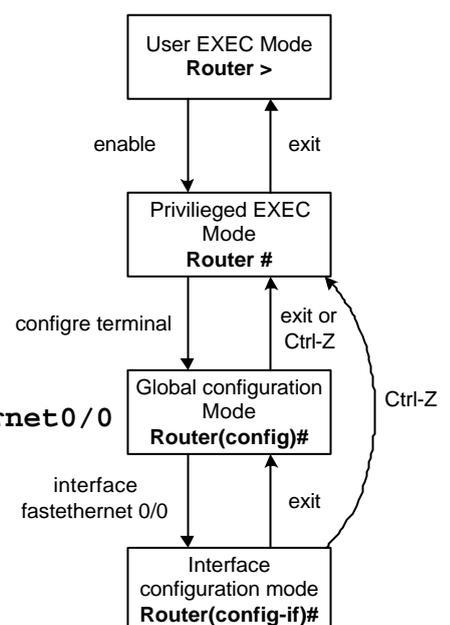


FIGURE 2.4 : Changement de mode

2.2.4. Quelques commandes utiles

Voici quelques commandes souvent utilisées :

➤ Router# **show version**

Affiche la configuration du hardware, la version du software, le nom et la source du fichier de configuration et l'image du *boot*.

➤ Router# **show running-config**

Affiche la configuration actuelle du routeur qui est contenue dans la RAM.

➤ Router# **show startup-config**

Affiche la configuration utilisée lors du démarrage du routeur qui est stockée dans la NVRAM.

Pour connaître les mots-clefs et les arguments disponibles associés à une commande, il suffit de mettre un « ? » après celle-ci. Par exemple :

➤ Router# show ?

➤ Router# **copy running-config startup-config**

Sauvegarde la configuration courante du routeur dans la NVRAM. Cette configuration sera utilisée lors du prochain démarrage du routeur. L'opération inverse est aussi possible pour recharger la configuration de démarrage en RAM.

➤ Router# **copy startup-config tftp:**

Permet d'envoyer la configuration contenue dans la NVRAM sur un serveur TFTP. Il est aussi possible d'envoyer la configuration contenue dans la RAM en copiant la « running-config ».

➤ Router# **copy tftp: startup-config**

Effectue l'opération inverse que précédemment, c'est-à-dire charger la NVRAM depuis in serveur TFTP. Cette opération est aussi possible pour la RAM en indiquant « running-config ».

3. Configuration du routeur comme serveur d'accès distant

3.1. Objectifs

Dans le cadre du *Remote Access* (RAS), il faut mettre en place une topologie de réseau permettant à un client (PC) d'accéder, via *ISDN*, au réseau Internet ou à un réseau intranet (FIGURE 3.1). Cela permet, par exemple, à un employé d'accéder au réseau de son entreprise depuis son domicile de façon totalement transparente. Internet sera simulé par un réseau Ethernet et un *PABX* jouera le rôle de l'*ISDN*. Le point d'accès à ce réseau se fera par un routeur Cisco 2621. Les buts à atteindre sont :

- configurer le routeur pour des clients possédant des adresses IP statique
- authentifier le client au niveau de l'accès
- configurer le routeur pour des clients possédant des adresses IP dynamique
- se familiariser avec les routeurs Cisco

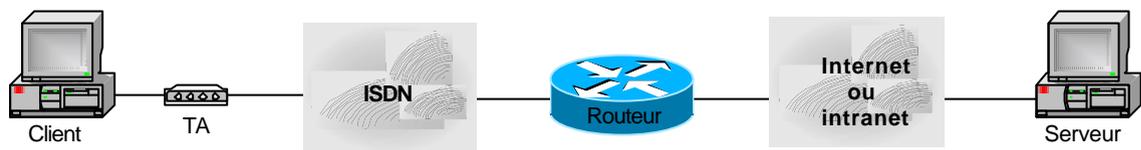


FIGURE 3.1 : Topologie du réseau pour RAS

La mise en place de ce réseau se fera à travers plusieurs étapes permettant de traiter les problèmes potentiels un à un.

3.2. Configuration « routeur à routeur »

3.2.1. Topologie

La topologie « routeur à routeur » (FIGURE 3.2) permet de configurer les routeurs et leurs interfaces en s'affranchissant des problèmes de compatibilité pouvant survenir entre ces derniers et un *TA* (*Terminal Adapter*). Pour ce montage, nous utilisons deux routeurs. Le premier est un Cisco 2621 possédant deux interfaces Ethernet 10-100 Mb/s (*fastEthernet* 0/0 & 0/1) et quatre interfaces *BRI* (*Basic Rate Interface* 1/0, 1/1, 1/2 & 1/3) pour l'*ISDN*. Le deuxième est un Cisco 2500 possédant une interface Ethernet (Ethernet 0) et une interface *BRI* (*Basic Rate Interface* 0).

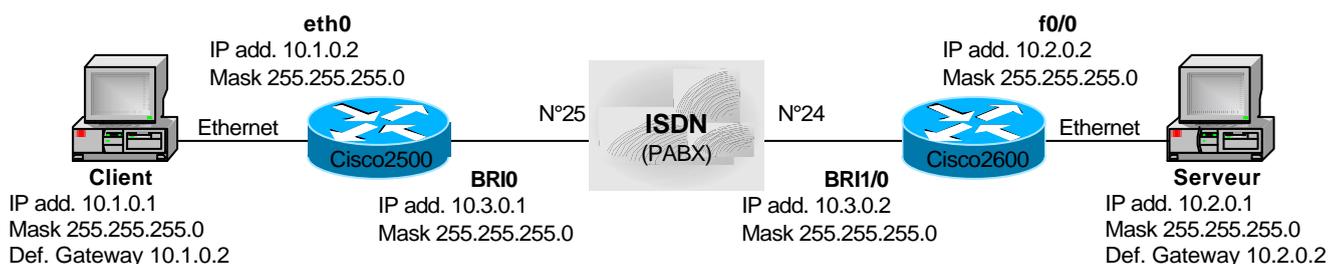


FIGURE 3.2 : Topologie « routeur à routeur »

Comme le montre la figure ci-dessus, toutes les adresses IP sont des adresses privées. Ce réseau est donc totalement autonome (*intranet*). Un PC joue le rôle du client et un deuxième simule un serveur se trouvant sur Internet.

3.2.2. Configuration des PCs (sur Windows 2000)

Pour commencer, il faut configurer les PCs comme le montre la FIGURE 3.2. Pour cela, cliquez sur **Start**, pointez sur **Settings** et cliquez sur **Control Panel**, puis sur **Network and Dial-up Connexions**. Faites un clic droit sur **Local Area Connexion**, puis **Properties**. La fenêtre ci-dessous apparaît.

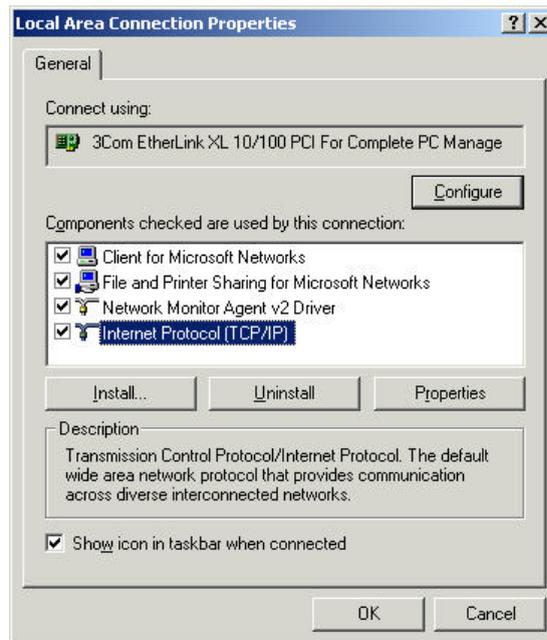


FIGURE 3.3 : Local Area Connexion Properties

Cliquez sur **Internet Protocol (TCP/IP)**, puis sur **Properties**. La fenêtre suivante s'ouvre :

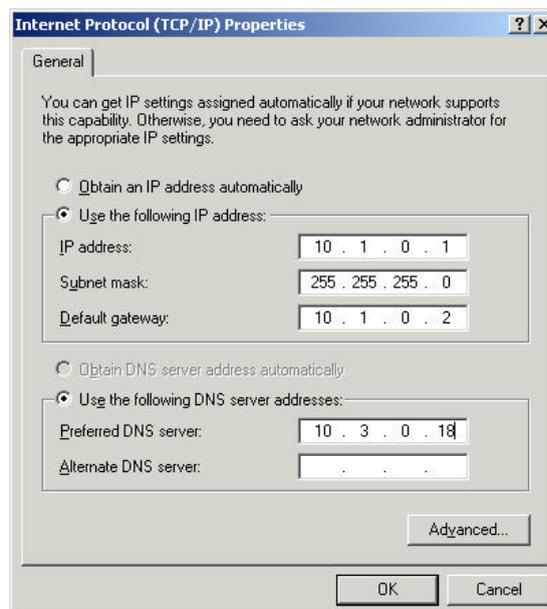


FIGURE 3.4 : Internet Protocol (TCP/IP) Properties

Cliquez sur **Use the following IP address**, puis remplir les différents champs selon la FIGURE 3.2. Terminer en cliquant sur **OK**, puis **OK**.

Remarques : Le serveur DNS n'est pas utilisé dans cette topologie. Toutefois il est nécessaire de remplir le champ **Preferred DNS server** (par ex. 10.3.0.18), sinon Windows essaie d'obtenir une adresse automatiquement.

3.2.3. Configuration et test des routeurs (HDLC)

Dans un premier temps, la configuration des deux routeurs est identique. Cette dernière leur permet d'établir la connexion *ISDN* lorsqu'un paquet doit être envoyé au réseau distant. Voici comment configurer le routeur Cisco2600.

Pour commencer, on peut donner un nom (*hostname*) au routeur. Ce nom apparaîtra comme prompt du routeur, mais surtout, il sera utilisé pour l'authentification d'une connexion PPP (voir § 3.2.4).

```
➤ Router(config)# hostname Cisco2600
➤ Cisco2600(config)#
```

Le routeur, lorsqu'il reçoit des paquets, doit savoir vers qui les envoyer, c'est à dire qu'il doit posséder une table de routage. Cette table est définie de manière statique avec la commande « ip route ».

ip route *prefix mask address*

- *prefix* : Réseau distant devant être atteint.
- *mask* : Masque du sous-réseau distant.
- *address* : Adresse du routeur distant.

Selon la FIGURE 3.2, si le Cisco2600 reçoit un paquet du client à destination du réseau 10.1.0.0, il doit envoyer le paquet vers l'interface 10.3.0.1.

```
➤ Cisco2500(config)# ip route 10.1.0.0 255.255.255.0 10.3.0.1
```

Il faut aussi spécifier le type de trafic autorisé à établir la connexion *ISDN*.

dialer-list *dialer-group protocol protocol-name {permit | deny | list access-list-number}*

- *dialer-group* : Nombre faisant le lien avec une interface.
- *protocol-name* : Protocole prit en considération.
- **permit | deny** : Autoriser ou non l'établissement de la liaison par ce trafic.
- **list** : Permet le renvoi vers une liste de contraintes.

Dans notre cas, tout le trafic IP est autorisé.

```
➤ Cisco2600(config)# dialer-list 1 protocol ip permit
```

Les types de commutateurs pour l'*ISDN* ne sont pas les mêmes dans tous les pays. En Suisse, il faut configurer le routeur avec «*basic-net3*». Pour cela, on utilise la commande suivante :

```
➤ Cisco2600(config)# isdn switch-type basic-net3
```

Pour configurer l'interface Ethernet, taper :

```
➤ Cisco2600(config)# interface fastethernet0/0
```

Comme le montre la FIGURE 3.2, cette interface possède une adresse IP. La commande suivante permet de la configurer :

ip address *ip-address mask*

- *ip-address* : Adresse ip de l'interface.
- *mask* : Masque de sous-réseau.

```
➤ Cisco2600(config-if)# ip address 10.2.0.2 255.255.255.0  
➤ Cisco2600(config-if)# exit
```

Passons à la configuration de l'interface BRI :

```
➤ Cisco2600(config)# interface BRI1/0  
➤ Cisco2600(config-if)# ip address 10.3.0.2 255.255.255.0
```

Il faut configurer l'interface pour lui indiquer le type d'encapsulation des paquets à employer. Pour un *RAS*, qui fonctionne avec une liaison de type point à point, Windows utilise le protocole *PPP* (*Point to Point Protocol*). C'est ce protocole qui sera choisi lors de la configuration des équipements. Avec ce protocole, les appareils passent par une phase où ils s'authentifient auprès de leur correspondant (voir § 3.2.4). C'est pourquoi, lors du premier test, une encapsulation HDLC est préférée afin de s'affranchir de cette phase d'authentification et de vérifier le bon fonctionnement de la configuration des routeurs.

```
➤ Cisco2600(config-if)# encapsulation HDLC
```

La commande suivante permet de faire le lien entre la *dialer-list* décrite précédemment et l'interface.

dialer-group *group-number*

- *group-number* : Indique le numéro de la *dialer-liste* à laquelle l'interface appartient. Ce numéro doit être identique à celui utilisé plus haut.

```
➤ Cisco2600(config-if)# dialer-group 1
```

En suite, il faut indiquer le numéro de téléphone que doit composer le routeur afin d'atteindre le réseau distant.

dialer map *protocol next-hop-address* [**name** *hostname*] [**speed** **56** | **64**]
[**broadcast**] *dialer-string*

- *protocol* : IP, IPX, DECnet, etc.
 - *next-hop-address* : Adresse du routeur adjacent.
 - **name** *hostname* : Optionnel. *Hostname* du routeur adjacent. Ce nom est utilisé pour l'authentification PPP. Il permet de faire le lien avec le *password* à utiliser pour accéder à ce routeur.
 - **speed** **56** | **64** : Optionnel. Spécifie le débit en kbps de la ligne *ISDN*. Par défaut, la valeur est de 64.
 - **broadcast** : Indique que les *broadcasts* et les *multicasts* peuvent être expédiés au routeur adjacent.
 - *dialer-string* : Numéro de téléphone composé lorsqu'un paquet doit aller à l'adresse *next-hop-address*.
- Cisco2600(config-if)# **dialer map ip 10.3.0.1 name Cisco2500 25**

Pour finir, un *timer* mettant fin à la liaison *ISDN* doit être mis en place.

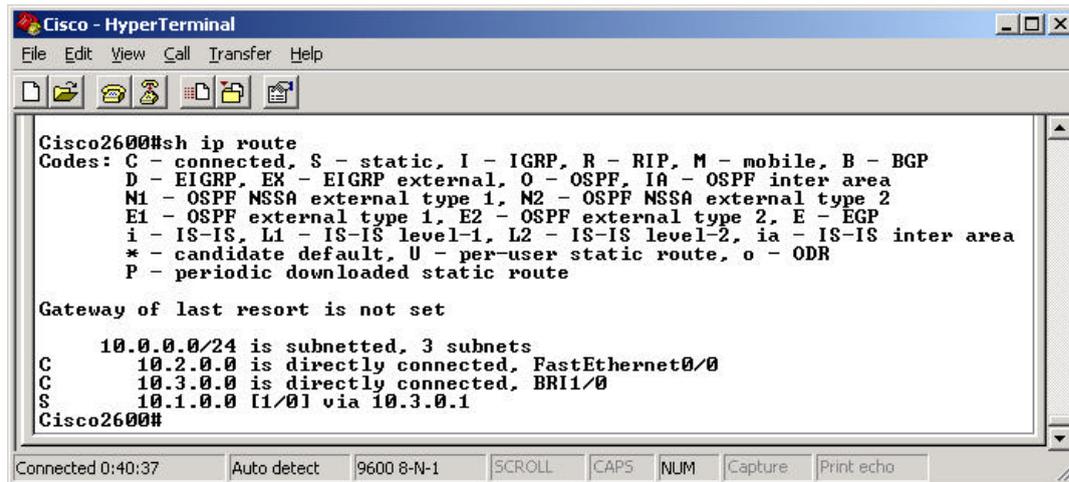
dialer idle-timeout *seconds*

- *seconds* : Nombre de secondes après le passage du dernier paquet jusqu'à la déconnection de la ligne *ISDN* (voir § 3.3.5)
- Cisco2600(config-if)# **dialer idle-timeout 120**

Par défaut, la fonction de routage des paquets IP est activée sur les routeurs. Si ce n'est pas le cas, il faut taper la commande suivante :

➤ Cisco2600(config)# **ip routing**

En tapant *show ip route*, il est possible de voir toutes les routes connues par le routeur. Sur la FIGURE 3.5, on retrouve la route définie avec la commande *ip route*. Elle est précédée d'un S qui indique quelle est une route statique. On remarque que les interfaces directement connectées au routeur sont automatiquement ajoutées à la table de routage. Ces dernières sont précédées d'un C. La commande *show running-config* permet de voir la configuration actuelle du routeur (annexe *Configuration « routeur à routeur » Cisco 2600 HDLC*)



```
Cisco2600#sh ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is not set

 10.0.0.0/24 is subnetted, 3 subnets
C       10.2.0.0 is directly connected, FastEthernet0/0
C       10.3.0.0 is directly connected, BRI1/0
S       10.1.0.0 [1/0] via 10.3.0.1
Cisco2600#
```

FIGURE 3.5 : Show ip route

Afin de ne pas trop charger ce document, seules les modifications à la configuration actuelle seront indiquées. Les configurations complètes des routeurs se trouvent en annexe.

La configuration du routeur Cisco2500 s'effectue de la même manière avec quelques différences. La route est :

➤ Cisco2500(config)# **ip route 10.2.0.0 255.255.255.0 10.3.0.2**

Les interfaces à configurer sont *Ethernet 0* et *BRI 0* selon la FIGURE 3.2. Le *dialer map* est aussi différent :

➤ Cisco2500(config-if)# **dialer map ip 10.3.0.2 name Cisco2600 24**

Pour la configuration complète, voir l'annexe *Configuration « routeur à routeur » Cisco 2500 HDLC*.

En effectuant un *ping 10.2.0.1* (serveur) depuis le poste client, on a vérifié le bon fonctionnement de l'installation. Des LED vertes, située en dessous des connecteurs du routeur, permettent de contrôler que les canaux B de l'*ISDN* sont activés. Le but n'est pas de détailler cet échange, car la configuration qui nous intéresse emploie le protocole PPP, mais simplement de tester le fonctionnement des routeurs et des PCs sans se préoccuper de l'authentification. Cette vérification étant concluante, nous pouvons nous concentrer sur PPP.

3.2.4. Point to Point Protocol (PPP)

Le protocole *PPP* (RFC 1661) permet d'encapsuler des paquets, de différentes sortes de protocoles, sur des liaisons point à point. Dans le cas étudié, la liaison point à point est le canal B de l'*ISDN* et les paquets envoyés sont de type IP.

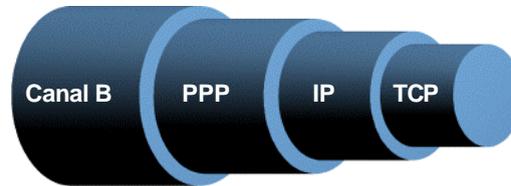


FIGURE 3.6 : Encapsulation des différents protocoles

Ce protocole offre :

- Une méthode d'encapsulation des paquets.
- Un contrôle de la liaison (*LCP* : *Link Control Protocol*), pour l'établissement, la configuration et le test de la connexion.
- La possibilité d'authentifier les équipements aux extrémités de la liaison (*PAP* : *Password Authentication Protocol* et *CHAP* : *Challenge-Handshake Authentication Protocol*)
- Une famille de protocole de contrôle (*NCP* : *Network Control Protocol*).

L'encapsulation des paquets est réalisé suivant la figure ci-dessous.

PPP Header	Protocol	Data	CRC
2	2	variable	2/4

FIGURE 3.7 : Format de la trame PPP

- *PPP Header* : Ce champ est de deux octets et vaut toujours 0xFF03.
- *Protocol* : Ce champ est d'un octet. Il identifie le type de protocole encapsulé dans le champ *Data*. Voici quelques valeurs qu'il peut prendre :

0xC021	Link Control Protocol (LCP)
0xC023	Password Authentication Protocol (PAP)
0xC025	Link Quality Report
0xC223	Challenge Handshake Authentication Protocol (CHAP)
0x8021	Internet Protocol Control Protocol (IPCP)
0x0021	Internet Protocol (IP)
0x003D	Multilink Protocol (MP)

- *Data* : Ce champ est de taille variable. Par défaut, il est de 1500 bytes, mais cette valeur est négociable lors de l'initialisation de la liaison (*LCP*). Le format des données dépend du protocole encapsulé.
- *CRC* : Redondance de 16 bits (défaut) ou 32 bits.

Voici les différentes phases d'établissement du protocole PPP après l'activation de la liaison point à point (canal B). La première est l'envoi de paquets *LCP* (*Link Control Protocol*). Ils permettent de négocier les paramètres de la connexion comme, par exemple, la longueur des paquets, l'utilisation des deux canaux B (*multilink*) et le type de protocole d'authentification. Ils permettent aussi de tester la ligne. La figure ci dessous représente la négociation LCP.

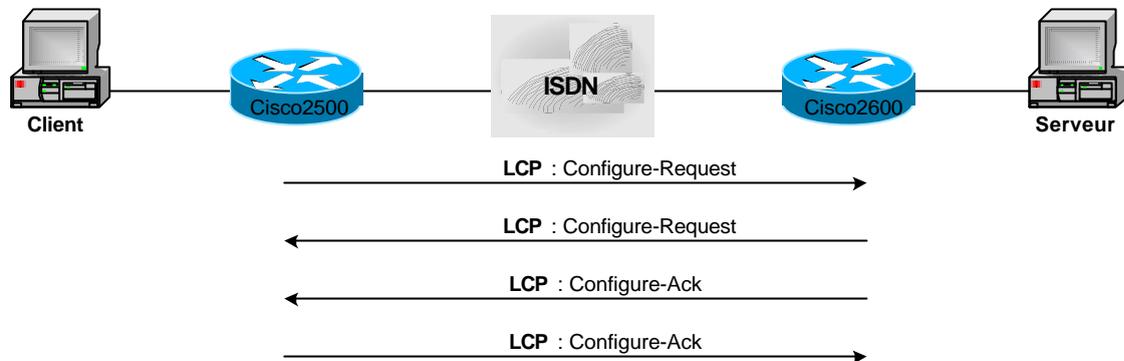


FIGURE 3.8 : Négociation LCP

PPP Header	Protocol (LCP)	Code	Identifïer	Length	Data	CRC
2	2	1	1	2	variable	2/4

FIGURE 3.9 : Format de la trame LCP

- *Code* : Ce champ est d'un octet. Il identifie le type de paquet LCP. Voici les valeurs qu'il peut prendre :
 - 1 Configure-Request
 - 2 Configure-Ack
 - 3 Configure-Nak
 - 4 Configure-Reject
 - 5 Terminate-Request
 - 6 Terminate-Ack
 - 7 Code-Reject
 - 8 Protocol-Reject
 - 9 Echo-Request
 - 10 Echo-Reply
 - 11 Discard-Request
- *Identifïer* : Ce champ est d'un octet. La valeur de ce champ est identique entre un paquet de requête (*Configure-Request*) et l'acquittement lui correspondant (*Configure-Ack*).
- *Length* : Ce champ est de deux octets. Il indique la longueur du paquet LCP. Cette longueur inclut les champs *Code*, *Identifïer*, *Length* et *Data*.
- *Data* : Ce champ est de taille variable. Il peut contenir une ou plusieurs options de configuration. Seul les options que les équipements veulent négocier sont envoyées. Chaque option est de la forme suivante :

Type	Length	Data
1	1	variable

FIGURE 3.10 : Format des options de configuration LCP

- *Type* : Ce champ est d'un octet. Il identifie le type d'options de configuration LCP. Voici quelques exemples des valeurs possibles :
 - 1 Maximum-Receive-Unit
 - 3 Authentication-Protocol
 - 4 Quality-Protocol
 - 17 Multilink Maximum Received Reconstructed Unit
 - 18 Multilink Short Sequence Number Header Format
 - 19 Endpoint Discriminator
- *Length* : Ce champ est d'un octet. Il indique la longueur de cette option de configuration.
- *Data* : Ce champ peut être nul ou de longueur variable. Il contient les informations spécifiques à cette option de configuration. Le format et la longueur de ce champ sont déterminés par *Type* et *Length*.

Pour plus de détails, le lecteur peut se référer à la RFC 1661 (The Point to Point Protocol).

La deuxième phase de l'établissement est l'authentification de l'interlocuteur (RFC 1334). Les protocoles employés sont *PAP* ou *CHAP*. La négociation du protocole utilisé est faite lors de la négociation *LCP*.

Le routeur Cisco2500 a une version d'IOS plus ancienne que celle du Cisco2600. Cela pose des problèmes de compatibilité lors de l'authentification. C'est pourquoi un deuxième routeur Cisco2600 est utilisé en remplacement du Cisco2500. Afin de garder les mêmes configurations, le *hostname* Cisco2500 est conservé pour le second routeur.

Password Authentication Protocol (PAP) est un protocole d'authentification très simple. Le routeur coté « client » envoie, dans le paquet *Authenticate-Request*, son *hostname* et son *password*. Le routeur coté « serveur » vérifie qu'il possède un *username* et un *password* correspondant à ce paquet. Si c'est le cas, il envoie *Autenticat-Ack* qui indique que le routeur « client » est authentifié. Ce protocole envoie le *username* et le *password* en clair.

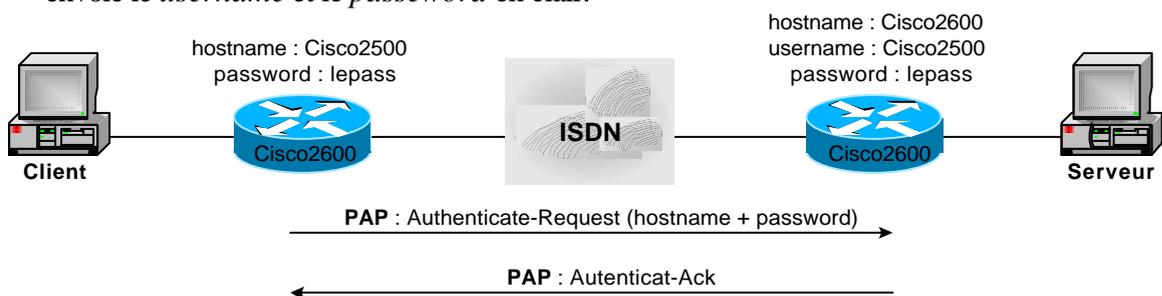


FIGURE 3.11 : Echanges PAP

La figure ci-dessous représente la trame pour le protocole PAP.

PPP Header	Protocol PAP	Code	Identifiant	Length	Data	CRC
2	2	1	1	2	variable	2/4

FIGURE 3.12 : Trame PAP

- *Code* : Ce champ est d'un octet. Il identifie le type de paquet PAP. Voici les valeurs qu'il peut prendre :
 - 1 Authenticate-Request
 - 2 Authenticate-Ack
 - 3 Authenticate-Nak
- *Identifiant* : Ce champ est d'un octet. La valeur de ce champ est identique entre un paquet de requête et l'acquittement lui correspondant.
- *Length* : Ce champ est de deux octets. Il indique la longueur du paquet *PAP*. Cette longueur inclut les champs *Code*, *Identifiant*, *Length* et *Data*.
- *Data* : Ce champ est de taille variable. Le format des données est déterminé par le champ *Code*.

Pour plus d'information, le lecteur peut se référer à la RFC 1334.

Challenge-Handshake Authentication Protocol (CHAP) a l'avantage de ne pas envoyer de *password* en clair. Après la phase d'établissement, le routeur du côté serveur envoie un paquet *Challenge* au routeur client afin de l'authentifier. Ce paquet contient une variable aléatoire et l'identité du routeur serveur (*hostname*). La taille de la variable aléatoire dépend de la méthode utilisée pour la générer, mais est indépendante de l'algorithme de « hashage ». Le routeur client concatène cette variable aléatoire et son *password* pour calculer un *hash*. (16 bytes avec MD5). La valeur du *hash*, ainsi que l'identité du routeur client sont renvoyées au routeur côté serveur dans un paquet *Response*. Ce routeur possède une liste des clients autorisés à se connecter (*username*). Il recalculé le *hash*, en utilisant le *password* correspondant au client et la variable aléatoire, et la compare avec la *hash* reçu du client. S'ils sont identiques, il renvoie le message *Success* qui indique au client qu'il est authentifié.

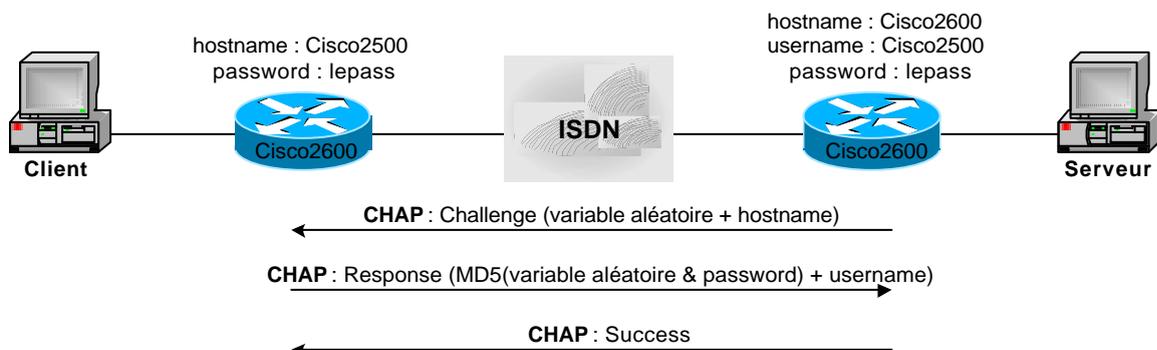


FIGURE 3.13 : Transfert CHAP

La figure ci-dessous représente la trame CHAP.

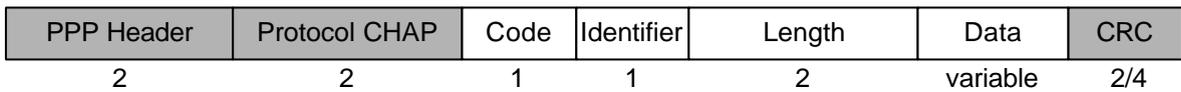


FIGURE 3.14 : Trame CHAP

- *Code* : Ce champ est d'un octet. Il identifie le type de paquet CHAP. Voici les valeurs qu'il peut prendre :
 - 1 Challenge
 - 2 Response
 - 3 Success
 - 4 Failure
- *Identifiant* : Ce champ est d'un octet. La valeur de ce champ est identique entre un paquet de requête et l'acquittement lui correspondant.
- *Length* : Ce champ est de deux octets. Il indique la longueur du paquet *CHAP*. Cette longueur inclut les champs *Code*, *Identifiant*, *Length* et *Data*.
- *Data* : Ce champ est de taille variable. Le format des données est déterminé par le champ *Code*. Pour les paquets *Challenge* ou *Response*, le champ data prend la forme illustrée à la FIGURE 3.15. Pour les paquets *Success* et *Failure*, ce champ peut être nul ou de longueur variable. Le protocole n'implémente pas de message pour ces deux paquets, toutefois, ce champ peut être utilisé pour envoyer un message lisible par l'utilisateur.

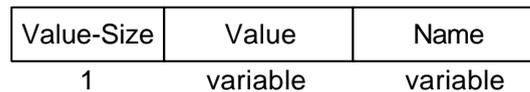


FIGURE 3.15 : Format des data des paquets *Challenge* et *Response*

- *Value-Size* : Ce champ est d'un octet et indique la longueur du champ *Value*.
- *Value* : Ce champ peut être d'un ou plusieurs octets. Il contient la variable aléatoire pour les paquets *Challenge* ou le *hash* pour les paquets *Response*.
- *Name* : Ce champ est de longueur variable. Il contient l'identité du système à l'origine du paquet (*hostname*). La longueur de ce champ est déterminée grâce au champ *Length*.

Pour plus d'information, le lecteur peut se référer à la RFC 1994.

La dernière phase de l'établissement est la configuration de la couche de niveau 3 (*NCP : Network Control Protocol*). Le protocole utilisé pour cette configuration dépend du type de paquets à transporter. Pour des paquets IP, le protocole utilisé est *IPCP (Internet Protocol Control Protocol)*. Ce protocole permet, par exemple, de

configurer l'adresse IP du client et de lui indiquer l'adresse du serveur *DNS* (voir § 3.5.1). Il permet aussi de négocier un protocole de compression des entêtes *TCP/IP*. *IPCP* utilise le même mécanisme d'échange de paquets que *LCP* (voir FIGURE 3.8). Par contre, il dispose d'un ensemble d'options de configuration lui étant propre.

PPP Header	Protocol IPCP	Code	Identifiant	Length	Data	CRC
2	2	1	1	2	variable	2/4

FIGURE 3.16 : Format de la trame *IPCP*

- *Code* : Champ d'un octet. Identifie le type de paquet *IPCP*. Les valeurs sont identiques à *LCP*
- *Identifiant* : Champ d'un octet. Valeur du champ identique entre un paquet de requête et l'acquittement lui correspondant.
- *Length* : Champ de deux octets. Indique la longueur du paquet *IPCP*. Cette longueur inclut les champs *Code*, *Identifiant*, *Length* et *Data*.
- *Data* : Ce champ est de taille variable. Il peut contenir une ou plusieurs options de configuration. Seules les options que les équipements veulent négocier sont envoyées. Chaque option est de la forme suivante :

Type	Length	Address
1	1	4

FIGURE 3.17 : Format des options de configuration *IPCP*

- *Type* : Champ d'un octet. Identifie le type d'options de configuration *IPCP*. Voici quelques exemples des valeurs possibles :
0x03 IP Address 0x81 PrimaryDNS 0x83 SecondaryDNS
 - *Length* : Champ d'un octet. Vaut toujours 6
 - *Data* : Champ de quatre octets. Contient les adresses
- Pour plus d'information, le lecteur peut se référer à la RFC 1332.

3.2.5. Configuration des routeur pour PPP

La configuration des routeurs pour *PPP* ne change pas beaucoup de celle vue précédemment (§ 3.2.3). Voici les changements à apporter pour Cisco2600.

Il faut utiliser l'encapsulation *PPP* pour l'interface *BR1/0*.

```
➤ Cisco2600(config-if)# encapsulation PPP
```

Il faut aussi indiquer le protocole d'authentification qui sera employé à l'établissement de la liaison. Nous utiliserons *CHAP*, car ce protocole est plus sûr que *PAP* (voir § 3.2.4).

```
➤ Cisco2600(config-if)# ppp authentication chap callin
```

Callin est optionnel. Il indique au routeur qu'il doit procéder à l'authentification de son interlocuteur uniquement lorsqu'il est appelé. Si cette option n'est pas utilisée, l'authentification est aussi effectuée lorsque le routeur est l'appelant. Nous utilisons *callin*, car pour un *RAS*, le but du routeur, offrant le point d'accès au réseau, est uniquement d'authentifier le client (qui est l'appelant).

La commande suivante permet de créer une liste de *username* et de *password*. Cette liste permet, au routeur client et au routeur serveur, de connaître le *password* à employer pendant la phase d'établissement. Cela est fait par comparaison des *usernames* de la liste avec le *hostname* du routeur distant.

```
username user name password type password
```

- *user name* : Nom des clients autorisés à se connecter au routeur.
- *type* : 0 indique un *password* sauvegardé en clair dans le routeur.
7 indique un *password* qui est caché dans la sauvegarde sur routeur.
- *password* : Mot de passe partagé avec le client.

```
➤ Cisco2600(config-if)# username Cisco2500 password 0 pass
```

La configuration complète de ce routeur se trouve dans l'annexe *Configuration « routeur à routeur » Cisco 2600 PPP*

La configuration du second routeur s'effectue de la même manière que celle vue ci-dessus, avec ces quelques différences :

- *hostname* est Cisco2500.
- *username* est Cisco2600 (*password* identique pour les deux routeurs).
- Adresses des interfaces selon la FIGURE 3.2.
- ip route 10.2.0.0 255.255.255.0 10.3.0.2
- dialer map ip 10.3.0.2 name Cisco2600 24

Pour vérifier le bon fonctionnement de l'installation, on effectue un *ping* sur le serveur (10.2.0.1) depuis le poste client (FIGURE 3.18). Sur la FIGURE 3.19 On constate que le premier paquet est perdu. Cela est dû au temps d'établissement de la liaison *ISDN* entre les deux routeurs.

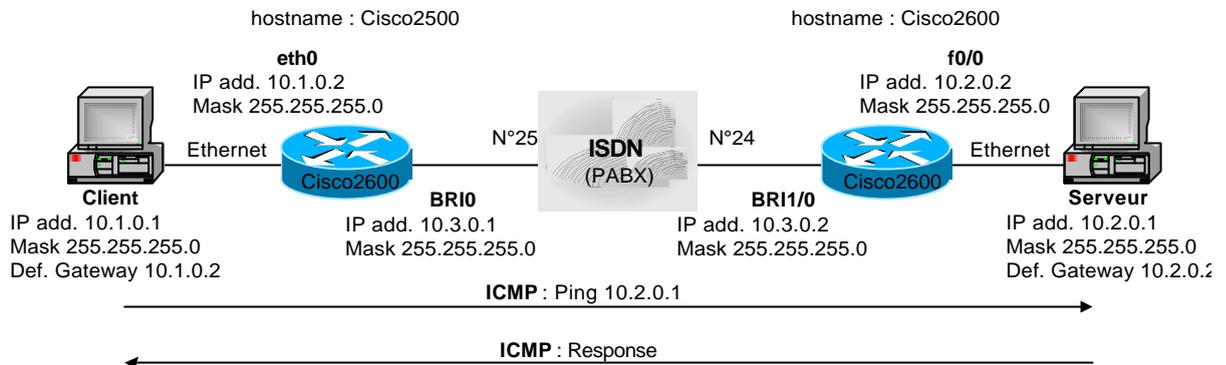


FIGURE 3.18 : Ping sur le serveur

```

C:\>ping 10.2.0.1
Pinging 10.2.0.1 with 32 bytes of data:
Request timed out.
Reply from 10.2.0.1: bytes=32 time=30ms TTL=126
Reply from 10.2.0.1: bytes=32 time=20ms TTL=126
Reply from 10.2.0.1: bytes=32 time=20ms TTL=126
Ping statistics for 10.2.0.1:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 20ms, Maximum = 30ms, Average = 17ms
C:\>ping 10.2.0.1
C:\>

```

FIGURE 3.19 : Capture du ping sur le serveur

```

Cisco2600>
04:39:113827783699: %LINK-3-UPDOWN: Interface BRI1/0:1, changed state to up
04:39:113816633317: BRI/0:1 PPP: Treating connection as a callin
04:39:26: %ISDN-6-LAYER2UP: Layer 2 for Interface BRI/0, TEI 75 changed to up
04:39:27: BRI/0:1 CHAP: O CHALLENGE id 20 len 30 from "Cisco2600"
04:39:27: BRI/0:1 CHAP: I RESPONSE id 20 len 30 from "Cisco2500"
04:39:27: BRI/0:1 CHAP: O SUCCESS id 20 len 4
04:39:28: %LINEPROTO-5-UPDOWN: Line protocol on Interface BRI1/0:1, changed state to up
04:39:32: %ISDN-6-CONNECT: Interface BRI1/0:1 is now connected to 25 Cisco2500
04:41:125413372471: %ISDN-6-DISCONNECT: Interface BRI1/0:1 disconnected from 25 Cisco2500, call lasted 123 seconds
04:41:29: %LINK-3-UPDOWN: Interface BRI1/0:1, changed state to down
04:41:30: %LINEPROTO-5-UPDOWN: Line protocol on Interface BRI1/0:1, changed state to down
04:41:40: %ISDN-6-LAYER2DOWN: Layer 2 for Interface BRI/0, TEI 75 changed to down
n_

```

FIGURE 3.20 : Capture de la console

La FIGURE 3.20 montre l'activation de la ligne *ISDN*, la phase d'authentification et enfin, quand le *idle timeout* est atteint, la désactivation de l'*ISDN*. Grâce à la

commande *debug ppp authentication*, on retrouve sur cette capture les paquets correspondant à l'authentification CHAP représentés à la FIGURE 3.13. On retrouve aussi certains champs décrits au paragraphe 3.2.4. Par exemple, pour le paquet *Response*, le champ *Identifieur* vaut 20 et le champ *Length* vaut 30 (1 octet pour le *Code*, 1 octet pour l'*Identifieur*, 2 octets pour *Length*, 1 octet pour *Value-Size*, 16 octets pour le *Value* (hash MD5) et 9 octets pour *Name* (Cisco2500 en code ASCII)).

3.3. Configuration « TA à routeur » avec adresse statique

3.3.1. Topologie

Cette topologie correspond plus à celle utilisée par un particulier pour un RAS. En effet, du côté client, le routeur est remplacé par un TA (*Terminal Adaptor*) et la liaison Ethernet laisse place à *USB*. Le reste de la configuration ne change pas. Dans cette première configuration, le client possède une adresse statique.

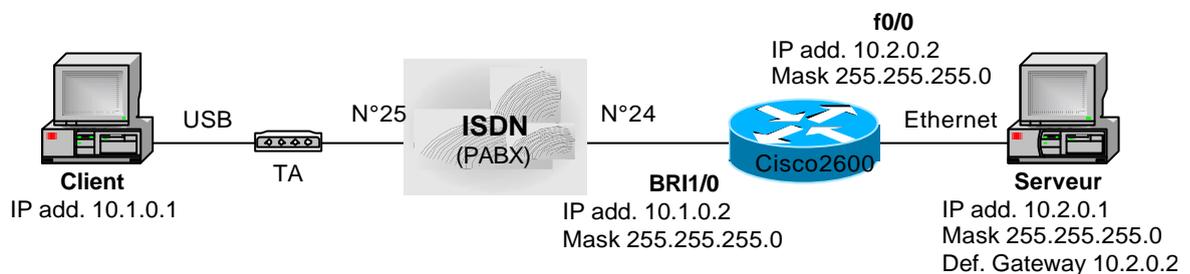


FIGURE 3.21 : Topologie du réseau

3.3.2. Configuration des PCs

La configuration du PC client consiste en l'installation du TA (Fritz!Card USB) et à la configuration du niveau IP. Pour commencer, il faut connecter le TA selon la FIGURE 3.21. La connexion au PC se fait en utilisant une interface *USB*. Le TA est *Plug and Play*. Lorsqu'il est connecté au PC, Windows 2000 détecte ce nouveau matériel et ouvre la fenêtre suivante :



FIGURE 3.22 : ISDN Switch Type or D-channel Protocol

Sélectionnez le protocole **European ISDN (DSS1)** dans le menu déroulant, puis faites un clic sur **Next**. Cliquez une deuxième fois sur **Next**. L'installation du matériel est terminée. A présent, il faut créer une nouvelle connexion utilisant le TA. Pour cela, cliquez sur **Start**, pointez sur **Settings** et cliquez sur **Control Panel**, puis sur **Network and Dial-up Connexion**. Dans la fenêtre, faites un clic droit, puis cliquez sur **New Connexion...** Cliquez sur **Next**. La fenêtre ci dessous apparaît.

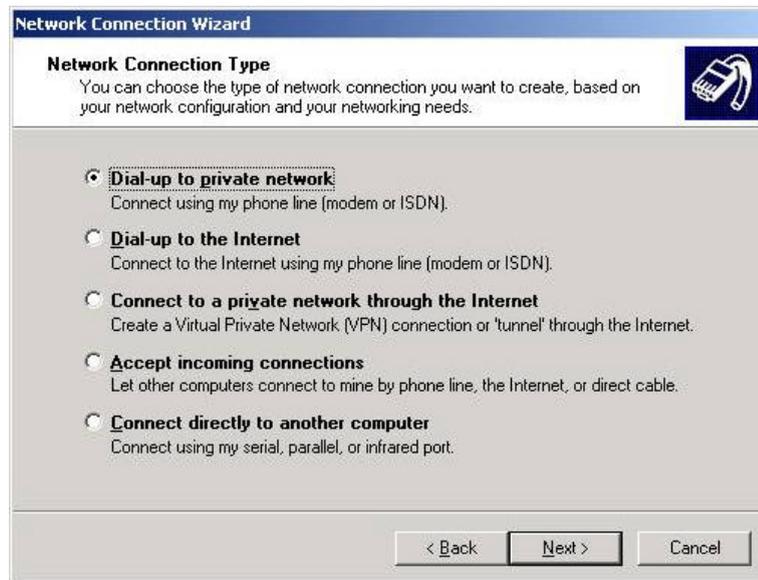


FIGURE 3.23 : Network Connexion Type

Sélectionnez **Dial-up to private network**, puis cliquez sur **Next**.

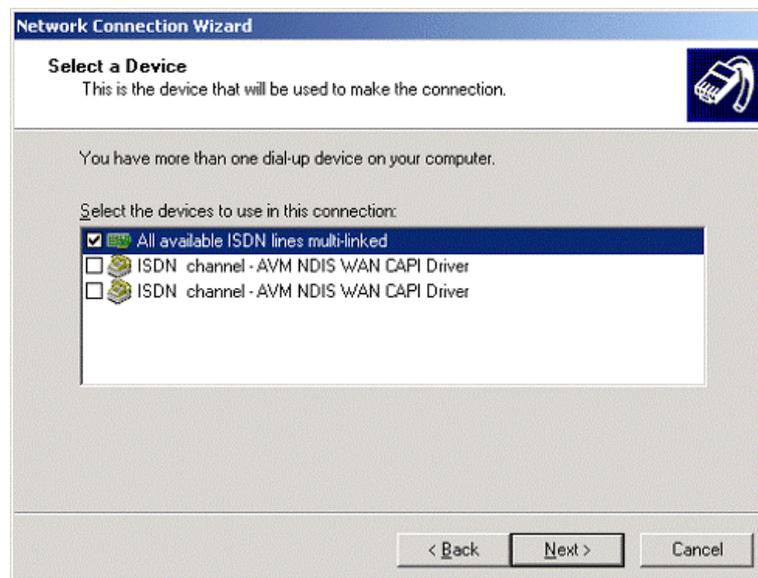


FIGURE 3.24 : Select a Device

Dans cette fenêtre, il est possible de choisir le nombre de canaux B utilisés pour la connexion *ISDN*. En sélectionnant **All available ISDN lines multi-linked**, le TA utilisera les deux canaux B. Cliquez sur **Next**.

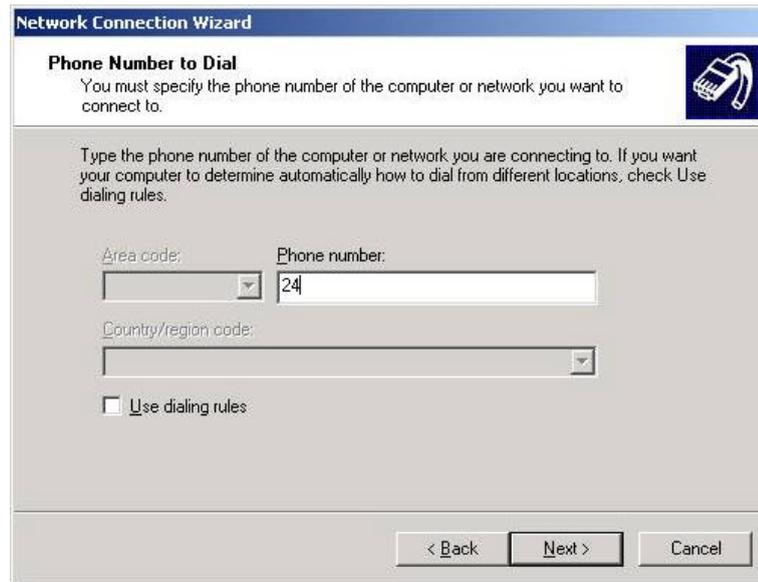


FIGURE 3.25 : Phone Number to Dial

Indiquez le numéro de téléphone à composer. Dans notre cas, le 24. Cliquez sur **Next**. Sélectionnez **Only for myself**, puis cliquez sur **Next**. Donnez un nom à la connexion (ex. « Dial-up connection »), puis cliquez sur **Finish**. Une nouvelle icône apparaît dans la fenêtre **Network and Dial-up Connexion**. Faites un clic droit sur cette icône, puis **Properties**. Sur la fenêtre qui vient de s'ouvrir, cliquez sur l'onglet **Networking**.

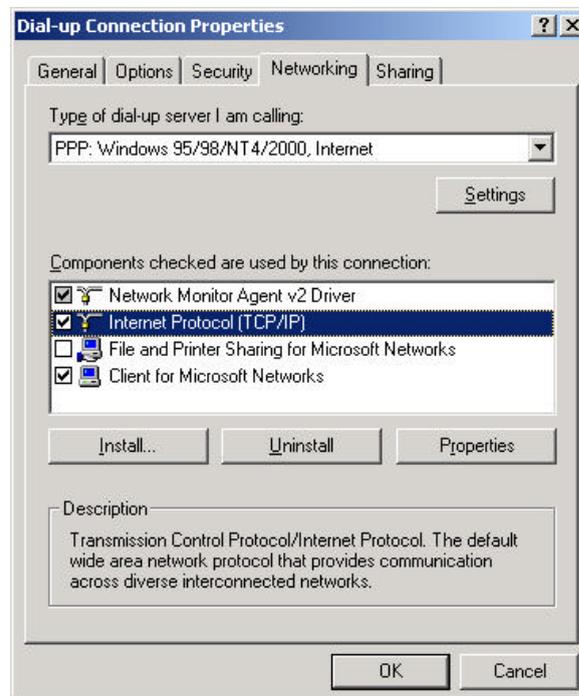


FIGURE 3.26 : Propriétés de connexion

Cliquez sur **Internet Protocol (TCP/IP)**, puis sur **Properties**.

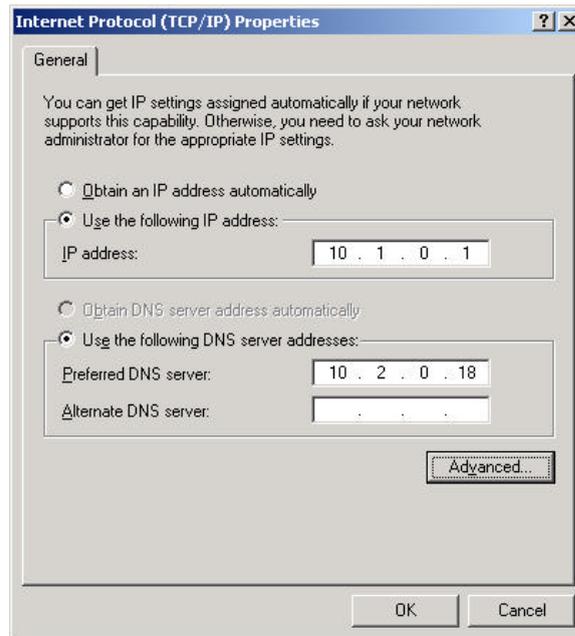


FIGURE 3.27 : Internet Protocol (TCP/IP) Properties

Cliquez sur **Use the following IP address**, puis remplir le champ **IP address** selon la FIGURE 3.21. Cliquez sur **OK**.

Sélectionnez l'onglet **Security**. Sélectionnez **Advanced (custom settings)**, puis cliquez sur **Settings**. La fenêtre suivante s'ouvre.

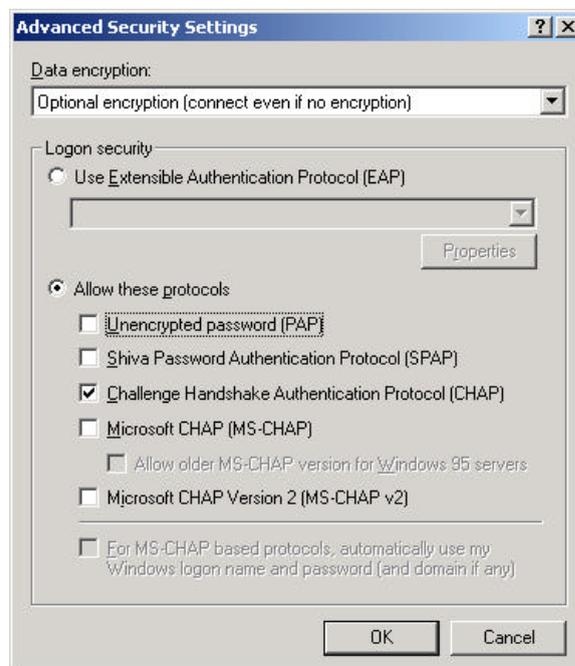


FIGURE 3.28 : Advanced encryption Settings

Sélectionnez le protocole d'authentification à utiliser, dans notre cas *CHAP*, puis, pour finir, cliquez sur **OK**, puis **OK**.

Remarques : On constate qu'il n'y a plus de *Default Gateway* et de *Subnet mask* du coté client. Ces paramètres sont inutiles pour une liaison *PPP* de ce type, car il n'y a qu'un chemin possible. En revanche, le réseau distant devra gérer ces paramètres pour permettre au client d'atteindre d'autres réseaux.

Le serveur DNS n'est pas utilisé dans cette topologie. Toutefois il est nécessaire de remplir le champ **Preferred DNS server** (par ex. 10.2.0.18), sinon Windows essaye d'obtenir une adresse automatiquement.

Le PC client est à présent configuré. La configuration du PC serveur est identique à la configuration « routeur à routeur ».

3.3.3. Configuration du routeur

Voici les quelques modifications à apporter, sur la configuration du Cisco2600, pour cette topologie (FIGURE 3.21).

Il faut configurer l'adresse IP de l'interface *BRI1/0*.

```
➤ Cisco2600(config-if)# ip address 10.1.0.2 255.255.255.0
```

Dans un *RAS*, ce n'est pas le serveur qui appelle le client, mais le client qui appelle le serveur. C'est pourquoi le *dialer map* n'a plus d'utilité dans la configuration du *BRI1/0* (voir remarque sur le *dialer map* § 3.3.4). Il en va de même pour les commandes *dialer-group* et *dialer-list* (voir remarque sur *idle-timeout* § 3.3.5). En ajoutant « *no* » devant une commande, cela permet de la supprimer.

```
➤ Cisco2600(config-if)# no dialer-group 1
➤ Cisco2600(config-if)# no dialer map ip 10.3.0.1 name
  Cisco2500 25
➤ Cisco2600(config)# no dialer-list 1 protocol ip permit
```

Enfin, le dernier changement concerne *idle-timeout*. Dans un *RAS*, c'est généralement le client qui gère le temps de connexion. En effet, il peut rester connecté le temps voulu. Par sécurité, il est possible de spécifier un *idle-timeout*, du coté client, dans l'onglet **Options** des propriétés de connexion. Pour annuler le *idle-timeout* sur le routeur, tapez :

```
➤ Cisco2600(config-if)# dialer idle-timeout 0
```

La configuration complète du routeur se trouve dans l'annexe *Configuration Cisco2600-TA avec adresse IP statique*.

Pour établir la connexion, double-cliquez sur l'icône « Dial-up Connection » créée précédemment (au § 3.3.2). La fenêtre de connexion suivante s'ouvre :

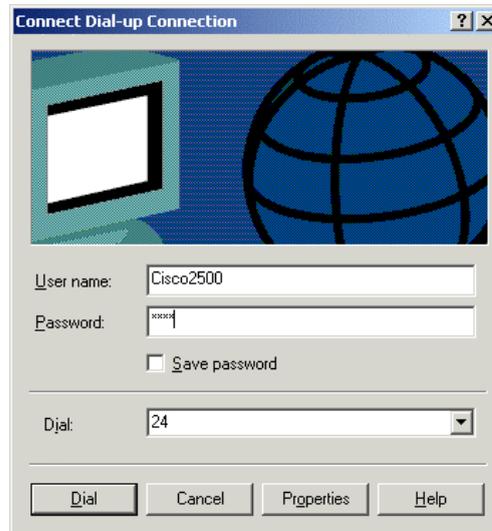


FIGURE 3.29 : Fenêtre de connexion

Le *User name* et le *Password* doivent correspondre à ceux contenus dans le routeur. Ce sont respectivement « Cisco2500 » et « pass ». Ils sont utilisés lors de l'authentification *CHAP*. Sur la console du routeur (FIGURE 3.30) on peut vérifier le bon déroulement de l'authentification.

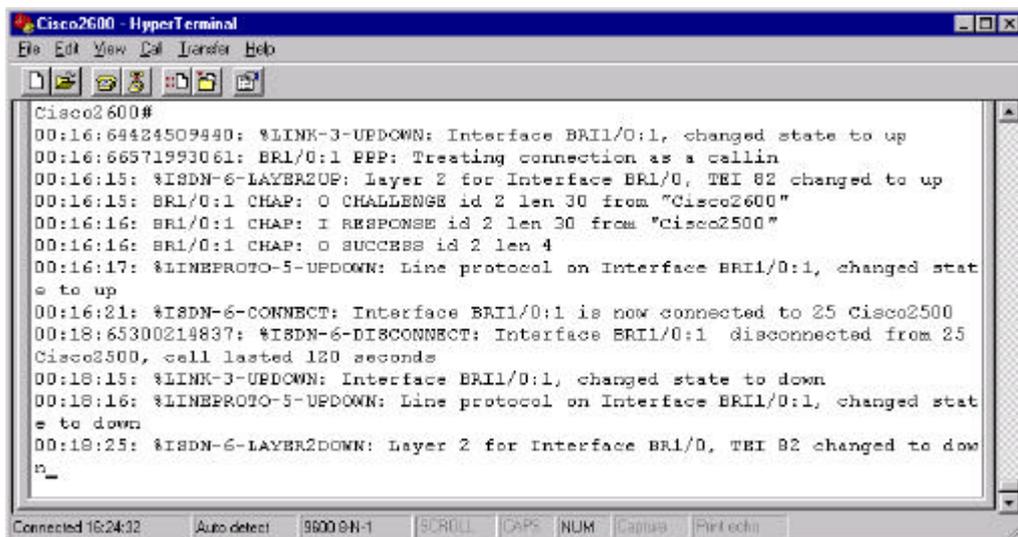
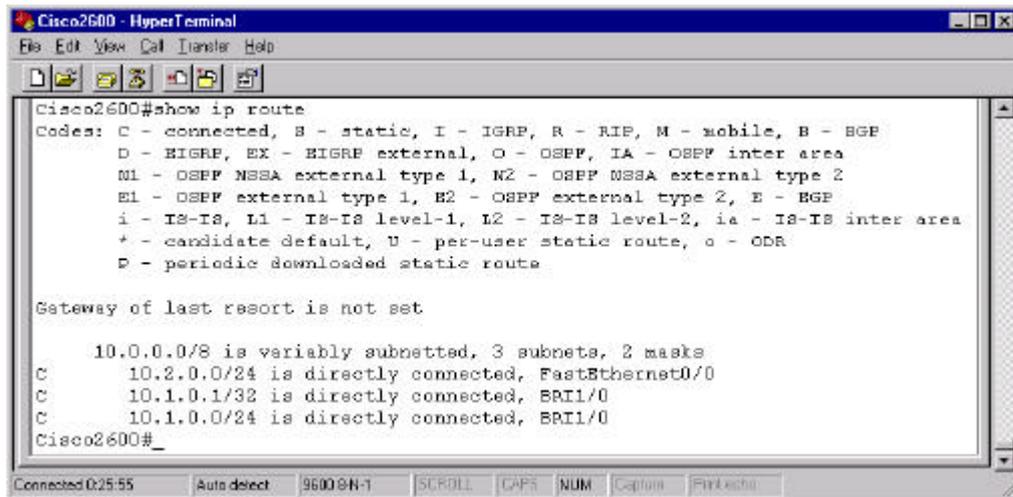


FIGURE 3.30 : Connexion et authentification

En tapant la commande *show iproute* dans la console du routeur (FIGURE 3.31), on constate que l'adresse du client est présente comme directement connecté sur *BR1/0*. On remarque aussi le masque de 32 bits à 1du client. Cela semble logique pour une connexion *PPP*, car tous les paquets du client changent de sous-réseau.

En effectuant un *ping*, depuis le poste client, sur le serveur (10.2.0.1), il est possible de vérifier le bon fonctionnement de la connexion.



```

Cisco2600 - HyperTerminal
File Edit View Call Transfer Help
Cisco2600#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - BGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is not set

 10.0.0.0/8 is variably subnetted, 3 subnets, 2 masks
C       10.2.0.0/24 is directly connected, FastEthernet0/0
C       10.1.0.1/32 is directly connected, BRI1/0
C       10.1.0.0/24 is directly connected, BRI1/0
Cisco2600#

```

FIGURE 3.31 : Routes de Cisco2600

3.3.4. Remarques sur le « dialer map »

Si une ligne de commande *dialer map* est présente dans la configuration de l'interface *BRI* utilisée (ex. *dialer map ip 10.3.0.1 name Cisco2500 25*), et que le PC client est configuré pour obtenir une adresse IP dynamiquement, celui-ci obtiendra l'adresse se trouvant dans la commande *dialer map* (10.3.0.1). Si plusieurs *dialer map* sont présents sur l'interface, l'adresse du premier sera attribuée au client. Par contre, si le client est configuré avec une adresse statique, celle-ci doit être identique à celle du *dialer map* (10.3.0.1), sinon une erreur apparaît et le routeur rejettera l'adresse du client.

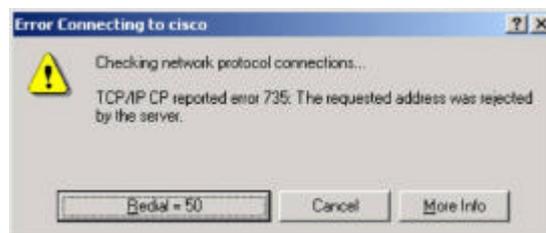


FIGURE 3.32 : Erreur de connexion

Si aucune commande *dialer map* n'est présente dans la configuration de l'interface, n'importe quelle adresse IP peut être utilisée pour le client. Cette adresse sera automatiquement ajoutée dans la table de routage du Cisco2600, comme nous l'avons vu sur la FIGURE 3.31.

3.3.5. Remarques sur « idle-timeout »

Le temporisateur *idle-timeout* a pour fonction de déconnecter la ligne *ISDN* lorsqu'il n'y a plus de « trafic intéressant » depuis un temps défini. Par « trafic intéressant », Cisco parle du trafic spécifié dans la commande *dialer-list* (ex. *dialer-list 1 protocol ip permit*). Tant qu'il y a du trafic de type IP, la connexion est maintenue. Si cette commande n'est pas présente dans la configuration du routeur, il est nécessaire de désactiver le temporisateur, sinon la liaison tombera une fois le temps écoulé, même s'il y a encore du trafic. Pour le désactiver, tapez :

➤ Cisco2600(config-if)# **dialer idle-timeout 0**

3.4. Configuration « unnumbered »

Afin de ne pas gaspiller des plages d'adresses, il est intéressant de donner au client une adresse se trouvant dans la plage du réseau distant (FIGURE 3.33). Bien sûr, le routeur ne permet pas qu'on lui attribue des adresses du même sous-réseau sur deux interfaces. Cela est contre la logique des routeurs. La commande *unnumbered* permet d'utiliser une interface du routeur sans adresse IP en l'associant avec, par exemple, une interface Ethernet. L'interface *unnumbered* devient transparente et est vue comme l'interface Ethernet.

ip unnumbered interface

- *interface* : Interface avec laquelle *unnumbered* est associée.

Pour réaliser cette configuration, il suffit juste de taper dans l'interface *BRI1/0* la commande :

```
➤ Cisco2600(config-if)# ip unnumbered FastEthernet0/0
```

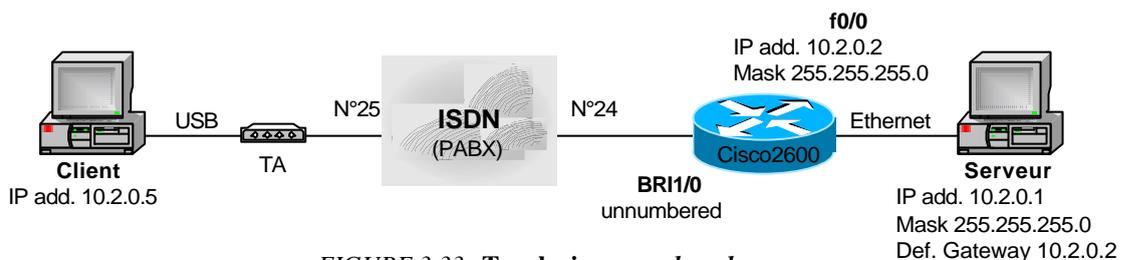


FIGURE 3.33 : Topologie *unnumbered*

La figure ci-dessous montre les routes contenues dans le routeur pour cette configuration. On constate qu'il n'y a plus de sous-réseau connecté au *BRI1/0*. Seul le client y est connecté.

```
Cisco2600 - HyperTerminal
File Edit View Call Transfer Help
Cisco2600#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
* - candidate default, U - per-user static route, o - ODR
E - periodic downloaded static route

Gateway of last resort is not set

  10.0.0.0/8 is variably subnetted, 2 subnets, 2 masks
C    10.2.0.0/24 is directly connected, FastEthernet0/0
C    10.2.0.5/32 is directly connected, BRI1/0
Cisco2600#_
```

FIGURE 3.34 : Routes de Cisco2600

3.5. Configuration dynamique de l'adresse IP du client

3.5.1. Principe

Quand on parle de configuration dynamique d'un client, notre première pensée est le protocole *DHCP* (*Dynamique Host Configuration Protocol*). Il est très simple de mettre en place un serveur *DHCP* sur les routeurs Cisco. Malheureusement, ce protocole fonctionne uniquement sur un réseau Ethernet. Il n'est pas utilisable sur *PPP*. Mais, comme nous l'avons vu plus haut (§ 3.2.4), *PPP* possède le protocole *IPCP* (*Internet Protocol Configuration Protocol*) qui permet la configuration de la couche IP. Pour cela, il suffit de paramétrer le routeur avec une plage d'adresse qu'il pourra attribuer au client. De plus, un serveur *DNS* doit lui être attribué. Comme nous l'avons dit plus haut (§ 3.3.2), pour une connexion *PPP*, le client ne possède plus de *Default Gateway* et de *Subnet mask*. Ceux-ci sont gérés par le réseau distant, dans notre cas le routeur. Voici comment le configurer.

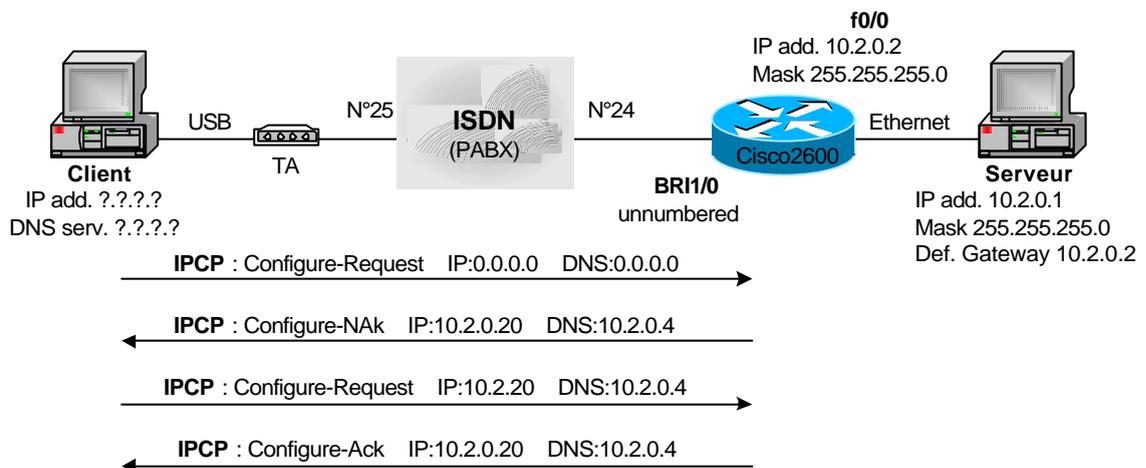


FIGURE 3.35 : Attribution dynamique d'adresse IP

Remarque : Cet échange est décrit au paragraphe 3.6.4.

3.5.2. Configuration du routeur

Le routeur doit posséder une plage d'adresses (*address-pool*) à attribuer. Pour cela, il faut taper :

```
> Cisco2600(config)# ip address-pool local
```

Puis il faut lui indiquer la plage à utiliser :

```
ip local pool name first-address last-address
```

- *Name* : Nom permettant d'associer le *pool* avec une interface.
- *first-address* : Première adresse de la plage.
- *last-address* : Dernière adresse de la plage.

```
➤ Cisco2600(config)# ip local pool dialin_pool 10.2.0.20
10.2.0.55
```

La commande permettant d'indiquer au client le serveur *DNS* à utiliser est :

ip name-server *DNS-server-address*

```
➤ Cisco2600(config)# ip name-server 10.2.0.4
```

Pour créer un *default gateway*, il suffit de spécifier une *ip route* indiquant l'adresse du chemin à utiliser pour atteindre d'autres réseaux. Pour l'instant, l'installation n'est pas encore connectée à d'autres réseaux. Cette commande prendra tous son sens dans la prochaine configuration (§ 3.6).

```
➤ Cisco2600(config)# ip route 0.0.0.0 0.0.0.0 10.2.0.3
```

Cette route est utilisée quand le routeur ne connaît pas d'autres chemins pour atteindre la destination. En faisant un *show ip route* (FIGURE 3.36) on reconnaît la route par défaut par le signe « S* ».

```
Cisco2600 - HyperTerminal
File Edit View Call Transfer Help
Cisco2600#sh ip route
Codes: C - connected, S - static, I - IGMP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is 10.2.0.3 to network 0.0.0.0

   10.0.0.0/8 is variably subnetted, 2 subnets, 2 masks
C       10.2.0.0/24 is directly connected, FastEthernet0/0
C       10.2.0.20/32 is directly connected, BR11/0
S*     0.0.0.0/0 [1/0] via 10.2.0.3
Cisco2600#_
Connected 0:33:43  Auto detect  9600 8-N-1  SCROLL  CAPS  NUM  Capture  Print echo
```

FIGURE 3.36 : Show ip route

Enfin, il faut associer le *pool* d'adresse à l'interface connecté au client pour lui indiquer les adresses qu'il peut attribuer.

peer default ip address pool *name*

```
➤ Cisco2600(config-if)# peer default ip address pool
dialin_pool
```

La configuration complète du routeur se trouve dans l'annexe *Configuration Cisco2600 pour attribution d'adresses dynamiques*

3.5.3. Configuration du PC client

A présent il faut configurer le PC pour qu'il demande une adresse IP, lors de l'établissement de la liaison. Dans l'onglet **Networking** des propriétés de connexion,

cliquez sur **Internet Protocol (TCP/IP)**, puis sur **Properties**. La fenêtre suivante s'ouvre.

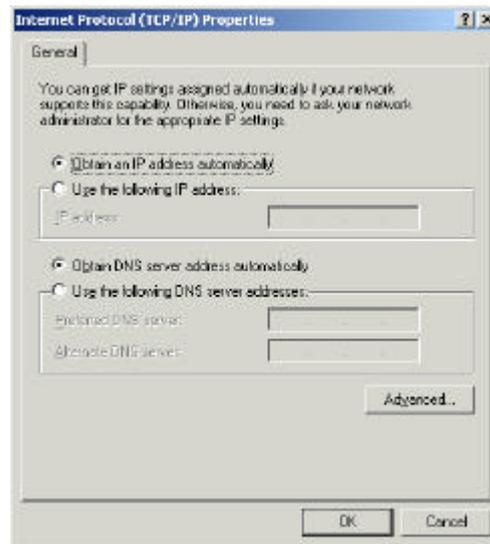


FIGURE 3.37 : Internet Protocol (TCP/IP) Properties

Sélectionnez les options **Obtain an IP address automatically** et **Obtain DNS server address automatically**. Refermez les fenêtres en cliquant sur **OK**.

Lorsque la connexion est établie, on peut vérifier les adresses reçues dynamiquement par le client. Pour cela, il suffit de taper `ipconfig/all` dans la fenêtre de commande de Windows 2000.

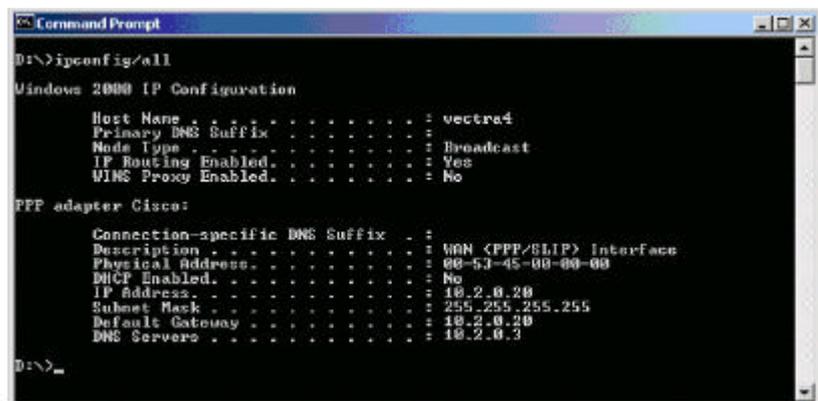


FIGURE 3.38 : Adresses reçues par le client

On constate que l'adresse IP et l'adresse du serveur *DNS* correspondent bien à celles spécifiées dans le routeur. On remarque aussi que l'adresse du *Default Gateway* est identique à l'adresse IP. Cela n'a aucune importance, car le *Default Gateway* est géré par le routeur.

3.6. Configuration pour accès distant au réseau du laboratoire

3.6.1. Topologie

Cette configuration est identique à la précédente, sauf que le routeur est connecté au réseau du labo. A présent, les serveurs se trouvent dans le réseau du labo ou dans un autre réseau (*WAN*). Cette topologie est plus réaliste que la précédente. Par contre, l'emploi d'adresses privées n'est plus possible, car elles sont non-routables. Le *pool* d'adresses du routeur sera constitué d'adresse du labo.

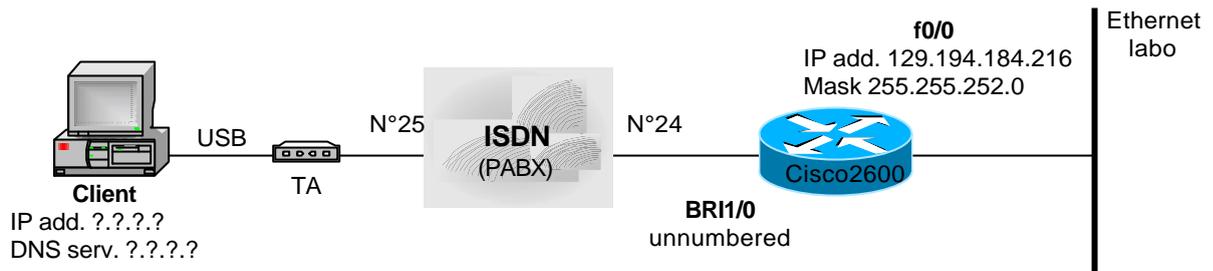


FIGURE 3.39 : Topologie du réseau

3.6.2. Configuration du routeur pour un client

Pour adapter la configuration précédente du routeur (§ 3.5) à cette topologie, il suffit de remplacer les adresses privées par des adresses publiques appartenant au labo. Voici les adresses IP attribuées.

- Interface *fastEthernet0/0* : IP 129.194.184.216 Mask 255.255.252.0
- *Address-pool* : De 129.194.186.214 à 129.194.186.217
- Serveur *DNS* : IP 129.194.4.6
- *Default Gateway* : IP 129.194.184.3

Voici les commandes utilisées :

- ```

➤ Cisco2600(config-if)# ip address 129.194.184.216
 255.255.252.0
➤ Cisco2600(config)# ip local pool dialin_pool
 129.194.186.214 129.194.186.217
➤ Cisco2600(config)# ip name-server 129.194.4.6
➤ Cisco2600(config)# ip route 0.0.0.0 0.0.0.0 129.194.184.3

```

Dans les configurations précédentes, nous avons oublié d'activer le *multilink* sur l'interface *BRI*. Cela a pour effet d'empêcher le client d'utiliser les 2 canaux B de la ligne *ISDN*. En effet, les paquets de niveau 3 peuvent être fragmentés si leur taille dépasse le *MRU* (*Maximum Receive Unit*) autorisé par *PPP* (FIGURE 3.40) ou, si ce n'est pas le cas, ils sont répartis sur les deux canaux B afin d'augmenter le débit binaire de la liaison *PPP*. Pour régler ce problème, il suffit d'ajouter la commande suivante dans la configuration du *BRI* :

- ```

➤ Cisco2600(config-if)# ppp multilink
  
```

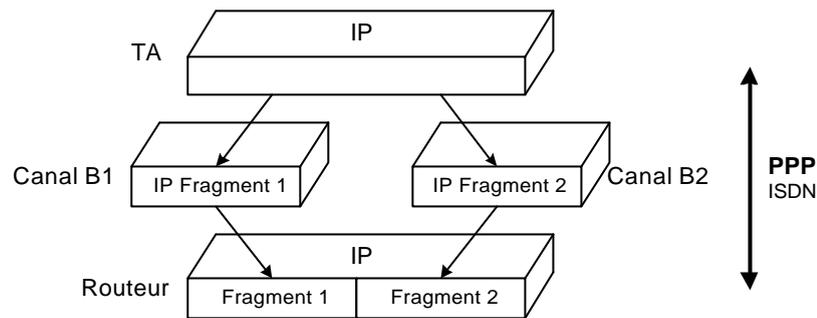


FIGURE 3.40 : Fragmentation des paquets IP pour le *multilink*

La configuration complète de routeur se trouve dans l'annexe *Configuration Cisco2600 pour accès au réseau du labo*.

Le PC ne nécessite pas de changement de configuration car ces adresses sont attribuées de manière dynamique.

Lorsque la connexion est établie, on peut vérifier les adresses reçues dynamiquement par le client. Pour cela, il suffit de taper *ipconfig/all* dans la fenêtre de commande de Windows 2000.

```

Command Prompt
D:\>ipconfig/all
Windows 2000 IP Configuration
Host Name . . . . . : vectra4
Primary DNS Suffix . . . . . :
Node Type . . . . . : Broadcast
IP Routing Enabled. . . . . : Yes
WINS Proxy Enabled. . . . . : No

PPP adapter Cisco:
Connection-specific DNS Suffix . . :
Description . . . . . : WAN (PPP/SLIP) Interface
Physical Address . . . . . : 00-53-45-00-00-00
DHCP Enabled. . . . . : No
IP Address . . . . . : 129.194.186.214
Subnet Mask . . . . . : 255.255.255.255
Default Gateway . . . . . : 129.194.186.214
DNS Servers . . . . . : 129.194.4.6
D:\>_

```

FIGURE 3.41 : Adresses reçues par le client

Pour vérifier le fonctionnement du *default gateway* et du *DNS*, on fait un *ping* sur le site *www.cern.ch*.

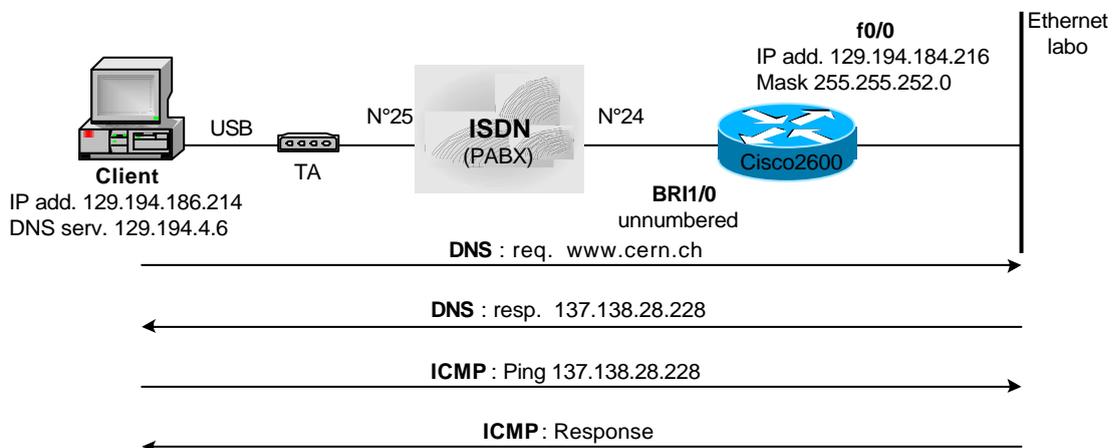


FIGURE 3.42 : Adresses reçues par le client

3.6.3. Configuration du routeur pour l'accès de plusieurs clients

Pour permettre à plusieurs clients de se connecter sur le réseau du labo en utilisant le RAS, il faut augmenter le nombre de connexion entre le routeur et l'ISDN.

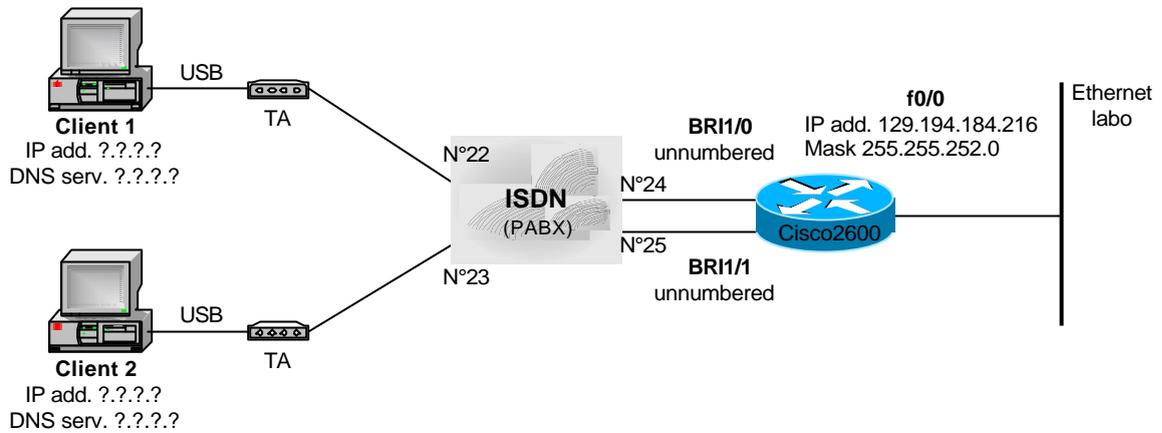


FIGURE 3.43 : Topologie pour plusieurs connexions RAS

Pour configurer le routeur comme sur la figure ci-dessus, il suffit de copier la configuration du *BRI1/0* sur le *BRI1/1*. Ces deux interfaces se partageront les adresses disponibles dans le *pool*.

Il est aussi possible de rajouter une liste de *username* et de *password* afin que chaque client en possède un différent. Par exemple :

```
➤ Cisco2600(config-if)# username pc1 password 0 12345
```

Cette configuration se trouve dans l'annexe *Configuration Cisco2600 pour accès au réseau du labo par plusieurs clients*

Voici encore quelques constatations faites pendant les tests de connexions des clients :

- Si le client 1 se connecte à l'interface *BRI1/0* en n'utilisant qu'un seul canal B, le client 2 peut aussi se connecter sur cette même interface.
- Si le client 1 se connecte à l'interface *BRI1/0* en utilisant les deux canaux B, le client 2 ne pourra plus se connecter à cette interface, car la ligne ISDN est occupée (FIGURE 3.44). Il faudrait configurer le central *ISDN* pour qu'il renvoie les appels sur les interfaces libres du routeur (numéro en PBX).



FIGURE 3.44 : The phone line is busy

- Le routeur associe le *username* et l'adresse IP qu'il attribue à un client. Donc, un même client reçoit toujours la même adresse IP. Mais, si toutes les adresses IP du *pool* sont associées, le routeur attribue l'adresse n'ayant plus servi depuis le plus longtemps.

3.6.4. Capture d'une phase d'établissement avec le débogueur du routeur Cisco

La figure ci dessous représente l'établissement d'une liaison *PPP* pour un canal B. Si l'option *multilink* est utilisée, l'établissement du deuxième canal B réeffectuera les échanges *LCP* et *CHAP*. Le débogueur du routeur Cisco permet de voir ces différentes étapes d'établissement.

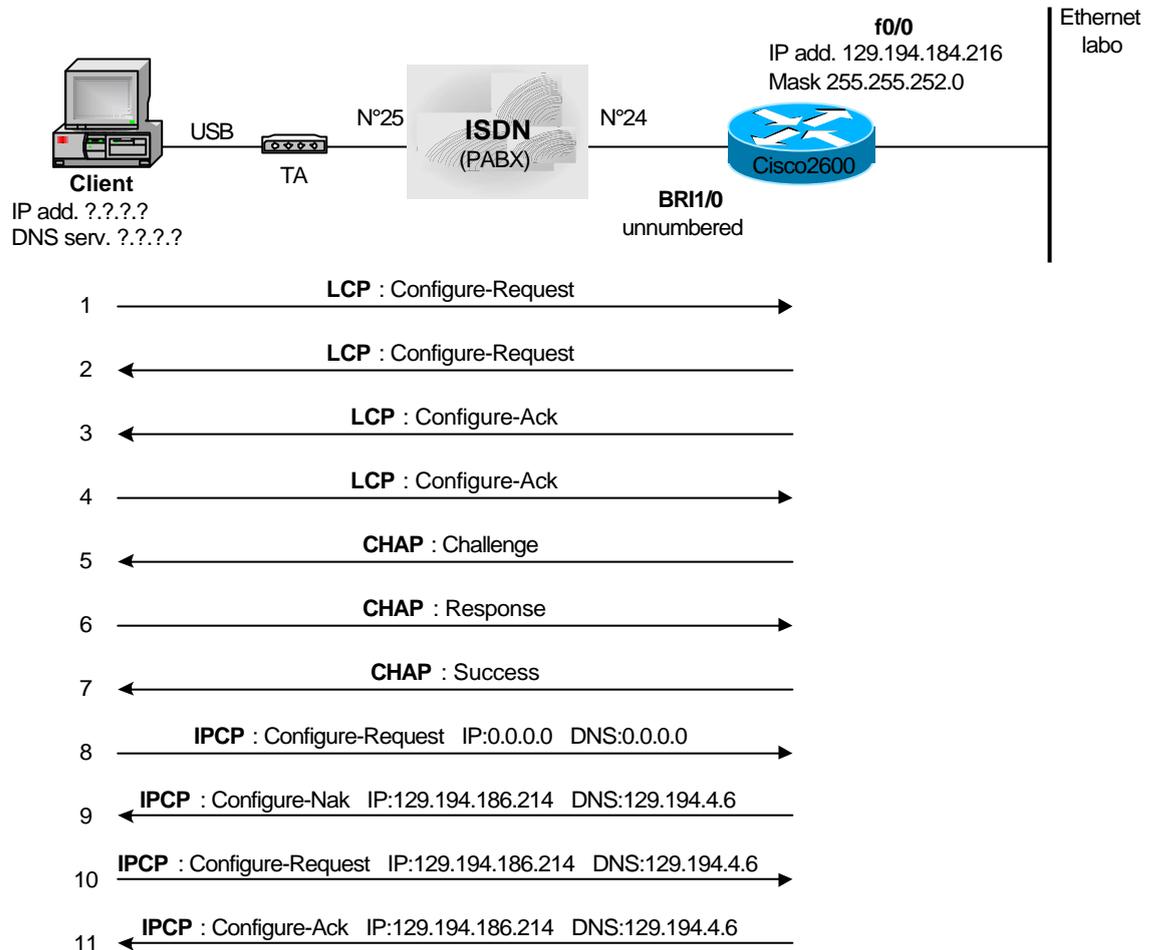


FIGURE 3.45 : Etablissement de la connexion PPP

La capture complète de l'établissement se trouve dans l'annexe *Capture de l'établissement entre un PC client utilisant la FRITZ!CARD USB et le routeur Cisco2600*. Pour activer les débogueurs utilisés pour cette capture, tapez les commandes suivantes :

- Cisco2600# **debug ppp authentication**
- Cisco2600# **debug ppp negotiation**
- Cisco2600# **debug isdn q931**

Lors de l'établissement de la connexion PPP, on retrouve les trois phases décrites dans le chapitre 3.2.4.

La première est la configuration des paramètres de connexion (*LCP*). Les deux équipements commencent par envoyer un paquet *Configure-Request* contenant les

options de connexion désirées par chaque interface (paquets 1 & 2). Par exemple, ces options peuvent être la taille maximale des paquets (*MRRU*), le type de protocole d'authentification (*CHAP* ou *PAP*) ou encore l'utilisation du *multilink*. Ces configurations sont acceptées par l'envoi de paquets d'acquittement (paquets 3 & 4). Cette phase *LCP* est effectuée pour chaque canal B activé.

La deuxième phase est l'authentification. On voit bien les trois paquets (5, 6 et 7) utilisés pour le protocole *CHAP*. L'authentification est effectuée pour chaque canal B.

La dernière phase est la configuration de la couche supérieure avec le protocole *IPCP*. Le client propose, au routeur, l'adresse IP et le serveur *DNS* qu'il désire utiliser. Dans notre cas (paquet 8), les adresses sont nulles, car le client n'en possède pas. Le routeur refuse ces adresses et lui envoie des adresses valides (paquet 9). Le client effectue une nouvelle demande avec les valeurs reçues (paquet 10). Le routeur valide cette configuration avec un acquittement (paquet 11). Cette phase de configuration n'est effectuée qu'une fois par connexion du client.

3.7. Conclusion

La configuration du routeur comme serveur d'accès à un réseau distant a pu être effectuée. Les difficultés ne se sont pas trouvées aux endroits attendus. En effet, je pensais que la phase d'authentification poserait beaucoup de problèmes à cause de la compatibilité entre Windows 2000 et les routeurs Cisco. Il n'en était rien. Le plus grand problème rencontré a été la configuration dynamique des adresses IP du client. Cela a été dû à une recherche dans une mauvaise direction. En effet, la première piste suivie a été d'utiliser le protocole *DHCP*, alors qu'il n'est pas utilisable sur *PPP*. Mais finalement ce problème a été réglé avec le protocole *IPCP*.

Ce travail m'a aussi permis de comprendre et de m'exercer à la configuration des routeurs Cisco. Cela est très important pour la suite du diplôme.

4. Introduction au protocole *MPLS*

4.1. *MPLS*, c'est quoi ?

L'acronyme *MPLS* désigne *Multiprotocol Label Switching*. Ce protocole est normalisé (RFC 3031) par l'*IETF* (*Internet Engineering Task Force*). Il assure les fonctions suivantes :

- Il offre une méthode d'encapsulation des paquets.
- Il spécifie des mécanismes pour administrer les flux de trafic de plusieurs types, comme les flux entre des matériels différents ou des applications différentes.
- Il est indépendant des protocoles des couches 2 et 3 (*Multiprotocol*).
- Il interagit avec des protocoles de routage, existant comme *RSVP* (*Resource Reservation Protocol*) ou *OSPF* (*Open Shortest Path First*).
- Il utilise la méthode du *label switching* pour *forwarder* les paquets.

Remarque : Le protocole *MPLS* peut être utilisé sur des réseaux IP, *ATM* et *Frame Relay*. Tous les scénarios étudiés lors de ce travail de diplôme sont appliqués à un réseau IP. Ce document ne traite donc pas d'*ATM* et de *Frame Relay*.

4.2. Concept

Au cours des dernières années, de nouvelles applications se sont développées sur Internet. Ces applications ont des besoins grandissants en terme de bande passante et de sécurité. Sur les réseaux IP traditionnels, l'analyse de l'adresse IP de destination des paquets est effectuée dans chaque routeur se trouvant entre la source et la destination du paquet (*hop-by-hop routing*). Cette analyse répétée est responsable d'une grande partie du temps de propagation des paquets à travers le réseau. De plus, les protocoles de *routing* actuels ont tendance à surexploiter certaines routes (ex. les routes les plus courtes entre deux points) alors que d'autres ne sont pratiquement pas utilisées.

Le protocole *MPLS* offre des solutions à ces problèmes. Le principe utilisé est l'encapsulation des paquets à acheminer à travers le réseau, à l'aide de labels. Les labels ont une signification locale. Ils permettent directement de déterminer un chemin virtuel vers une destination. Les routeurs utilisent uniquement ces labels pour *switcher* les paquets tout au long de ce chemin. Comme nous verrons plus loin, cette opération est plus simple et plus performante que le routage IP traditionnel. *MPLS* permet d'effectuer du *traffic engineering*. Par exemple, il est possible de forcer certains paquets à employer une route prédéfinie afin de désengorger des routes saturées. Il permet aussi la création de *VPN* (*Virtual Private Network*).

4.3. Elément du réseau MPLS

Le protocole *MPLS* définit de nouveaux éléments qui constitue son réseau. **Il est important de bien les connaître pour comprendre la suite de ce document.** La FIGURE 4.1 illustre certains de ces éléments.

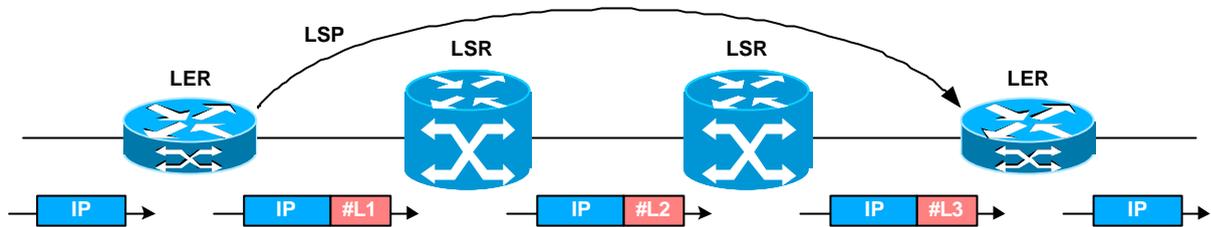


FIGURE 4.1 : Eléments d'un réseau MPLS

LSP : Label-Switched Paths

Dans *MPLS*, les données transmises circulent sur des *LSP*. Les *LSP* sont des chemins, allant de la source à la destination, constitués d'une séquence de labels entre chaque nœud traversé. Par exemple, sur la FIGURE 4.1, le *LSP* est constitué de la séquence de labels L1, L2 et L3. Un *LSP* est unidirectionnel. Le trafic de retour doit donc emprunter un autre *LSP*.

FEC : Forward Equivalence Class

Un *FEC* est la représentation d'un groupe de paquets qui ont les mêmes besoins quant à leur transport. Les paquets d'un tel groupe reçoivent le même traitement au cours de leur acheminement. Les *FEC* sont basées sur les besoins en terme de service ou sur un préfixe d'adresse IP.

LER : Label Edge Routers

Un *LER* (ou *Edge-LSR*) est un routeur, supportant les labels, se trouvant à l'extrémité du réseau *MPLS*. Il possède une ou plusieurs interfaces connectées au réseau d'accès (par exemple un réseau IP). C'est lui qui assigne et supprime les labels au fur et à mesure que le trafic entre et sort du réseau *MPLS*.

LSR : Label Switching Router

Un *LSR* est un routeur à haut débit au cœur du réseau *MPLS*. Il participe à l'établissement des *LSP*. Son rôle est de *forwarder* les paquets *MPLS* vers leur destination en utilisant la méthode du *label switching*.

LIB : Label Information Base

Un *LIB* est une table contenant tous les labels assignés par un *LSR* (ou un *LER*) et le *mapping* de ces labels avec les labels reçus par les *LSR* voisins. Le protocole *LDP* (*Label Distribution Protocol*) diffuse le contenu de cette table aux autres *LSR*.

LFIB : Label Forwarding Information Base

Un *LFIB* est une table contenant les informations utilisées, par un *LSR* ou un *LER*, pour *forwarder* les paquets. Cette table est construite à partir du *LIB*.

Le routage des paquets *MPLS* sur le réseau peut être divisé en deux parties :

- *Forwarding* : Cette opération est effectuée par un *LSR* afin de déterminer où doivent aller les paquets reçus et les envoyer sur la bonne interface. Le *LSR* détermine cela grâce au label des paquets et à une table de *forwarding*.
- *Control* : La partie contrôle consiste à distribuer les informations de routage entre les *LSR* afin de constituer les tables de *forwarding*.

4.4. Encapsulation des paquets

Le protocole *MPLS* définit un label permettant d'encapsuler les paquets à transporter. Ce label est appelé *shim header*. Il est inséré entre l'entête de la couche deux et l'entête de la couche trois. La figure ci-dessous montre l'encapsulation d'un paquet IP.



FIGURE 4.2 : Encapsulation d'un paquet IP avec un label *MPLS*

Comme le montre la figure ci-dessous, le *shim header* est constitué de 32 bits.

Label	Exp	Stack	TTL
20	3	1	8

FIGURE 4.3 : Format du label *MPLS* ou *Shim header*

- *Label* : Ce champ est de 20 bits. Il contient la valeur du label. C'est cette valeur qui est utilisée pour le *label switching*.
- *Exp* ou *Experimental* : Ce champ est de 3 bits. Il contient des informations sur la *class-of-service* du paquet.
- *Stack* : Ce champ est de 1 bit. Il indique le dernier label d'une pile de label (*label stack*) quand il est à 1.
- *TTL (Time To Live)* : Ce champ est de 8 bits. Il prend la valeur du *TTL* de l'entête *IP* lors de l'encapsulation du paquet

Le bit de *stack* permet l'utilisation d'une pile de label (*label stack*). Une pile de label est la combinaison de deux ou plusieurs labels pour encapsuler un paquet. Un simple routage de paquet n'utilise pas de pile de label. Cette pile est surtout utilisée pour le *Traffic Engineering* ou la création de *VPN (Virtual Private Network)*.

Les paquets entrant ou sortant d'un réseau *MPLS* sont respectivement encapsulés ou décapsulés par un *LER*. A l'entrée du réseau *MPLS*, le *LER* assigne les paquets à un *FEC* et leur ajoute le label correspondant (*push*). En effet, les labels sont associés aux *FEC* et chaque label correspond à un *LSP*. Le paragraphe 4.7.3 explique comment les *LSP* sont créés. A l'extrémité du *LSP* (sortie du réseau *MPLS*), le *LER* retire le label des paquets (*pop*) et utilise l'entête IP afin de les *forwarder* vers leur destination.

4.5. Forwarding des paquets

Comme nous l'avons dit précédemment, les paquets circulant sur un réseau *MPLS* possèdent un label situé entre l'entête de la couche 2 et l'entête de la couche 3. Ce label est utilisé par les *LSR* qui reçoivent un paquet, sur une de leurs entrées, pour déterminer sa destination et l'envoyer sur la sortie appropriée. Cette opération correspond au *forwarding* des paquets. Dans *MPLS*, la méthode de *forwarding* utilisée est le *label switching*. Voici comment cela fonctionne :

Le *LSR* possède une table de *forwarding* appelée *Label Forwarding Information Base (LFIB)*. Cette table est divisée en deux parties, *incoming label* et *subentry*. La partie *incoming label* contient les numéros des labels correspondant aux différents *LSP* traversant le *LSR*. La partie *subentry* contient les informations permettant de *forwarder* les paquets sur ces *LSP*. Chaque *subentry* est associée à un *incoming label*. Lorsque le *LSR* reçoit un paquet sur l'une de ses interfaces, il extrait le label de celui-ci. Le numéro de ce label est recherché dans les *incoming label* du tableau. La *subentry* correspondante contient le numéro du nouveau label à insérer dans le paquet, l'interface par laquelle ce paquet doit être envoyé et l'adresse du prochain *LSR*. Une fois le label ajouté, le paquet est envoyé au prochain *LSR*. Dans les *LER*, la *subentry* indique simplement au routeur qu'il doit retirer le label du paquet.

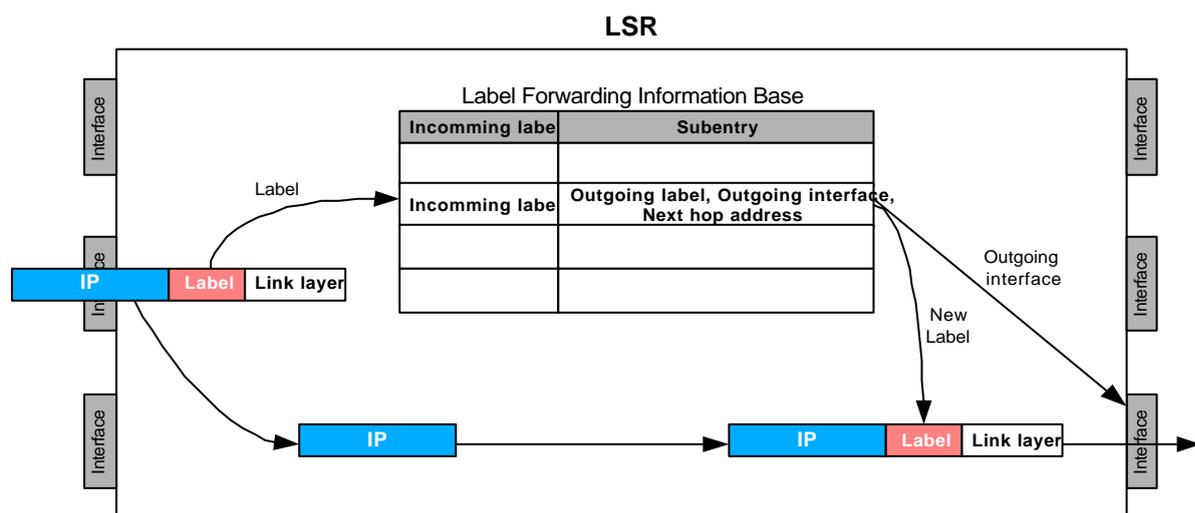


FIGURE 4.4 : Forwarding d'un paquet dans un *LSR*

Les *LSR* remplacent donc le label des paquets reçus avant de les *forwarder*. Ce mécanisme se nomme *label swapping*. Le *switching* des paquets est simple et très performant, car, du point de vue matériel, cela correspond à un accès mémoire.

4.6. Routage des paquets

Pour que les *LSR* puissent *forwarder* les paquets vers leur destination, ils doivent posséder une table de routage leur permettant de connaître la topologie du réseau. Cette table est construite de façon classique par un protocole de type *IGP* (*Interior Gateway Protocol*). Par exemple, le protocole utilisé peut être *IS-IS* (*Intermediate System to Intermediate System routing protocol*), *RIP* (*Routing Information Protocol*), *OSPF* (*Open Shortest Path First*) ou *BGP* (*Border Gateway Protocol*). Certains de ces protocoles sont utilisés pour les applications qui vont suivre. Le but de ce travail n'est pas de les étudier, c'est pourquoi seule leur utilisation sera expliquée.

Chaque *LSR* (ou *LER*) construit son *LIB* (*Label Information Base*) en associant un label à chaque préfixe de la table de routage. Les préfixes correspondent aux destinations connues par les routeurs. Dans ce cas, on peut dire que chaque préfixe représente un *FEC*. Le *LIB* contient donc les associations (*bindings*) entre les *FEC* et les labels. Le protocole *LDP* (*Label Distribution Protocol*, voir § 4.7) exploite ces *LIB* pour déterminer et échanger des valeurs de label qui seront utilisées pour *switcher* les paquets, caractéristiques d'un *FEC*, le long des routes (*LSP*) ainsi établies.

Le *LIB* associe donc un label pour chaque préfixe de la table de routage. De cette manière, les *LSR* connaissent le label à utiliser pour atteindre toutes les destinations sur le réseau. La table de *forwarding* (*LFIB*) est créée à partir du *LIB* et ne contient que les labels actuellement utilisés pour le *forwarding* des paquets par le *LSR*. Si un nouveau chemin (*LSP*) est créé, le *LSR* possède donc toutes les informations nécessaires, dans le *LIB*, pour mettre à jour sa table de *forwarding* (*LFIB*).

4.7. Distribution des labels avec le protocole *LDP*

4.7.1. Introduction

Le protocole *LDP* (*Label Distribution Protocol*) définit un ensemble de procédures et de messages permettant à un *LSR* d'informer ses voisins sur les labels à utiliser pour *forwarder* les paquets sur le réseau. Ces messages permettent, par exemple, de transmettre l'association (*mapping*) entre les labels et les *FEC* afin d'établir les chemins (*LSP*) à travers le réseau. Les principaux mécanismes de ce protocole sont décrits ci-dessous. Toutefois, le lecteur peut se référer à la RFC 3036 pour plus de détail.

4.7.2. Types de messages *LDP*

Deux *LSR* utilisant *LDP* pour se transmettre des informations, sur l'association entre labels et *FEC*, sont appelés *LDP Peers*. Afin de pouvoir dialoguer, ils établissent une session *LDP*. Cette session est bidirectionnelle. Le protocole définit quatre catégories de message permettant ce dialogue :

- Les messages *Discovery* sont utilisés pour annoncer et maintenir la présence d'un *LSR* sur un réseau.

- Les messages **Session** sont utilisés pour établir, maintenir et terminer une session entre deux *LDP Peers*.
- Les messages **Advertisement** sont utilisés pour créer, échanger et supprimer des *mapping* de labels et de *FEC*.
- Les messages **Notification** sont utilisés pour transmettre des messages d'information et signaler des erreurs.

Les messages *Discovery* sont utilisés sur *UDP*. Périodiquement, les *LSR* envoient des paquets **Hello**, en multicasts, aux autres *LSR* du sous-réseau. Ce mécanisme permet à chaque *LSR* d'apprendre qui sont ses voisins. Lorsqu'un *LSR* apprend l'adresse IP d'un de ses voisins, il établit une connexion *TCP* permettant la mise en place de la session *LDP*.

Il existe plusieurs messages utilisés lors d'une session *LDP*. Voici les plus communément utilisés :

- **Initialisation** : Ce message est envoyé pour commencer une session *LDP*. Il permet de négocier certains paramètres de la session comme par exemple la plage de labels à utiliser entre ses deux *LSR* ou les *timers* pour les messages *Keepalive*. Si les paramètres sont acceptables, le *Peer* répond par un message *Keepalive*. Sinon, un message de notification d'erreur est envoyé et l'initialisation est abandonnée.
- **Keepalive** : Ces messages sont périodiquement envoyés par un *LSR* afin de prévenir les *LSR Peers* de leur bon fonctionnement. L'absence de ces messages, après un temps déterminé, est considérée, par un *LSR*, comme la disparition de son *Peer* (changement de topologie ou panne). Cela conduit à la terminaison de la session *LDP* et au retrait des labels, associés aux *Peers*, du *LIB*.
- **Label mapping** : C'est ce message qui est utilisé pour la distribution des labels. Il permet d'envoyer, aux autres *LSR*, les liens entre les *FEC* et les labels.
- **Label withdrawal** : Ce message permet d'effectuer l'opération inverse que le message *label mapping*. Il permet de révoquer des associations de *FEC* et de labels.
- **Label release** : Ce message est utilisé par un *LSR* qui a reçu un *label mapping* dont il n'a pas d'utilité. Par exemple, cela est le cas d'un *LSR* utilisant le mode *conservative* (voir ci-dessous) et dont le *LSR* ayant envoyé le *label mapping* n'est pas le prochain routeur pour ce *FEC*.
- **Label request** : Ce message est utilisé par un *LSR* pour demander au *LSR Peer* de lui retourner un label associé à un *FEC*.
- **Label request abort** : Ce message permet de révoquer une demande de label.
- **Address** : Ce message est utilisé par un *LSR* pour indiquer, à un *LSR Peer*, les adresses des ses interfaces.

4.7.3. Modes de distribution des labels

La distribution des labels et leur assignement peut être effectuée selon un nombre de modes différents. Un de ces modes est *unsolicited downstream*, par opposition à *downstream-on-demand*. *Unsolicited downstream* est le mode le plus courant. Lorsqu'un *LSR* est mis en service, il construit sa table d'association entre *FEC* et labels (*LIB*) et informe ses *Peers* sur ces associations (message *label mapping*). Ainsi, lorsqu'un *Peer* doit envoyer un paquet, correspondant à un certain *FEC*, à travers le *LSR*, il connaît le label à employer. Au contraire, dans le mode *downstream-on-demand*, c'est au *LSR* désirant transmettre un paquet, pour un certain *FEC*, de demander à son *Peer* de lui indiquer le numéro du label à utiliser (message *label request*).

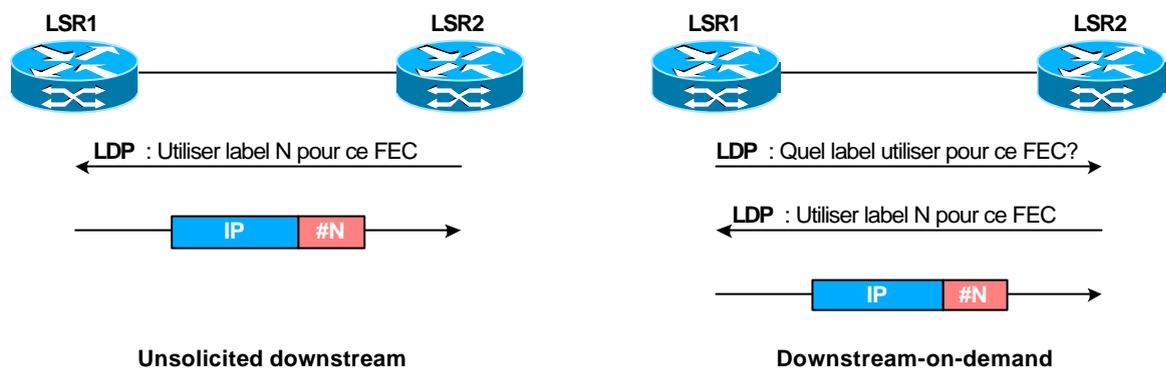


FIGURE 4.5 : *Unsolicited downstream* par opposition à *downstream-on-demand*

Une autre variante est la distribution ordonnée (*ordred*) des labels, pour constituer un *LSP*, en opposition à une distribution indépendante (*independant*). Dans le mode indépendant, chaque *LSR* prend la décision d'assigner un label à un *FEC*, de manière indépendante, et prévient ses *Peers* de cette association. De cette façon, les *LSP* sont directement créés par convergences des *FEC*. Au contraire, dans le mode ordonné, la distribution des labels se fait depuis les *LSR* se trouvant à l'extrémité du *LSP* à créer. En général, c'est le *LER* se trouvant à la sortie du *LSP* qui indique, aux *LSR* voisins, le label à utiliser pour atteindre le sous-réseau auquel il est connecté. L'information est ainsi propagée de routeurs à routeurs jusqu'à l'autre extrémité du *LSP*.

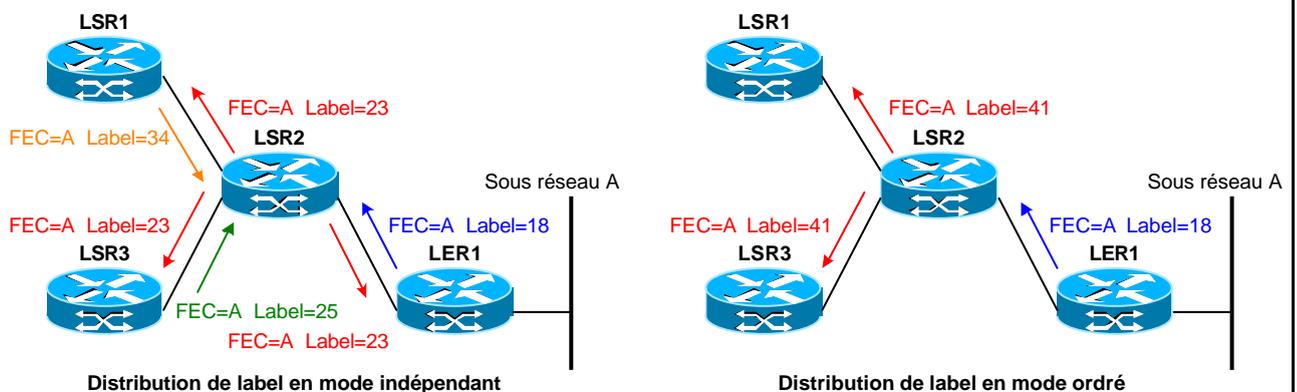


FIGURE 4.6 : Distribution de labels en mode indépendant ou ordonné

La rétention des labels peut se faire selon un mode libéral (*liberal*) ou conservatif (*conservative*). En mode conservatif, un *LSR* ne retient que les *mapping* des labels et

des *FEC* actuellement utilisés par le routeur pour *forwarder* les paquets. En mode libéral, le *LSR* retient tous les *mapping* reçus de ses *Peers*. Ce mode a comme avantage de permettre une réaction plus rapide lorsqu'une liaison est interrompue entre deux *LSR*. En effet, le *LSR* connaît déjà le label à employer pour envoyer un paquet vers un autre routeur. Par contre, le stockage de tous les *mappings* nécessite l'utilisation de beaucoup de mémoire dans les *LSR*. Cela n'est pas toujours possible lorsque le réseau devient trop important.

Le choix des modes utilisés par les *LSR* sont négociés durant la phase d'initialisation de la session *LDP*.

4.7.4. Exemple d'échange *LDP*

L'exemple illustré par la figure ci-dessous représente un échange *LDP* entre deux *LSR* utilisant la distribution de labels en mode indépendants et *unsolicited downstream*.

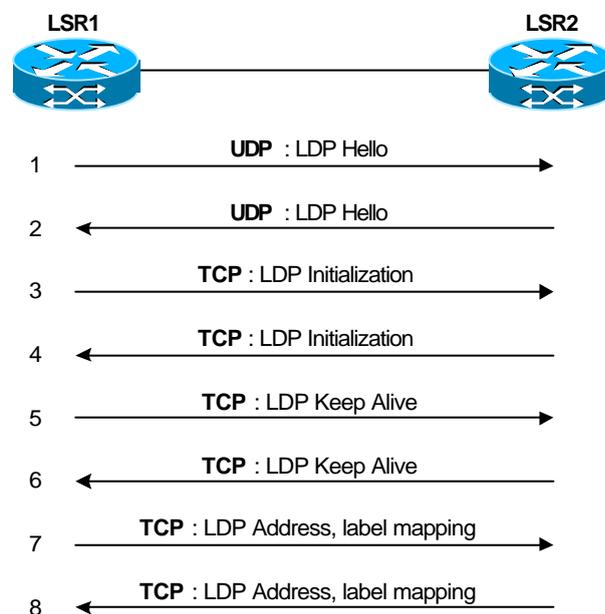


FIGURE 4.7 : Echange *LDP* entre deux *LSR*

Les *LSR* diffusent des paquets *Hello*, sur leur sous-réseau, en utilisant le protocole *UDP* en multicast (paquets 1 & 2), afin de se faire connaître par les autres *LSR*. Ce mécanisme permet à un *LSR* de découvrir l'adresse IP de son voisin (*LSR Peer*). Cette adresse connue, une connexion *TCP* peut être établie entre ces deux routeurs. L'emploi de *TCP* permet de garantir la bonne transmission des informations de la session *LDP*. Il faut noter que chaque paquet *TCP* envoyé est acquitté par le récepteur. Ces paquets ne sont pas représentés afin de ne pas surcharger la figure ci-dessus. Une fois la connexion *TCP* établie, la session *LDP* peut commencer. Pour cela, chaque *LSR* envoie un paquet d'initialisation (paquets 3 & 4). Ces paquets permettent de négocier certaines options de la session *LDP* comme, par exemple, la taille maximale des paquets (*Max PDU Length*), l'intervalle entre chaque message *Keep Alive* (*Keep Alive Time*) ou encore le mode de distribution des labels (*unsolicited downstream* ou *downstream-on-demand*). Si les options sont acceptées par le *LSP Peer*, celui-ci répond par un message *Keep Alive* (paquets 5 & 6). Si ce n'est pas le cas, un message de notification d'erreur est envoyé et la session *LDP* est

terminée. Une fois les paramètres d'initialisation acceptés, les *LSR* peuvent commencer à s'échanger les différents liens entre labels et *FEC* (*Label mapping*) (paquets 7 & 8). Les *LSR* transmettent aussi au *LSR Peer* les adresses IP de leurs interfaces (*Address*).

4.7.5. Format des paquets

Les échanges *LDP* sont effectués avec des paquets nommés *Protocol Data Unit* (*PDU*). Chaque *PDU* peut transporter un ou plusieurs messages *LDP*. Voici le format de ces paquets :

Version	PDU Length	LDP Identifier	LDP Messages
16	16	48	variable

FIGURE 4.8 : Format du paquet *LDP*

- *Version* : Ce champ est de 16 bits. Il indique la version du protocole *LDP* utilisée pour ce message.
- *PDU Length* : Ce champ est de 16 bits. Il indique la longueur totale du paquet *LDP*. Cette longueur n'inclut pas les champs *Version* et *PDU Length*.
- *LDP Identifier* : Ce champ est de 48 bits. Les 32 premiers bits identifient le *LSR* à l'origine de ce paquet. Cette valeur correspond à l'adresse IP de ce *LSR*. Les 16 derniers bits correspondent au *label space* du *LSR*. Cette valeur est toujours nulle, pour un réseau IP.
- *LDP Messages* : Ce champ est de taille variable. Il contient un ou plusieurs messages. Ces messages sont construits de la manière suivante :

U	Message Type	Message Length	Message ID	Mandatory Parameters	Optional Parameters
1	15	16	32	variable	variable

FIGURE 4.9 : Format des messages *LDP*

- *U* : Ce champ est de 1 bit. Si le message reçu est inconnu et que le bit est à 0, un message de notification est retourné à l'expéditeur. Si le bit est à 1, le message inconnu est simplement ignoré.
- *Message Type* : Ce champ est de 15 bits. Il identifie le type de message. Voici les principales valeurs qu'il peut prendre :

0x0001	Notification
0x0100	Hello
0x0200	Initialisation
0x0201	KeepAlive
0x0300	Address
0x0301	Address Withdraw
0x0400	Label Mapping

0x0401	Label Request
0x0402	Label Withdraw
0x0403	Label Release
0x0404	Label Abort Req

- *Message Length* : Ce champ est de 16 bits. Il représente la somme des octets composant les champs *Message ID*, *Mandatory Parameters* et *Optional Parameters*.
- *Message ID* : Ce champ est de 32 bits. Il contient une valeur devant être copiée si un message de notification est renvoyé à l'expéditeur. Cette valeur facilite, pour l'expéditeur, l'identification du message ayant provoqué une notification.
- *Mandatory Parameters* : Ce champ est de longueur variable. Il contient les paramètres requis pour le message défini par le champ *Message Type*. Ces paramètres sont définis dans un format *TLV* décrit par la FIGURE 4.10. Certains messages n'ont pas de paramètres requis.
- *Optional Parameters* : Ce champ est de longueur variable. Il contient des paramètres optionnels. Ces paramètres sont aussi définis dans un format *TLV*. La plupart des messages n'ont pas de paramètres optionnels.

Les paramètres des messages *LDP* utilisent un format *Type-Length-Value (TLV)*. La figure ci-dessous représente ce format :

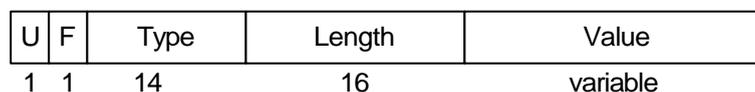


FIGURE 4.10 : Format d'un TLV

- *U* : Ce champ est de 1 bit. Si le message reçu est inconnu et que le bit est à 0, un message de notification est retourné à l'expéditeur. Si le bit est à 1, le message inconnu est simplement ignoré.
- *F* : Ce champ est d'un bit. Ce bit n'a de signification que si le bit *U* est à 1. Si *F* est à 1, cela indique que les *TLV* inconnus doivent quand même être *forwardés*. Si *F* est à 0, les *TLV* inconnus ne sont pas *forwardés*.
- *Type* : Indique comment le champ *Value* doit être interprété. Voici certaines valeurs qu'il peut prendre :

0x0100	FEC
0x0101	Address List
0x0200	Generic Label
0x0300	Status
0x0400	Common Hello Parameters
0x0500	Common Session Parameters

- *Length* : Ce champ est de 16 bits. Il définit la longueur, en octets, du champ *Value*.
- *Value* : Ce champ est de longueur variable. Sa longueur est déterminée par le champ *Length*. Il contient les informations correspondant au champ *Type*. Le format de ces informations prend des formes très diverses selon le type de *TLV*. Il serait beaucoup trop long de les décrire. C'est pourquoi le lecteur peut se référer à la RFC 3036 (*LDP Specification*) afin d'obtenir plus d'informations sur ces différents formats.

5. Etude du mécanisme de *forwarding* MPLS sur des machines Linux

5.1. Introduction

L'objectif des scénarios qui vont suivre, est d'analyser comment *MPLS* encapsule les paquets à l'aide de labels, afin de les *forwarder* sur le réseau. Ce réseau sera constitué de PCs Linux jouant le rôle des équipements *MPLS*. Pour permettre cela, le patch « *mpls-linux-0.993* » doit être appliqué sur ces PCs. Ce patch est toujours en développement. Il contient encore quelques bugs. On peut le trouver sur le site Internet <http://sourceforge.net/projects/mpls-linux/>. Ce site possède aussi un forum permettant d'obtenir beaucoup d'informations sur le fonctionnement du patch.

La distribution de Linux utilisée est la Mandrake 8.0. La version du *kernel* Linux utilisé est la 2.4.10. Lors de l'installation du patch sur les machines, de nombreux problèmes ont été rencontrés. Par exemple, nous avons eu des difficultés à recompiler le noyau Linux. Cela était dû à la mauvaise déclaration d'une variable dans un des fichiers sources. Certains fichiers de compilation ont dû être modifiés. Plusieurs jours ont été nécessaires pour régler ces problèmes. L'installation de ce patch et les modifications des fichiers sont expliqués dans l'annexe « Installation des patchs sur LINUX ».

La figure ci-dessous montre la topologie du réseau réalisé. Il est constitué de cinq PCs. Deux d'entre eux possèdent trois cartes Ethernet 10-100BaseT, un autre en possède deux et les deux derniers en possèdent une.

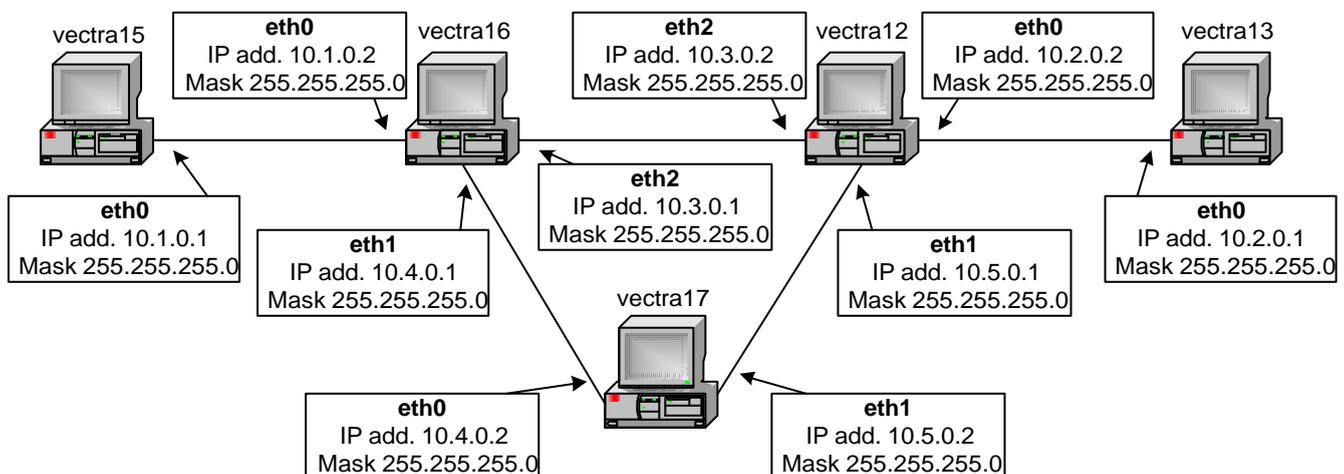


FIGURE 5.1 : Topologie du réseau

Le patch « *mpls-linux-0.993* » permet aux machines Linux d'encapsuler (insérer le label) et de *forwarder* les paquets reçus ou envoyés sur leurs interfaces. Par contre, les *LSP*, les *FEC* et les tables de *forwarding* sont créés manuellement (statiquement). Pour cela, ce patch met à disposition la commande ***mplsadm***. L'utilisation de cette commande sera expliquée dans les scénarios ci-dessous.

5.2. Configuration des interfaces des PCs

Pour configurer les interfaces Ethernet sur Mandrake 8.0, procédez de la manière suivante. Cliquez sur **Start Application** (bouton avec un «K» en bas à droite), pointez sur **Configuration**, puis sur **Networking** et cliquez sur **Netconf**. La fenêtre suivante s'ouvre :

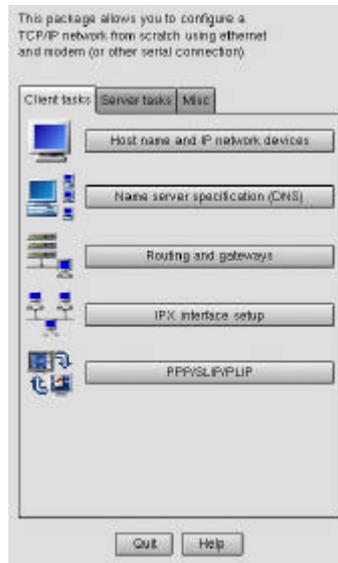


FIGURE 5.2 : Fenêtre Netconf

Dans l'onglet **Client tasks**, cliquez sur **Hostname and IP network devices**. La fenêtre ci-dessous apparaît :

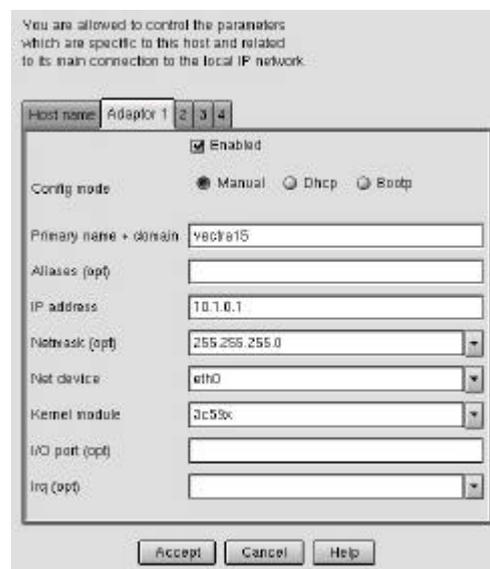


FIGURE 5.3 : Configuration des interfaces

Dans chaque onglet de cette fenêtre, une interface peut être configurée. Sélectionnez une interface dans le champ **Net device** et configurez les champs **IP address** et **Netmask** selon la figure 5.1. Cliquez sur **Accept**, puis sur **Quit**. Une fenêtre indiquant les modifications qui vont être effectuées s'ouvre. Pour valider ces changements, cliquer sur **Do it**.

5.3. Encapsulation des paquets avec les labels MPLS

5.3.1. Objectif

Ce premier scénario a pour but la création de deux chemins ou *LSP* (voir figure ci-dessous). Ces chemins sont associés à deux *FEC*. Le premier *FEC* permet d'atteindre la machine vectra13. Pour cela un *LSP* utilisant les labels 21, 22 et 23 est créé. Le deuxième *FEC* permet le retour vers le PC vectra15, car chaque *LSP* est unidirectionnel. Le *LSP* créé utilise les labels 41,42 et 43.

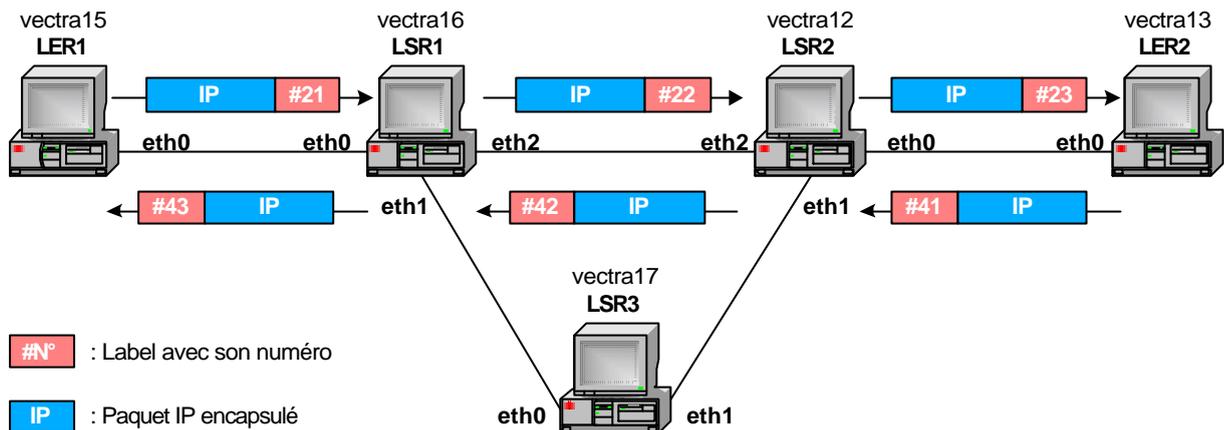


FIGURE 5.4 : Réseau avec 2 LSP

5.3.2. Configuration des LER

Vectra15 et Vectra13 jouent le rôle de *LER*. Les paquets qu'ils envoient sont directement encapsulés si leur adresse de destination correspond au *FEC* existant. Voici comment les configurer :

- Vectra15 :

Pour commencer, il faut ajouter une route afin d'indiquer au *LER* l'existence de vectra13. Pour cela, taper la commande suivante :

```
route add [-net] prefix / mask gw gateway
```

- *-net* : Indique que le *prefix* correspond à un sous-réseau.
- *prefix* : Adresse d'un *host* ou d'un sous-réseau.
- *mask* : Masque de sous-réseau.
- *gateway* : Adresse du chemin à utiliser pour atteindre l'adresse du *prefix*

➤ `route add 10.2.0.1/32 gw 10.1.0.2`

Pour chaque interface il faut définir un *label space*. Ce *label space* à toujours une valeur nul pour un réseau IP.

Mplsadm -v -L interface : label space

- *-v* : Permet d'avoir un écho de la commande tapée.
- *-L* : Option indiquant la configuration du *label space* sur une interface.
- *interface* : Interface sur laquelle le *label space* vas être configuré
- *label space* : Valeur du *label space*.

➤ `mplsadm -v -L eth0:0`

La commande suivante va définir le *FEC* vers vectra 13, et créer un lien avec le label 21 utilisé pour encapsuler les paquets vers cette destination.

**Mplsadm -v -A -B -O label type : label : interface : next type : next hop
-f prefix / len**

- *-v* : Permet d'avoir un écho de la commande tapée.
- *-A* : Indique l'ajout d'un élément.
- *-B* : Indique que le *FEC* et le label de sortie sont associés.
- *-O* : Indique la configuration d'un label de sortie.
- *label type* : Indique le type de label utilisé (*generic, ATM, fram relay*).
- *label* : Numéro du label.
- *interface* : Interface par lequel le paquet doit être envoyé.
- *next type* : Indique le type du paquet encapsulé.
- *Next hop* : Adresse du prochain *LSR* ou *LER*.
- *-f* : Indique la création d'un *FEC*.
- *prefix* : Adresse de destination du *LSP*.
- *len* : Masque du sous-réseau.

➤ `mplsadm -v -A -B -O gen:21:eth0:ipv4:10.1.0.2 -f 10.2.0.1/32`

Remarque : Pour supprimer cette commande, remplacer l'option **-A** (add) par **-D** (del).

Pour terminer, il faut indiquer au *LER* que les paquets comportant le label 43 doivent être décapsulés.

Mplsadm -v -A -I label type : label : label space

- *-v* : Permet d'avoir un écho de la commande tapée.
- *-A* : Indique l'ajout d'un élément.
- *-I* : Indique la configuration d'un label en entrée.
- *label type* : Indique le type de label utilisé (*generic, ATM, fram relay*).
- *label* : Numéro du label.
- *label space* : Valeur du *label space*.

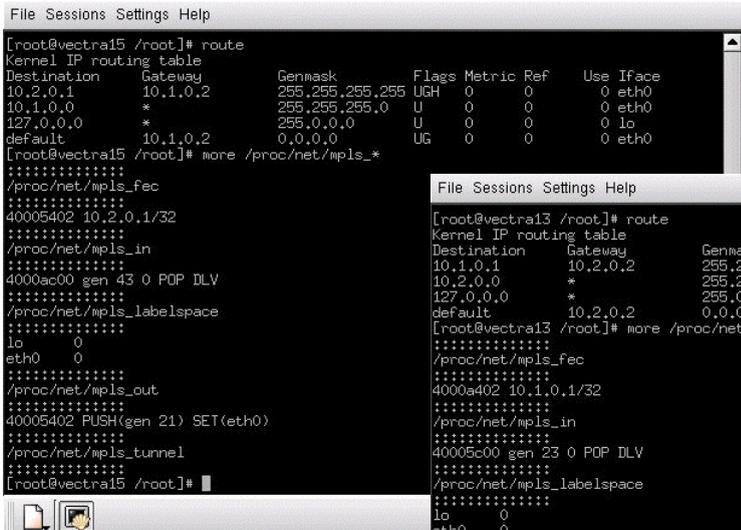
➤ `mplsadm -v -A -I gen:43:0`

- Vectra13 :

La configuration de vectra13 se fait de la même manière que pour vectra15.

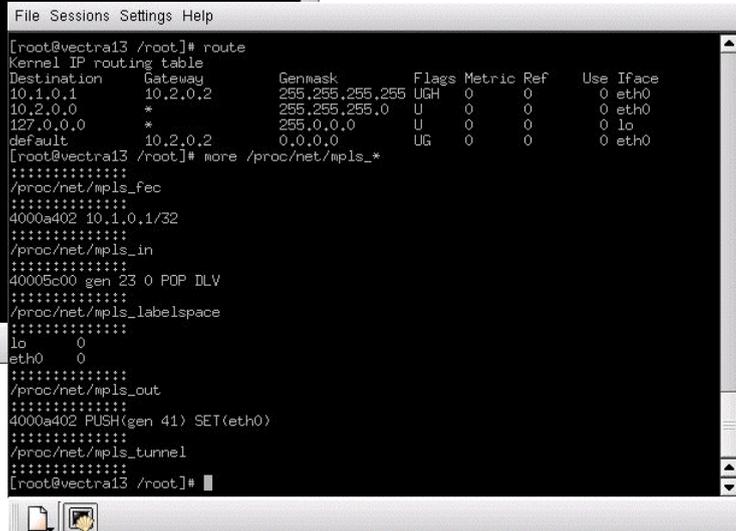
- `route add 10.1.0.1/32 gw 10.2.0.2`
- `mplsadm -v -L eth0:0`
- `mplsadm -v -A -B -O gen:41:eth0:ipv4:10.2.0.2 -f 10.1.0.1/32`
- `mplsadm -v -A -I gen:23:0`

Pour contrôler la configuration de ces deux machines, la commande **route** permet d'afficher toute la table de routage de la machine. En tapant **more /proc/net/mpls_***, il est possible de voir la configuration des *LER* (voir figures ci-dessous).



```
File Sessions Settings Help
[root@vectra15 /root]# route
Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref Use Iface
10.2.0.1 10.1.0.2 255.255.255.255 UGH 0 0 0 eth0
10.1.0.0 * 255.255.255.0 U 0 0 0 eth0
127.0.0.0 * 255.0.0.0 U 0 0 0 lo
default 10.1.0.2 0.0.0.0 UG 0 0 0 eth0
[root@vectra15 /root]# more /proc/net/mpls_*
:::
/proc/net/mpls_fec
:::
40005402 10.2.0.1/32
/proc/net/mpls_in
:::
4000ac00 gen 43 0 POP DLV
/proc/net/mpls_labelspace
:::
lo 0
eth0 0
/proc/net/mpls_out
:::
40005402 PUSH(gen 21) SET(eth0)
/proc/net/mpls_tunnel
:::
[root@vectra15 /root]#
```

FIGURE 5.5 : Vectra15



```
File Sessions Settings Help
[root@vectra13 /root]# route
Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref Use Iface
10.1.0.1 10.2.0.2 255.255.255.255 UGH 0 0 0 eth0
10.2.0.0 * 255.255.255.0 U 0 0 0 eth0
127.0.0.0 * 255.0.0.0 U 0 0 0 lo
default 10.2.0.2 0.0.0.0 UG 0 0 0 eth0
[root@vectra13 /root]# more /proc/net/mpls_*
:::
/proc/net/mpls_fec
:::
4000a402 10.1.0.1/32
/proc/net/mpls_in
:::
40005c00 gen 23 0 POP DLV
/proc/net/mpls_labelspace
:::
lo 0
eth0 0
/proc/net/mpls_out
:::
4000a402 PUSH(gen 41) SET(eth0)
/proc/net/mpls_tunnel
:::
[root@vectra13 /root]#
```

FIGURE 5.6 : Vectra13

En regardant la FIGURE 5.5 on constate les choses suivantes. On voit la route vers vectra13 (10.2.0.1). On constate que le *FEC* pour cette destination a été créé. Le numéro 40005402 précédant le *FEC* et le label 21 permettent de créer le lien entre eux deux. Lorsqu'un paquet correspondant à ce *FEC* doit être envoyé, le label 21 sera ajouté (*PUSH*) et envoyé par l'interface eth0 (*SET*). Lorsqu'un paquet sera reçu avec le label 43, ce label sera retiré (*POP*) et le paquet sera délivré à l'adresse IP de destination (*DLV*).

5.3.3. Configuration des LSR

A présent, il faut configurer vectra16 et vectra12 comme des *LSR* afin de *forwarder* les paquets sur les *LSP*.

- Vectra16 :

On commence par définir le *label space* de chaque interface.

- `mplsadm -v -L eth0:0`
- `mplsadm -v -L eth1:0`
- `mplsadm -v -L eth2:0`

Remarque : eth1 n'est pas encore utilisé dans cette topologie, mais il faut quand même définir son *label space* pour que le LSR fonctionne correctement.

La commande suivante permet de définir le *swapping* des labels afin de créer un *LSP*. Par exemple, le *LSR* reçoit un paquet avec le label 21, il va le remplacer par le label 22 et *forwarder* le paquet par l'interface eth2. Voici comment configurer cela :

**Mplsadm -v -A -I label type : label :label space
-O label type : label :interface :next type :next hop -B**

- *-v* : Permet d'avoir un écho de la commande tapée.
- *-A* : Indique l'ajout d'un élément.
- *-I* : Indique la configuration d'un label en entrée.
- *-O* : Indique la configuration d'un label de sortie.
- *label type* : Indique le type de label utilisé (*generic, ATM, fram relay*).
- *label* : Numéro du label.
- *interface* : Interface par lequel le paquet doit être envoyé.
- *next type* : Indique le type du paquet encapsulé.
- *Next hop* : Adresse du prochain *LSR* ou *LER*.
- *label space* : Valeur du *label space*.
- *-B* : Indique que le label d'entrée et le label de sortie sont associés.

- `mplsadm -v -A -I gen:21:0 -O gen:22:eth2:ipv4:10.3.0.2 -B`

Idem pour le second *LSP* :

- `mplsadm -v -A -I gen:42:0 -O gen:43:eth0:ipv4:10.1.0.1 -B`

- Vectra12 :

La configuration de vectra12 se fait de manière identique que celle de vectra16.

- `mplsadm -v -L eth0:0`
- `mplsadm -v -L eth1:0`
- `mplsadm -v -L eth2:0`
- `mplsadm -v -A -I gen:22:0 -O gen:23:eth0:ipv4:10.2.0.1 -B`
- `mplsadm -v -A -I gen:41:0 -O gen:42:eth2:ipv4:10.3.0.1 -B`

Les figures ci-dessous montrent la configuration des deux *LSR* obtenues grâce aux commandes **route** et **more /proc/net/mpls_***. Par exemple, sur la FIGURE 5.7, on voit que lorsqu'un paquet possédant le label 21 est reçu, le label est retiré (*POP*) et que le paquet est *forwardé* (*FWD*) à travers l'interface eth2 (*SET*) et encapsulé avec le label 22 (*PUSH*). On remarque le lien entre l'entrée et la sortie grâce au numéro 40005400.

```

[root@vectra16 /root]# route
Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref Use IFace
10.4.0.0 * 255.255.255.0 U 0 0 0 eth1
10.1.0.0 * 255.255.255.0 U 0 0 0 eth0
10.3.0.0 * 255.255.255.0 U 0 0 0 eth2
127.0.0.0 * 255.0.0.0 U 0 0 0 lo
default * 0.0.0.0 U 0 0 0 eth0

[root@vectra16 /root]# more /proc/net/mpls_*
../../../../
/proc/net/mpls_in
40005400 gen 21 0 POP FWD(40005804)
4000a900 gen 42 0 POP FWD(4000a002)
../../../../
/proc/net/mpls_labelSpace
lo 0
eth0 0
eth1 0
eth2 0
../../../../
/proc/net/mpls_out
40005904 PUSH(gen 22) SET(eth2)
4000ac02 PUSH(gen 43) SET(eth0)
../../../../
/proc/net/mpls_tunnel
../../../../
[root@vectra16 /root]#

```

FIGURE 5.7 : Vectra16

```

[root@vectra12 /root]# route
Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref Use IFace
10.2.0.0 * 255.255.255.0 U 0 0 0 eth0
10.3.0.0 * 255.255.255.0 U 0 0 0 eth2
10.5.0.0 * 255.255.255.0 U 0 0 0 eth1
127.0.0.0 * 255.0.0.0 U 0 0 0 lo
default * 0.0.0.0 U 0 0 0 eth0

[root@vectra12 /root]# more /proc/net/mpls_*
../../../../
/proc/net/mpls_in
40005900 gen 22 0 POP FWD(40005e02)
4000a400 gen 41 0 POP FWD(4000a804)
../../../../
/proc/net/mpls_labelSpace
lo 0
eth0 0
eth1 0
eth2 0
../../../../
/proc/net/mpls_out
40005c02 PUSH(gen 23) SET(eth0)
4000a904 PUSH(gen 42) SET(eth2)
../../../../
/proc/net/mpls_tunnel
../../../../
[root@vectra12 /root]#

```

FIGURE 5.8 : Vectra12

5.3.4. Tests et analyse des paquets

Pour l'analyse du trafic sur le réseau, le logiciel Ethereal est utilisé. En effet, ce programme décode les paquets *MPLS*. Cet analyseur est disponible pour Windows et Linux. La première solution envisagée a été d'installer l'analyseur sur chaque machine. Malheureusement, on constate que le décodage des paquets sortant des interfaces est erroné. Par contre les paquets entrant dans les interfaces sont décodés correctement. Pour résoudre ce problème, les mesures sont faites en insérant un hub sur les connexions reliant les différentes machines. Un analyseur Ethereal connecté sur ce hub reçoit ainsi tous les paquets en entrée sur son interface et peut donc les analyser correctement.

```

[root@vectra15 /root]# ping 10.2.0.1
PING 10.2.0.1 (10.2.0.1) from 10.1.0.1 : 56(84) bytes of data.
64 bytes from 10.2.0.1: icmp_seq=0 ttl=255 time=355 usec
64 bytes from 10.2.0.1: icmp_seq=1 ttl=255 time=352 usec
64 bytes from 10.2.0.1: icmp_seq=2 ttl=255 time=348 usec
64 bytes from 10.2.0.1: icmp_seq=3 ttl=255 time=346 usec
--- 10.2.0.1 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/mdev = 0,346/0,350/0,355/0,013 ms
[root@vectra15 /root]#

```

FIGURE 5.9 : Ping sur vectra13

Comme le montre la figure ci-dessus, le test des *LSP* est fait avec un ping sur vectra13 (ping 10.2.0.1). Des captures de paquets ont été effectuées sur les différentes connexions du réseau. Toutes ces captures se trouvent dans les annexes *Captures 1*. Un des paquets «*Echo request*» capturés entre vectra15 et vectra16 est représenté ci-dessous :

```
Frame 1 (102 on wire, 102 captured)
  Arrival Time: Nov  6, 2001 11:07:37.562153
  Time delta from previous packet: 0.000000 seconds
  Time relative to first packet: 0.000000 seconds
  Frame Number: 1
  Packet Length: 102 bytes
  Capture Length: 102 bytes
Ethernet II
  Destination: 00:01:02:0e:da:0e (00:01:02:0e:da:0e)
  Source: 00:01:02:0e:da:97 (00:01:02:0e:da:97)
  Type: MPLS label switched packet (0x8847)
MultiProtocol Label Switching Header
  MPLS Label: Unknown (21)
  MPLS Experimental Bits: 0
  MPLS Bottom Of Label Stack: 1
  MPLS TTL: 64
Internet Protocol, Src Addr: 10.1.0.1 (10.1.0.1), Dst Addr: 10.2.0.1 (10.2.0.1)
  Version: 4
  Header length: 20 bytes
  Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
    0000 00.. = Differentiated Services Codepoint: Default (0x00)
    .... ..0. = ECN-Capable Transport (ECT): 0
    .... ...0 = ECN-CE: 0
  Total Length: 84
  Identification: 0x0000
  Flags: 0x04
    .1.. = Don't fragment: Set
    ..0. = More fragments: Not set
  Fragment offset: 0
  Time to live: 64
  Protocol: ICMP (0x01)
  Header checksum: 0x26a5 (correct)
  Source: 10.1.0.1 (10.1.0.1)
  Destination: 10.2.0.1 (10.2.0.1)
Internet Control Message Protocol
  Type: 8 (Echo (ping) request)
  Code: 0
  Checksum: 0x1e61 (correct)
  Identifier: 0xa004
  Sequence number: 00:00
  Data (56 bytes)
```

FIGURE 5.10 : Paquet *Echo request*

Sur ce paquet, on constate que le label *MPLS* est bien inséré entre l'entête Ethernet et l'entête IP. On voit les quatre champs qui le constituent. Le numéro du label est 21, le champ *Experimental* vaut 0, le bit de *Stack* est à 1 et le champ *TTL* vaut 64. Sur les captures annexées, on voit que tous les paquets sont bien encapsulés avec les labels indiqués à la FIGURE 5.4. Par contre, on constate que le champ *TTL* de ces paquets n'est pas décrétementé lors du passage dans les *LSR*. Ce bug a déjà été signalé sur le forum du site Internet sourceforge.net.

Les deux *LSP* ont bien été constitués et le *swapping* des labels fonctionne correctement.

5.4. Création d'un *stack* de labels

5.4.1. Objectif

Comme nous l'avons vu dans le chapitre 4.4, il est possible de créer une pile de labels (*label stack*). Par exemple, ce mécanisme peut être utilisé pour rerouter les paquets sur un chemin de secours quand une ligne entre deux *LSR* est coupée. Reprenons le même réseau que la FIGURE 5.4. Admettons que la liaison entre LSR1 et LSR2 soit coupée. Quand le LSR1 reçoit un paquet avec le label 21, il le remplace par le label 22. Mais comme la connexion vers LSR2 est coupée, il ne peut pas envoyer le paquet sur l'interface eth2. LSR1 utilise donc un chemin de secours. Ce chemin correspond à un *LSP* constitué par les labels 61 et 62. Donc, vectra16 ajoute un nouveau label (61) au paquet et l'envoie sur l'interface eth1 (FIGURE 5.11). Quand LSR3 reçoit le paquet avec le label 61, il remplace ce label avec le label 62 et *forward* le paquet vers LSR2. Quand LSR2 reçoit un paquet avec le label 62, il supprime le label au sommet de la pile et regarde le numéro du label restant. En voyant le label 22, LSR2 sait qu'il doit le remplacer par le label 23 et l'envoyer sur l'interface eth0. Le paquet arrive à destination. Ce reroutage est fait de manière transparente pour vectra15 et vectra13. En effet, il n'y a aucun changement pour ces machines. Ce mécanisme diminue donc le nombre de paquets perdus. Lorsque LSR1 détecte la coupure, le reroutage est effectué sans devoir attendre que l'information sur ce défaut se propage aux extrémités du *LSP*

Remarque : Les *LSR* utilisés sur le *backbone* détectent les coupures de lignes et reroute les paquets automatiquement. Sur nos machines cela n'est pas possible. Les paquets sont reroutés manuellement.

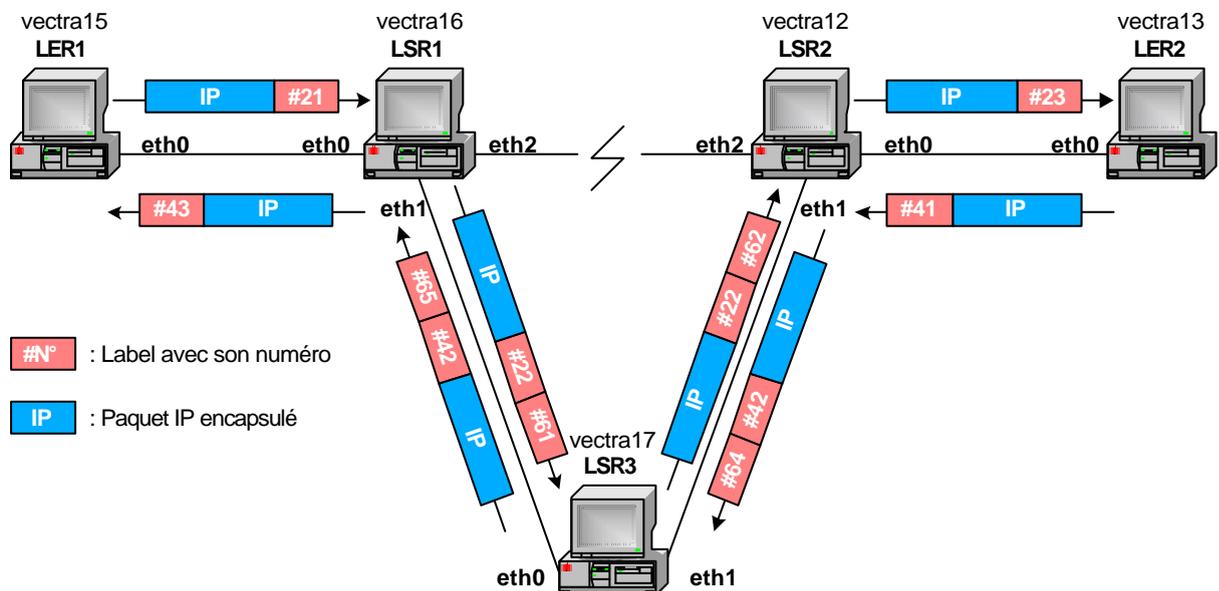


FIGURE 5.11 : Reroutage des paquets

Comme indiqué ci-dessus, il n'y a aucun changement de configuration à apporter à vectra15 et vectra13. Seule la configuration des *LSR* doit être modifiée.

5.4.2. Configuration des LSR

- Vectra16 :

Voici ce qu'il faut ajouter à la configuration de vectra16 pour utiliser le chemin de secours. Pour commencer, il faut indiquer le label utilisé sur ce chemin. Ce label est le 61.

```
➤ mplsadm -v -A -O gen:61:eth1:ipv4:10.4.0.2
```

Ensuite, il faut rerouter les paquets, possédant un label 22 et devant utiliser l'interface eth2, sur l'interface eth1 et utiliser le label 61.

```
➤ mplsadm -v -O gen:22:eth2 -o push:gen:22:fwd:gen:61:eth1
```

-o permet d'ajouter ou de modifier un label sur un paquet en sortie. Dans notre cas, ce sont les paquets sortant sur l'interface eth2 avec un label 22 qui doivent être modifiés. *Push* indique l'ajout d'un label au sommet de la pile. *Fwd* indique que le paquet est *forwardé* avec le label 61 sur l'interface eth1.

Lorsque le *LSR* reçoit un paquet avec le label 65, il doit supprimer ce label du sommet de la pile (*pop*) et regarder le label se trouvant en dessous pour *forwarder* le paquet (*peek*).

```
➤ mplsadm -v -A -I gen:65:0
➤ mplsadm -v -I gen:65:0 -i pop:peek
```

-i permet de supprimer ou de modifier un label sur un paquet en entrée.

- Vectra12 :

Les modifications de configuration sur vectra12 sont du même type que sur vectra16.

```
➤ mplsadm -v -A -O gen:64:eth1:ipv4:10.5.0.2
➤ mplsadm -v -O gen:42:eth2 -o push:gen:42:fwd:gen:64:eth1
➤ mplsadm -v -A -I gen:62:0
➤ mplsadm -v -I gen:62:0 -i pop:peek
```

-Vectra17 :

Vetra17 doit *forwarder* les paquets à travers les bonnes interfaces et remplacer les labels selon la FIGURE 5.11.

```
➤ mplsadm -v -L eth0:0
➤ mplsadm -v -L eth1:0
➤ mplsadm -v -A -I gen:61:0 -O gen:62:eth1:ipv4:10.5.0.1 -B
➤ mplsadm -v -A -I gen:64:0 -O gen:65:eth0:ipv4:10.4.0.1 -B
```

Les figures ci-dessous montrent la configuration des *LSR* en utilisant la commande **more /proc/net/mpls_***. Sur la FIGURE 5.12 on constate que lorsqu'un paquet est reçu avec le label 21, ce label est supprimé (*pop*), le paquet est *forwardé* (*fwd*) et

encapsulé avec le label 22 (*push*). Ensuite, le paquet est *forwardé* et encapsulé une deuxième fois avec le label 61. Finalement, le paquet est envoyé par l'interface eth1. Lorsque le paquet reçu possède le label 65, celui-ci est supprimé (*pop*). Le label se trouvant en dessous est utilisé pour déterminer le chemin du paquet (*peek*).

```

File Sessions Settings Help
[root@vectra16 /root]* more /proc/net/mpls_*
#####
/proc/net/mpls_fac
#####
/proc/net/mpls_in
#####
40005400 gen 21 0 PDP FND(40005B04)
40005800 gen 42 0 PDP FND(40005C02)
40010400 gen 65 0 PDP FED<
#####
/proc/net/mpls_labelspace
#####
lo 0
eth0 0
eth1 0
eth2 0
#####
/proc/net/mpls_out
#####
40005B04 PUSH(gen 22) FND(4000F403)
40005C02 PUSH(gen 43) SET(eth0)
4000F403 PUSH(gen 61) SET(eth1)
#####
/proc/net/mpls_tunnel
#####
[root@vectra16 /root]*

```

FIGURE 5.12 : Vectra16

```

File Sessions Settings Help
[root@vectra12 /root]* more /proc/net/mpls_*
#####
/proc/net/mpls_fac
#####
/proc/net/mpls_in
#####
40005800 gen 22 0 PDP FND(40005C02)
40005B00 gen 41 0 PDP FND(40005B04)
4000FB00 gen 62 0 PDP FED<
#####
/proc/net/mpls_labelspace
#####
lo 0
eth0 0
eth1 0
eth2 0
#####
/proc/net/mpls_out
#####
40005C02 PUSH(gen 23) SET(eth0)
40005B04 PUSH(gen 42) FND(40010003)
40010003 PUSH(gen 64) SET(eth1)
#####
/proc/net/mpls_tunnel
#####
[root@vectra12 /root]*

```

FIGURE 5.13 : Vectra12

```

File Sessions Settings Help
[root@vectra17 /root]* more /proc/net/mpls_*
#####
/proc/net/mpls_fac
#####
/proc/net/mpls_in
#####
4000F400 gen 61 0 PDP FND(4000FB03)
40010000 gen 64 0 PDP FND(40010402)
#####
/proc/net/mpls_labelspace
#####
lo 0
eth0 0
eth1 0
#####
/proc/net/mpls_out
#####
4000F403 PUSH(gen 62) SET(eth1)
40010402 PUSH(gen 65) SET(eth0)
#####
/proc/net/mpls_tunnel
#####
[root@vectra17 /root]*

```

FIGURE 5.14 : Vectra17

5.4.3. Tests et analyse des paquets

En effectuant un ping sur vectra13 (ping 10.2.0.1) depuis vectra15, nous avons constaté que les paquets n'atteignent pas leur destination. En branchant l'analyseur sur les différentes liaisons, nous avons vu que les paquets circulent normalement jusqu'au LSR2. Ensuite, LSR2 ne *forwarde* plus les paquets vers vectra13. En effectuant un ping sur vectra15 (ping 10.1.0.1) depuis vectra13 nous avons constaté le même phénomène. Les paquets circulent normalement jusqu'au LSR1, mais ne sont pas *forwardé* vers vectra15. La cause de ce problème n'a pas pu être expliquée. Toutefois, les soupçons se portent sur la commande *peek*. Cette commande est utilisée après le retrait du label se trouvant au sommet du *label stack*. Elle indique à LSR qu'il doit utiliser le label se trouvant maintenant au sommet de la pile pour *forwarder* les paquets. Comme nous l'avons dit plus haut, certains bugs sont encore possible avec le patch « mpls-linux-0.993 ».

Ce problème ne nous empêche pas de capturer des paquets possédant un *label stack*. Ces paquets sont capturés sur la liaison entre vectra16 et vectre17 et sur la liaison entre vectra17 et vectra12. Afin de capturer des paquets circulant dans les deux sens, des ping sont effectuer sur vectra13 depuis vectra15 et inversement. Les captures se trouvent dans l'annexe *Captures 2*. Ci-dessous est représenté un paquet capturé entre vectra16 et vectra17. On voit bien l'empilement des deux labels entre l'entête Ethernet et l'entête IP. Les numéros des labels (61&22) correspondent bien aux valeurs de la FIGURE 5.11. On constate aussi que le bit de *stack* du label supérieur est bien à 0 et celui du label inférieur est à 1.

```
Frame 1 (106 on wire, 106 captured)
  Arrival Time: Nov  7, 2001 09:45:40.707262
  Time delta from previous packet: 0.000000 seconds
  Time relative to first packet: 0.000000 seconds
  Frame Number: 1
  Packet Length: 106 bytes
  Capture Length: 106 bytes
Ethernet II
  Destination: 00:01:02:0e:d9:f4 (00:01:02:0e:d9:f4)
  Source: 00:01:02:b7:2e:60 (00:01:02:b7:2e:60)
  Type: MPLS label switched packet (0x8847)
MultiProtocol Label Switching Header
  MPLS Label: Unknown (61)
  MPLS Experimental Bits: 0
  MPLS Bottom Of Label Stack: 0
  MPLS TTL: 64
MultiProtocol Label Switching Header
  MPLS Label: Unknown (22)
  MPLS Experimental Bits: 0
  MPLS Bottom Of Label Stack: 1
  MPLS TTL: 64
Internet Protocol, Src Addr: 10.1.0.1 (10.1.0.1), Dst Addr: 10.2.0.1 (10.2.0.1)
  Version: 4
  Header length: 20 bytes
  Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
    0000 00.. = Differentiated Services Codepoint: Default (0x00)
      .... ..0. = ECN-Capable Transport (ECT): 0
      .... ...0 = ECN-CE: 0
  Total Length: 84
  Identification: 0x0000
  Flags: 0x04
    .1.. = Don't fragment: Set
    ..0. = More fragments: Not set
  Fragment offset: 0
  Time to live: 64
  Protocol: ICMP (0x01)
  Header checksum: 0x26a5 (correct)
  Source: 10.1.0.1 (10.1.0.1)
  Destination: 10.2.0.1 (10.2.0.1)
Internet Control Message Protocol
  Type: 8 (Echo (ping) request)
  Code: 0
  Checksum: 0xd8c4 (correct)
  Identifier: 0x2305
  Sequence number: 00:00
  Data (56 bytes)
```

FIGURE 5.15 : Paquet avec un label stack

Comme on le voit, l'empilement des labels fonctionne correctement. Malheureusement, lorsque le label supérieur est supprimé par un LSR, le paquet n'est plus *forwardé*.

5.5. Réseau IP et MPLS

5.5.1. Objectif

Dans les deux scénarios précédents, les paquets étaient encapsulés depuis leur source jusqu'à leur destination. En effet, les machines émettrice et réceptrice étaient des *LER*. Afin de mieux comprendre le rôle de ces *LER* dans un réseau *MPLS*, nous allons créer un réseau client serveur. Ces deux machines se trouvent respectivement sur le sous-réseau 10.1.0.0/24 et 10.2.0.0/24. Elles émettent et reçoivent des paquets IP classiques. Chaque sous réseau possède un *LER*. Le but est de créer deux *LSP* reliant ces sous réseaux (FIGURE 5.16). Les *FEC* associés à ces *LSP* correspondent aux sous-réseaux à atteindre.

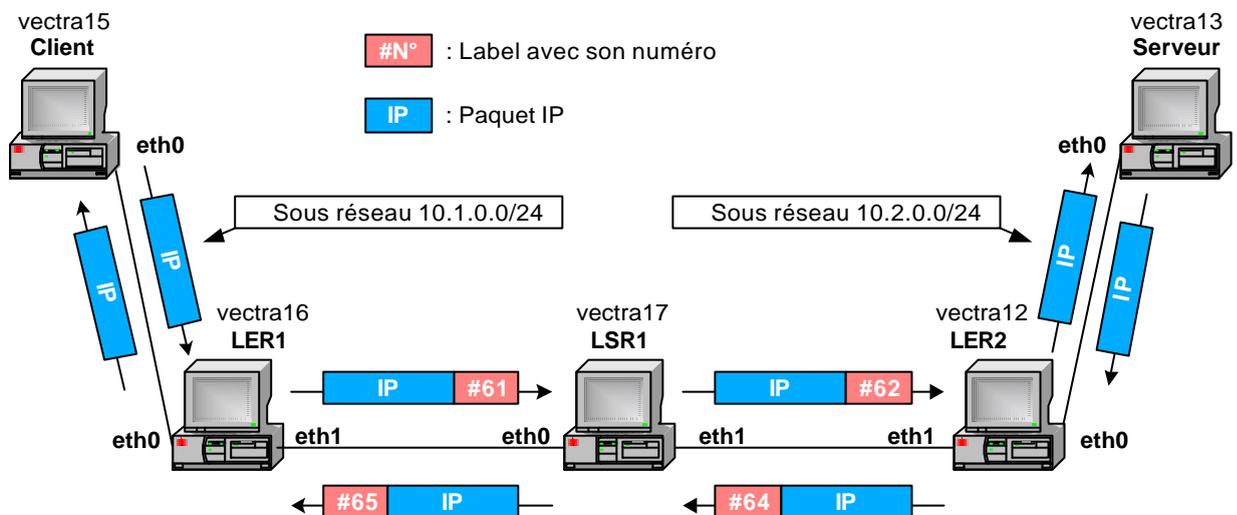


FIGURE 5.16 : Sous réseaux reliés par deux LSP

5.5.2. Configuration des LER

Quand un *LER* reçoit, sur l'une de ses interfaces, un paquet *MPLS* à destination du sous-réseau auquel il est connecté, ce paquet est décapsulé et *forwardé* en fonction de l'adresse IP de destination. Pour permettre de *forwarder* ce paquet IP, l'option «*IP : advanced router*» doit être activée dans le noyau Linux (voir annexe *Installation des patches sur LINUX*). De plus pour permettre la fonction de *forwarding*, la commande suivante doit être tapée sur les deux *LER*.

```
➤ echo "1" > /proc/sys/net/ipv4/ip_forwarding
```

- Vectra16 :

Pour commencer, il faut ajouter une route afin d'indiquer au *LER*, l'existence de sous-réseau 10.2.0.0/24. Pour cela, taper la commande ci-dessous :

```
➤ route add -net 10.2.0.0/24 gw 10.4.0.2
```

Pour chaque interface il faut définir un *label space* de 0. L'interface eth2 n'est pas utilisée dans cette topologie, mais doit aussi être configurée.

- mplsadm -v -L eth0:0
- mplsadm -v -L eth1:0
- mplsadm -v -L eth2:0

La commande suivante va définir le *FEC* vers le sous-réseau 10.2.0.0/24, et créer un lien avec le label 61 utilisé pour encapsuler les paquets pour cette destination.

- mplsadm -v -A -B -O gen:61:eth1:ipv4:10.4.0.2 -f 10.2.0.0/24

Pour terminer, il faut indiquer au *LER* que les paquets comportant le label 65 doivent être décapsulés.

- mplsadm -v -A -I gen:65:0

- Vectra12 :

La configuration de vectra12 se fait de la même manière que pour vectra16.

- route add 10.1.0.0/24 gw 10.5.0.2
- mplsadm -v -L eth0:0
- mplsadm -v -L eth1:0
- mplsadm -v -L eth2:0
- mplsadm -v -A -B -O gen:64:eth1:ipv4:10.5.0.2 -f 10.1.0.0/24
- mplsadm -v -A -I gen:62:0

Les figures ci-dessous montrent la configuration des *LER* en utilisant la commande `more /proc/net/mpls_*` :

```

[root@vectra16 /root]# route
Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref Use Iface
10.2.0.0 * 10.4.0.2 255.255.255.0 U 0 0 0 eth1
10.4.0.0 * 255.255.255.0 U 0 0 0 eth1
10.1.0.0 * 255.255.255.0 U 0 0 0 eth0
10.3.0.0 * 255.255.255.0 U 0 0 0 eth2
127.0.0.0 * 255.0.0.0 U 0 0 0 lo
default * 0.0.0.0 U 0 0 0 eth0

[root@vectra16 /root]# more /proc/net/mpls_*
#####
/proc/net/mpls_fec
#####
4000403 10.2.0.0/24
#####
/proc/net/mpls_ln
#####
4000400 gen 65 0 ROP DLY
#####
/proc/net/mpls_labelspace
#####
lo 0
eth0 0
eth1 0
eth2 0

/proc/net/mpls_out
#####
4000403 PUSH(gen 61) SET(eth1)
#####
/proc/net/mpls_tunnel
#####
[root@vectra16 /root]#

```

FIGURE 5.17 : Vectra16

```

[root@vectra12 /root]# route
Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref Use Iface
10.2.0.0 * 10.5.0.2 255.255.255.0 U 0 0 0 eth0
10.1.0.0 * 255.255.255.0 U 0 0 0 eth1
10.3.0.0 * 255.255.255.0 U 0 0 0 eth2
127.0.0.0 * 255.0.0.0 U 0 0 0 lo
default * 0.0.0.0 U 0 0 0 eth0

[root@vectra12 /root]# more /proc/net/mpls_*
#####
/proc/net/mpls_fec
#####
4000803 10.1.0.0/24
#####
/proc/net/mpls_ln
#####
4000803 gen 62 0 ROP DLY
#####
/proc/net/mpls_labelspace
#####
lo 0
eth0 0
eth1 0
eth2 0

/proc/net/mpls_out
#####
4000803 PUSH(gen 64) SET(eth1)
#####
/proc/net/mpls_tunnel
#####
[root@vectra12 /root]#

```

FIGURE 5.18 : Vectra12

Sur la FIGURE 5.17 on voit la route vers le sous-réseau 10.2.0.0/24. On constate que le *FEC* pour cette destination a été créé. Lorsqu'un paquet correspondant à ce *FEC* doit être envoyé, le label 21 sera ajouté (*PUSH*) et envoyé par l'interface eth1 (*SET*). Lorsqu'un paquet sera reçu avec le label 65, ce label sera retiré (*POP*) et le paquet sera délivré à l'adresse IP de destination (*DLV*).

5.5.3. Configuration des autres machines

- Vectra17 :

La configuration de vectra17 est identique à celle du paragraphe 5.4.

- Vectra15 et vectra13 :

Ces PCs envoient des paquets IP classiques. Aucune configuration *MPLS* n'est donc nécessaire. Par contre, il faut vérifier, avec la commande **route**, que les **default gateway** correspondent aux *LER*. Si cela n'est pas le cas, voici comment faire :

Pour vectra15 :

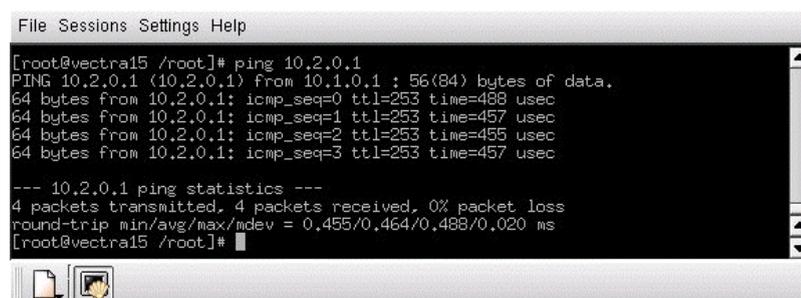
- Route add default gw 10.1.0.2

Pour vectra13 :

- Route add default gw 10.2.0.2

5.5.4. Tests et analyse des paquets

Comme le montre la FIGURE 5.19, le réseau a été testé en effectuant des ping sur le serveur à partir du poste client (ping 10.2.0.1).



```
File Sessions Settings Help
[root@vectra15 /root]# ping 10.2.0.1
PING 10.2.0.1 (10.2.0.1) from 10.1.0.1 : 56(84) bytes of data.
64 bytes from 10.2.0.1: icmp_seq=0 ttl=253 time=488 usec
64 bytes from 10.2.0.1: icmp_seq=1 ttl=253 time=457 usec
64 bytes from 10.2.0.1: icmp_seq=2 ttl=253 time=455 usec
64 bytes from 10.2.0.1: icmp_seq=3 ttl=253 time=457 usec
--- 10.2.0.1 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/mdev = 0.455/0.464/0.488/0.020 ms
[root@vectra15 /root]#
```

FIGURE 5.19 : Ping sur le serveur

Une capture des paquets circulant entre chaque machine a été faite. Ces captures se trouvent dans l'annexe *Captures 3*. On peut constater que les paquets IP provenant du client sont bien encapsulés par LER1, comme indiqué sur la FIGURE 5.16. De même, LER2 décapsule les paquets destinés au serveur. Les *LSP* créés entre les deux sous-réseaux fonctionnent correctement. On constate aussi que, dans les *LER*, le champ *TTL* des paquets IP est bien copié sur le label *MPLS* après avoir été décrémenté. L'opération inverse est aussi effectuée. Par contre, comme nous l'avons vu plus haut, le *LSR* ne décrémente pas le champ *TTL* à cause d'un bug dans le patch « mpls-linux-0.993 ».

5.6. Conclusion

Dans cette partie du travail, certains mécanismes de *forwarding* des paquets *MPLS* ont pu être mis en pratique. Malheureusement, le patch « *mpls-linux-0.993* » utilisé pour ces essais est encore une version en développement. La conséquence de cela a été un travail de plusieurs jours afin de pouvoir régler les problèmes rencontrés lors de l'installation de ce patch. De plus, comme nous l'avons vu plus haut, certains bugs sont encore présents.

Ce travail m'a permis de bien comprendre comment l'encapsulation et le *forwarding* des paquets sont effectués avec le protocole *MPLS*. Il m'a aussi permis d'acquérir des connaissances sur certains mécanismes de Linux comme par exemple la compilation du *kernel*.

6. Etude du mécanisme de distribution des labels avec *LDP* sur Linux

6.1. Introduction

Les scénarios qui vont suivre ont pour but de mieux comprendre comment les labels peuvent être distribués dynamiquement. En effet, pour les tests du chapitre 5, les tables de *forwarding* étaient configurées manuellement dans les *LSR* et les *LER*. Dans les tests suivants, ces machines utilisent le protocole *LDP* (*Label Distribution Protocol*) afin de se transmettre les informations nécessaires à la constitution des *LFIB*. La topologie du réseau utilisé est la même que celle étudiée précédemment (voir FIGURE 5.1). Pour utiliser *LDP* sur ce réseau, il faut ajouter le patch « *ldp-portable-0.060* » sur les machines Linux (le patch « *mpls-linux-0.993* » doit être installé). L'installation de ce patch est expliquée dans l'annexe *Installation des patchs sur LINUX*.

6.2. Transfert *LDP* entre deux *LER*

6.2.1. Objectifs

Le but de ce test est de vérifier que le patch « *ldp-portable-0.060* » fonctionne correctement et de capturer les échanges *LDP* décrits au paragraphe 4.6. Pour cela, *vecetra16* et *vecetra12* fonctionnent comme des *LER* utilisant *LDP*. *Vecetra15* et *vecetra13* jouent respectivement le rôle de client et de serveur. *Vecetra17* n'est pas utilisé dans ce test.

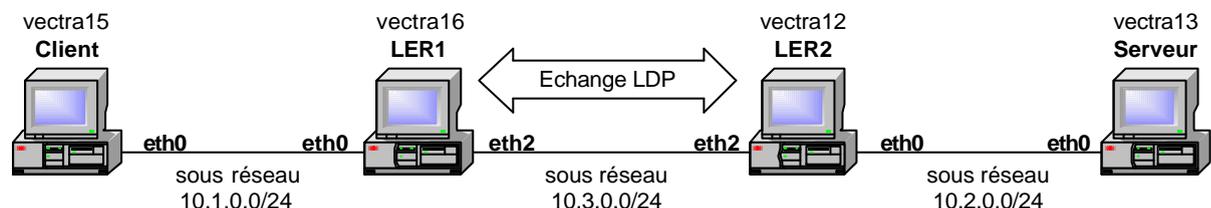


FIGURE 6.1 : Topologie du réseau

6.2.2. Configuration des machines

Vecetra15 et *vecetra13* envoient des paquets IP classiques. Aucune configuration *MPLS* n'est donc nécessaire. Par contre, il faut vérifier que leur *default gateway* correspondent aux *LER* auxquels ils sont connectés. Si cela n'est pas le cas, voici comment faire :

Pour *vecetra15* :

- `Route add default gw 10.1.0.2`

Pour *vecetra13* :

- `Route add default gw 10.2.0.2`

- Vectra16 :

Le patch « ldp-portable-0.060 » impose certaines limites. Il ne permet pas d'attribuer de label pour un sous-réseau complet. Il associe un label uniquement aux destinations exactes (adresses avec un masque de 32 bits) de la table de routage. C'est pourquoi il faut ajouter, à cette table, la route (exacte) permettant d'atteindre le serveur :

```
➤ route add 10.2.0.1/32 gw 10.3.0.2
```

Pour chaque interface, il faut définir le *label space* de 0. L'interface eth1 n'est pas utilisée dans cette topologie, mais doit aussi être configurée ainsi :

```
➤ mplsadm -v -L eth0:0  
➤ mplsadm -v -L eth1:0  
➤ mplsadm -v -L eth2:0
```

Nous allons maintenant configurer *LDP* afin de créer dynamiquement les tables de routages. Pour lancer *LDP* tapez la commande suivante :

```
➤ ldp_linux
```

Selon le protocole *LDP*, chaque routeur possède un identificateur (adresse IP du routeur) permettant d'identifier la machine à l'origine du paquet (champ *LDP Identifier*). La commande **add global** permet de définir cet identificateur. Malheureusement, nous avons constaté qu'en utilisant l'adresse IP de vectra16, aucun label n'est distribué. Le seul moyen trouvé pour effectuer un échange de label est de mettre l'adresse IP du client comme identificateur. Ainsi un label permettant d'atteindre ce client est envoyé. Les raisons de ce dysfonctionnement ne sont pas éclaircies.

```
➤ prompt> add global 10.1.0.1
```

Il faut indiquer au *LER* les interfaces sur lesquelles le protocole *LDP* doit être activé. La commande ci-dessous enclenche le mécanisme de recherche des *LSR* voisins sur l'interface eth2 :

```
➤ prompt> add interface eth2
```

La FIGURE 6.2 montre la table de routage de *LER1* et sa table de *forwarding* après l'échange de labels avec *LER2*. On constate qu'un *FEC* permettant d'atteindre le serveur (10.2.0.1) est créé. Le numéro de label attribué, par *LER2*, pour cette destination, est le 16. Cet échange de label est décrit plus bas. Sur la FIGURE 6.3, on voit ce qui se passe lors de la configuration *LDP*. Lorsque la commande **add global** est exécutée, le processus *LDP* récolte les informations nécessaires à son fonctionnement. Il mémorise les interfaces disponibles et la table de routage. La dernière ligne de cette capture montre la réception d'un *FEC* envoyé par le *Peer*. On constate que ce *FEC* correspond bien à l'adresse du serveur (codage hexadécimal).

```
File Sessions Settings Help
[root@vectra16 /root]# route
Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref Use Iface
10.2.0.1 10.3.0.2 255.255.255.255 UGH 0 0 0 eth2
10.2.0.0 10.3.0.2 255.255.255.0 UG 0 0 0 eth2
10.4.0.0 * 255.255.255.0 U 0 0 0 eth1
10.1.0.0 * 255.255.255.0 U 0 0 0 eth0
10.3.0.0 * 255.255.255.0 U 0 0 0 eth2
127.0.0.0 * 255.0.0.0 U 0 0 0 lo
default * 0.0.0.0 U 0 0 0 eth0
[root@vectra16 /root]# more /proc/net/mps_*
.....
/proc/net/mps_fec
.....
40004004 10.2.0.1/32
.....
/proc/net/mps_in
.....
40004000 gen 16 0 POP DLV
.....
/proc/net/mps_labelspace
.....
lo 0
eth0 0
eth1 0
eth2 0
.....
/proc/net/mps_out
.....
40004004 PUSH(gen 16) SET(eth2)
.....
/proc/net/mps_tunnel
.....
[root@vectra16 /root]#
```

FIGURE 6.2 : Configuration vectra16

```
File Sessions Settings Help
[root@vectra16 /root]# ldp_linux
prompt>add global 10.1.0.1
Adding interface eth0
Adding interface eth1
Adding interface eth2
Adding route 0a020001/32 via 4
Adding route 0a020000/24 via 4
Adding route 0a040000/24 via 3
Adding route 0a010000/24 via 2
Adding route 0a030000/24 via 4
Adding route 7f000000/8 via 0
Adding route 00000000/0 via 2
Adding global object with LSRID 0a010001
prompt>add interface eth2
Adding interface eth2
prompt>FEC: 0100020a/32
```

FIGURE 6.3 : LDP sur vectra16

- Vectra12 :

La configuration de *LER2* se fait de la même manière que celle de *LER1*. Il faut simplement utiliser l'adresse IP du serveur comme identificateur *LDP*.

- route add 10.1.0.1/32 gw 10.3.0.1
- mplsadm -v -L eth0:0
- mplsadm -v -L eth1:0
- mplsadm -v -L eth2:0
- ldp_linux
- prompt> add global 10.2.0.1
- prompt> add interface eth2

```
File Sessions Settings Help
[root@vectra12 /root]# route
Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref Use Iface
10.1.0.1 10.3.0.1 255.255.255.255 UGH 0 0 0 eth2
10.2.0.0 * 255.255.255.0 U 0 0 0 eth0
10.1.0.0 10.3.0.1 255.255.255.0 UG 0 0 0 eth2
10.3.0.0 * 255.255.255.0 U 0 0 0 eth2
10.5.0.0 * 255.255.255.0 U 0 0 0 eth1
127.0.0.0 * 255.0.0.0 U 0 0 0 lo
default * 0.0.0.0 U 0 0 0 eth0
[root@vectra12 /root]# more /proc/net/mps_*
.....
/proc/net/mps_fec
.....
40004004 10.1.0.1/32
.....
/proc/net/mps_in
.....
40004000 gen 16 0 POP DLV
.....
/proc/net/mps_labelspace
.....
lo 0
eth0 0
eth1 0
eth2 0
.....
/proc/net/mps_out
.....
40004004 PUSH(gen 16) SET(eth2)
.....
/proc/net/mps_tunnel
.....
[root@vectra12 /root]#
```

FIGURE 6.4 : Configuration vectra12

```
File Sessions Settings Help
[root@vectra12 /root]# ldp_linux
prompt>add global 10.2.0.1
Adding interface eth0
Adding interface eth1
Adding interface eth2
Adding route 0a010001/32 via 4
Adding route 0a020000/24 via 2
Adding route 0a010000/24 via 4
Adding route 0a030000/24 via 4
Adding route 0a050000/24 via 3
Adding route 7f000000/8 via 0
Adding route 00000000/0 via 2
Adding global object with LSRID 0a020001
prompt>add interface eth2
Adding interface eth2
prompt>FEC: 0100010a/32
```

FIGURE 6.5 : LDP sur vectra12

6.2.3. Tests et analyse des paquets

Avant l'activation de *LDP*, un analyseur Ethereal est placé entre *LER1* et *LER2* afin de capturer les paquets échangés. Cette capture de paquets se trouve dans l'annexe *Captures 4*. Voici comment s'est déroulé cet échange :

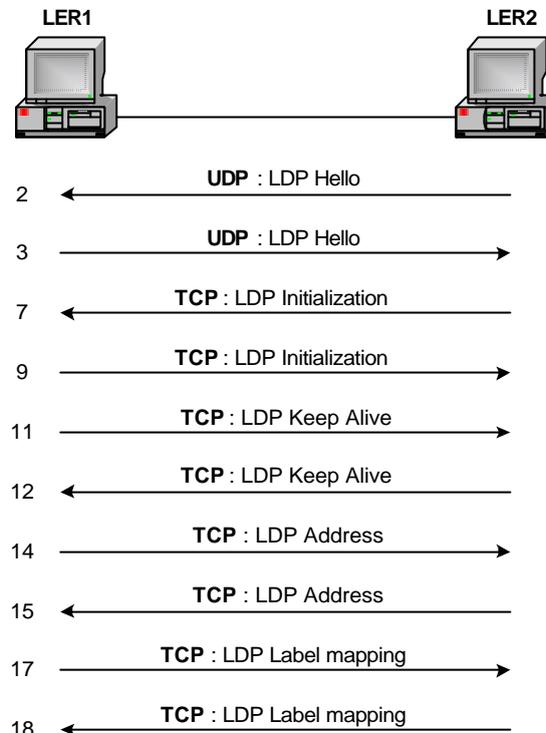


FIGURE 6.6 : Echange *LDP* entre deux *LER*

Cet échange se fait selon le principe décrit au paragraphe 4.7.4. Les numéros indiqués ci-dessus représentent le numéro des paquets de la capture. Seuls les paquets intéressants sont représentés ici.

Voici, ci dessous, le format d'un paquet *LDP Hello* (paquet 2).

```

Label Distribution Protocol
Header
  Version: 1
  PDU Length: 30
  LDP Identifier
    LSR ID: 0x0a020001
    Label Space ID: 0x0000
Hello
  Message Type: Hello (0x0100)
  Message Length: 20
  Message ID: 0x00000001
  Common Hello Parameters
    TLV Type: Common Hello Parameters (0x0400)
    TLV Length: 4
    TLV Value: 000F0000
    Hold Time: 15
    0... .... = Targeted Hello: Link Hello
    .0.. .... = Hello Requested: Source does not request periodic hellos
    ..00 0000 0000 0000 = Reserved: 0x0000
  Configuration Sequence Number
    TLV Type: Configuration Sequence Number (0x0402)
    TLV Length: 4
    Configuration Sequence Number: 0x00000004
  
```

FIGURE 6.7 : Paquet *Hello*

On retrouve les divers champs décrits dans le paragraphe 4.7.5. Le champ *LSR ID* contient bien l'adresse définie grâce à la commande *add global*. Pour les raisons citées ci-dessus, cette adresse est celle du serveur. Le champ *Hold Time* indique la périodicité des messages *Hello*. Dans ce cas, la valeur est de 15 secondes.

Ci-dessous est représenté le paquet d'initialisation envoyé par *LER2* (paquet 7).

```

Label Distribution Protocol
  Header
    Version: 1
    PDU Length: 32
    LDP Identifier
      LSR ID: 0x0a020001
      Label Space ID: 0x0000
  Initialization
    Message Type: Initialization (0x0200)
    Message Length: 22
    Message ID: 0x00000002
    Common Session Parameters
      TLV Type: Common Session Parameters (0x0500)
      TLV Length: 14
      TLV Value: 0001002D000010000A0100010000

```

FIGURE 6.8 : Paquet d'initialisation

Les informations nécessaires à la configuration de la session *LDP* se trouvent dans le champ *TLV Value*.

TLV Value: 0001002D000010000A0100010000

- 0x0001 Indique la version du protocole *LDP* (version 1).
- 0x002D Définit la périodicité des paquets *Keep Alive* (45 secondes).
- 0x0000 Le premier bit de cette valeur indique le mode de distribution des labels. La valeur 0 indique le mode *unsolicited downstream*.
- 0x1000 Indique la taille maximale d'un paquet *LDP* (*PDU*) (4096 octets).
- 0x0A010001 Représente l'ID du *LSR* de destination (10.1.0.1).
- 0x0000 Représente la valeur du *label space*. (toujours à 0 pour Ethernet).

Les options de session envoyées par *LER1* (paquet 9) sont identiques à celles indiquées ci-dessus (à l'exception de l'ID du *LSR* de destination).

Les *LER* acceptent cette configuration en répondant par un paquet *Keep Alive*. Ci-dessous est représenté le paquet 11 :

```

Label Distribution Protocol
  Header
    Version: 1
    PDU Length: 14
    LDP Identifier
      LSR ID: 0x0a010001
      Label Space ID: 0x0000
  Keep Alive
    Message Type: Keep Alive (0x0201)
    Message Length: 4
    Message ID: 0x00000003

```

FIGURE 6.9 : Paquet *Keep Alive*

Les deux *LER* transmettent les adresses de leurs interfaces utilisant *LDP* en utilisant le message *Address*. Cela permet au *LSR* récepteur de faire le lien entre ces adresses

et le *LSR Identifier*, afin de compléter sa table de *forwarding (next hop address)*. Le paquet 14 envoyé par *LER1* est représenté ci-dessous.

```

Label Distribution Protocol
  Header
    Version: 1
    PDU Length: 24
    LDP Identifier
      LSR ID: 0x0a010001
      Label Space ID: 0x0000
  Address
    Message Type: Address (0x0300)
    Message Length: 14
    Message ID: 0x00000004
    Address List
      TLV Type: Address List (0x0101)
      TLV Length: 6
      TLV Value: 00010A030001

```

FIGURE 6.10 : Paquet Address

Le champ *TLV Value* contient le format des adresses et les adresses des interfaces supportant *LDP*.

TLV Value: 00010A030001

- 0x0001 Indique le format IPV4.
- 0x0A030001 Correspond à l'adresse de la seule interface utilisant *LDP* sur *LER1* (10.3.0.1).

Le paquet représenté ci-dessous est le plus intéressant (paquet 17). En effet, c'est lui qui indique les liens entre *FEC* et labels (*label mapping*).

```

Label Distribution Protocol
  Header
    Version: 1
    PDU Length: 34
    LDP Identifier
      LSR ID: 0x0a010001
      Label Space ID: 0x0000
  Label Mapping
    Message Type: Label Mapping (0x0400)
    Message Length: 24
    Message ID: 0x00000007
    Forwarding Equivalence Classes
      TLV Type: Forwarding Equivalence Classes (0x0100)
      TLV Length: 8
      Prefix FEC Element
        FEC Element Type: Prefix FEC (2)
        FEC Element Address Type: IP (IPv4) (1)
        FEC Element Length: 32
        FEC Element Prefix Value: 10.1.0.1 (10.1.0.1)
    Generic Label
      TLV Type: Generic Label (0x0200)
      TLV Length: 4
      Generic Label: 0x00000010

```

FIGURE 6.11 : Paquet Label mapping

Les deux champs qui nous intéressent sont *FEC Element Prefix Value* et *Generic Label*. Grâce à ce paquet, *LER1* signale à *LER2* qu'il doit utiliser le label 16 (0x00000010) pour les paquets à destination du client (*FEC* 10.1.0.1). *LER2* peut ainsi compléter sa table de *forwarding* (FIGURE 6.4).

LSR2 envoie aussi ses liens entre *FEC* et labels, dans le paquet 10, afin de permettre à *LER1* de constituer sa table de *forwarding* (FIGURE 6.2).

La FIGURE 6.12 représente la configuration obtenue grâce au protocole *LDP*. Les paquets 33 et 34 de la *capture 4* annexée correspondent à un *ping* effectué depuis le client sur le serveur. On constate que ces paquets sont encapsulés avec les bonnes valeurs de label.

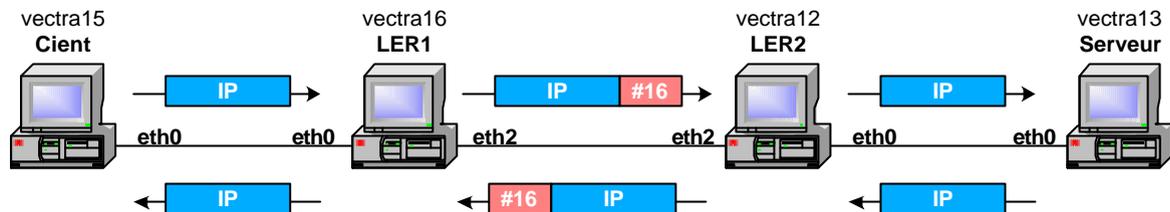


FIGURE 6.12 : Configuration du réseau par *LDP*

Une fois l'échange de label terminé, les *LER* continuent à s'envoyer des messages *Keep Alive* toutes les 15 secondes. Ce mécanisme permet de détecter d'éventuelles pannes ou coupure de connexion. Si tel est le cas, les labels transmis par le *LER* voisin sont simplement retirés de la table de *forwarding*.

6.3. Réseau MPLS utilisant LDP

6.3.1. Objectifs

Le but de ce test est de vérifier que des chemins (*LSP*) peuvent être créés dynamiquement, à travers le réseau *MPLS* ci-dessous, grâce au protocole *LDP*. Ces *LSP* doivent être créés entre *LER1* et *LER2* afin de permettre le passage de paquets entre le client et le serveur.

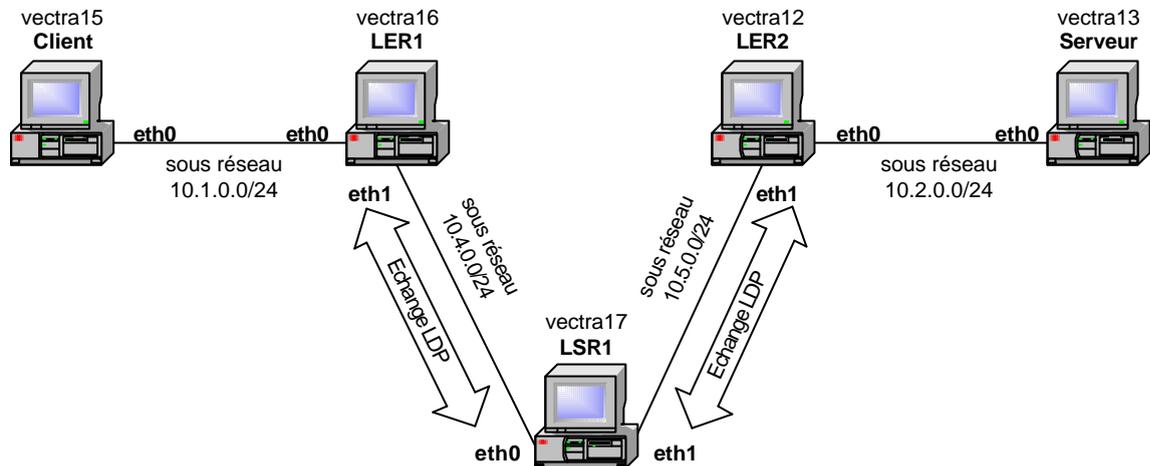


FIGURE 6.13 : Topologie du réseau

6.3.2. Configuration des machines

- Vectra16 :

La configuration de cette machine est identique à celle utilisée précédemment. Il suffit d'activer *LDP* sur l'interface eth1

- ldp_linux
- prompt> add global 10.1.0.1
- prompt> add interface eth1

```
File Sessions Settings Help
[root@vectra16 /root]# route
Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref Use Iface
10.2.0.1 10.4.0.2 255.255.255.255 UGH 0 0 0 eth1
10.4.0.0 * 255.255.255.0 U 0 0 0 eth1
10.1.0.0 * 255.255.255.0 U 0 0 0 eth0
10.3.0.0 * 255.255.255.0 U 0 0 0 eth2
127.0.0.0 * 255.0.0.0 U 0 0 0 lo
default * 0.0.0.0 U 0 0 0 eth0
[root@vectra16 /root]# more /proc/net/mpls_*
::::::::::::
/proc/net/mpls_fec
::::::::::::
40004403 10.2.0.1/32
::::::::::::
/proc/net/mpls_in
::::::::::::
40004000 gen 16 0 POP DLV
::::::::::::
/proc/net/mpls_labelSpace
::::::::::::
lo 0
eth0 0
eth1 0
eth2 0
::::::::::::
/proc/net/mpls_out
::::::::::::
40004403 PUSH(gen 17) SET(eth1)
::::::::::::
/proc/net/mpls_tunnel
::::::::::::
[root@vectra16 /root]#
```

FIGURE 6.14 : Configuration vectra16

```
File Sessions Settings Help
[root@vectra16 /root]# ldp_linux
prompt>add global 10.1.0.1
Adding interface eth0
Adding interface eth1
Adding interface eth2
Adding route 0a020001/32 via 3
Adding route 0a040000/24 via 3
Adding route 0a010000/24 via 2
Adding route 0a030000/24 via 4
Adding route 7f000000/8 via 0
Adding route 00000000/0 via 2
Adding global object with LSRID 0a010001
prompt>add interface eth1
Adding interface eth1
prompt>PRT: ldp_state recv_init; cannot find adj
index: 3 0a050002/32 2
FEC: 0100020a/32
```

FIGURE 6.15 : LDP sur vectra16

- Vectra12 :

La configuration de cette machine est aussi identique à celle utilisée précédemment. Il suffit d'activer *LDP* sur l'interface eth1

- ldp_linux
- prompt> add global 10.2.0.1
- prompt> add interface eth1

```
File Sessions Settings Help
[root@vectra12 /root]# route
Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref Use Iface
10.1.0.1 * 10.5.0.2 255.255.255.255 UGH 0 0 0 eth1
10.2.0.0 * 255.255.255.0 U 0 0 0 eth0
10.3.0.0 * 255.255.255.0 U 0 0 0 eth2
10.5.0.0 * 255.255.255.0 U 0 0 0 eth1
127.0.0.0 * 255.0.0.0 U 0 0 0 lo
default * 0.0.0.0 U 0 0 0 eth0
[root@vectra12 /root]# more /proc/net/mpls_*
::::::::::::
/proc/net/mpls_fec
::::::::::::
40004c03 10.1.0.1/32
::::::::::::
/proc/net/mpls_in
::::::::::::
40004000 gen 16 0 POP DLV
::::::::::::
/proc/net/mpls_labelspace
::::::::::::
lo 0
eth0 0
eth1 0
eth2 0
::::::::::::
/proc/net/mpls_out
::::::::::::
40004c03 PUSH(gen 19) SET(eth1)
::::::::::::
/proc/net/mpls_tunnel
::::::::::::
[root@vectra12 /root]#
```

FIGURE 6.16 : Configuration vectra12

```
File Sessions Settings Help
[root@vectra12 /root]# ldp_linux
prompt>add global 10.2.0.1
Adding interface eth0
Adding interface eth1
Adding interface eth2
Adding route 0a010001/32 via 3
Adding route 0a020000/24 via 2
Adding route 0a030000/24 via 4
Adding route 0a050000/24 via 3
Adding route 7f000000/8 via 0
Adding route 00000000/0 via 2
Adding global object with LSRID 0a020001
prompt>add interface eth1
Adding interface eth1
prompt>PRI: ldp_state_recv_init: cannot find adj
index: 3 0a050002/32 2
FEC: 0100010a/32
```

FIGURE 6.17 : LDP sur vectra12

- Vectra17 :

La configuration du *LSR* s'effectue de la même manière que pour les *LER*. Il doit connaître l'adresse IP du client et du serveur afin d'associer un label à ces préfixes. Pour cela, il faut ajouter les routes suivantes dans sa table de routage :

- route add 10.1.0.1/32 gw 10.4.0.1
- route add 10.2.0.1/32 gw 10.5.0.1

Pour chaque interface, il faut définir le *label space* de 0.

- mplsadm -v -L eth0:0
- mplsadm -v -L eth1:0

Contrairement au *LER*, l'identificateur *LDP* du *LSR* peut être configuré normalement avec une adresse désignant cette machine :

- ldp_linux
- prompt> add global 10.5.0.2

LDP doit être activé sur les deux interfaces du *LSR*.

- prompt> add interface eth0
- prompt> add interface eth1

Sur la FIGURE 6.18, on constate que le *LSR* possède deux *FEC*, après l'échange *LDP*. L'un correspond au préfixe du client (10.1.0.1) et l'autre à celui du serveur (10.2.0.1). Sur la FIGURE 6.19, le *LSR* affiche les *mappings* de labels constituant les deux *LSP* ainsi créés (voir FIGURE 6.21). Ces *mappings* se retrouvent dans la table de *forwarding*. Par exemple, le *LSP* allant du client vers le serveur est constitué des labels 17 et 16. Lorsque le *LSR* reçoit un paquet avec le label 17, ce label est supprimé (*pop*), le paquet est *forwardé* (*fwd*) et encapsulé avec le label 16 (*push*) avant d'être renvoyé par l'interface *eth1*.

```
File Sessions Settings Help
[root@vectra17 /root]# route
Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref Use IFace
10.2.0.1 10.5.0.1 255.255.255.255 UGH 0 0 0 eth1
10.1.0.1 10.4.0.1 255.255.255.255 UGH 0 0 0 eth0
10.4.0.0 * 255.255.255.0 U 0 0 0 eth0
10.5.0.0 * 255.255.255.0 U 0 0 0 eth1
127.0.0.0 * 255.0.0.0 U 0 0 0 lo
default * 0.0.0.0 U 0 0 0 eth0
[root@vectra17 /root]# more /proc/net/mpls_*
.....
/proc/net/mpls_fec
.....
40004002 10.1.0.1/32
40004003 10.2.0.1/32
.....
/proc/net/mpls_in
.....
40004000 gen 16 0 PDP DLV
40004400 gen 17 0 PDP FWD(40004003)
40004800 gen 18 0 PDP DLV
40004c00 gen 19 0 PDP FWD(40004002)
.....
/proc/net/mpls_labelspace
.....
lo 0
eth0 0
eth1 0
.....
/proc/net/mpls_out
.....
40004002 PUSH(gen 16) SET(eth0)
40004003 PUSH(gen 16) SET(eth1)
.....
/proc/net/mpls_tunnel
.....
[root@vectra17 /root]#
```

FIGURE 6.18 : Configuration vectra17

```
File Sessions Settings Help
[root@vectra17 /root]# ldp_linux
prompt>add global 10.5.0.2
Adding interface eth0
Adding interface eth1
Adding route 0a020001/32 via 3
Adding route 0a010001/32 via 2
Adding route 0a040000/24 via 2
Adding route 0a050000/24 via 3
Adding route 7f000000/8 via 0
Adding route 00000000/0 via 2
Adding global object with LSRID 0a050002
prompt>add interface eth0
Adding interface eth0
prompt>add interface eth1
Adding interface eth1
prompt>FEC: 0100010a/32
FEC: 0100020a/32
ldp_mpls_in2out_add: 17 <-> 16
PRT: Gratuitous search!!
ldp_mpls_in2out_add: 19 <-> 16
```

FIGURE 6.19 : LDP sur vectra17

6.3.3. Teste et analyse des paquets

Avant l'activation de *LDP*, un analyseur Ethereal est placé entre *LER2* et *LSR1* afin de capturer les paquets échangés. Cette capture de paquets se trouve dans l'annexe *Captures 5*. Les échanges *LDP* et le format des paquets sont identiques à ceux décrits lors du test précédent (voir § 6.2.3). *LER2* indique à *LSR1* qu'il faut utiliser le label 16 pour envoyer un paquet au serveur (paquet 28). De même, *LSR1* indique à son voisin qu'il doit utiliser le label 19 pour atteindre le client (paquet 31).

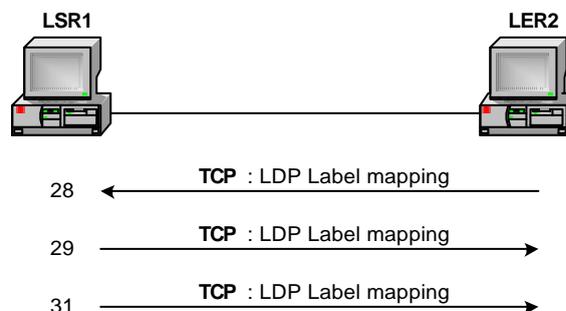


FIGURE 6.20 : Echanges LDP

Par contre, on constate que *LSR1* envoie un autre paquet *label mapping* (paquet 29) dont le *FEC* correspond à l'identificateur *LDP* utilisé pour cette machine (10.5.0.2). Cela veut-il dire que le programme «*ldp-portable-0.060*» attribue un label uniquement aux adresses correspondant aux identificateurs *LDP*? Cela expliquerait pourquoi, sur les *LER*, l'identificateur *LDP* doit correspondre aux adresses du client et du serveur pour que la distribution des labels soit effectuée.

La FIGURE 6.21 représente la configuration obtenue grâce au protocole *LDP*. Les paquets 35 et 36 de la *capture 5* annexée, correspondent à un *ping* effectué depuis le client sur le serveur. On constate que ces paquets sont encapsulés avec les bonnes valeurs de label.

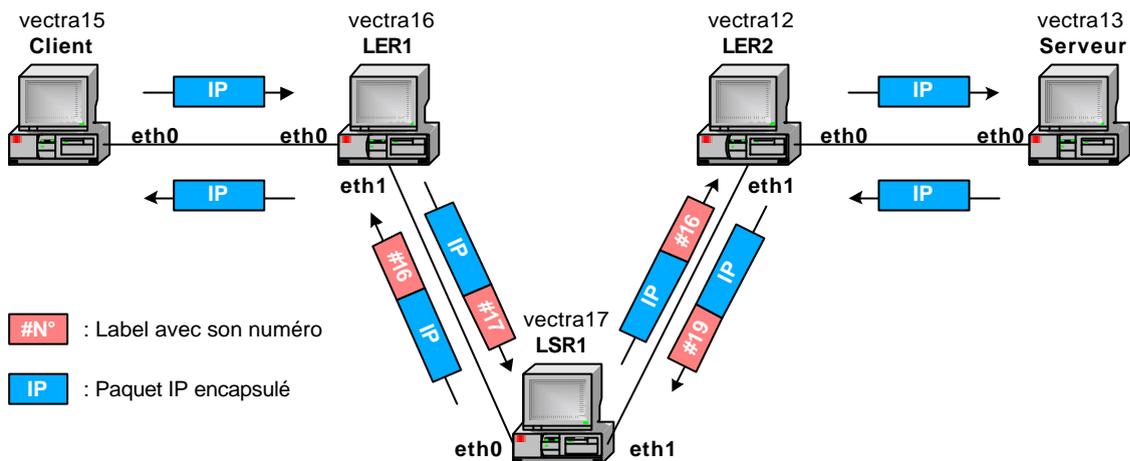


FIGURE 6.21 : Création des LSP par LDP

6.4. Conclusion

Lors des tests décrits ci-dessus, nous avons constaté que pour faire fonctionner la distribution des labels avec le programme «*ldp-portable-0.060*», il faut utiliser, sur les *LER*, les adresses du client et du serveur comme identificateur *LDP*. Cela est certainement un bug de «*ldp-portable-0.060*». Il faut quand même remarquer que ce patch est encore une version expérimentale. La version définitive fonctionnera certainement correctement.

Malgré ces quelques problèmes, le fonctionnement du protocole *LDP* et les différents messages échangés ont pu être étudiés correctement.

7. Mise en place d'un réseau utilisant MPLS avec des routeur Cisco

7.1. Introduction

Les tests qui vont suivre consistent à utiliser le protocole *MPLS* pour effectuer du *traffic engineering* (§ 8) ou créer des *VPN* (§ 9). *MPLS* sur Linux ne permet pas d'effectuer ces tests. En effet, seuls des équipements utilisés dans le *core network* implémentent ces fonctionnalités. Les routeurs Cisco les implémentent à partir de la série 7200. Le laboratoire ne disposant pas de tels équipements, ces tests ont été effectués au laboratoire IPSS de Swisscom à Bümpliz.

Lors de ces tests, des protocoles comme *OSPF* (*Open Shortest Path First*), *RSVP* (*Resources Reservation Protocol*) ou *BGP* (*Border Gateway Protocol*) vont être utilisés. Une fois de plus, ce travail n'as pas pour but l'étude de ces protocoles. Seule leur utilité est expliquée.

Ce chapitre décrit comment configurer les routeurs afin de préparer le réseau de tests. Ce réseau est constitué de cinq routeurs Cisco 7200 et de deux PCs connectés selon la FIGURE 7.1. Toutes les interfaces des routeurs sont de type *fastethernet*. La version d'*IOS* utilisée dans les routeurs est la « c7200-p-mz.120-18.ST.BIN ».

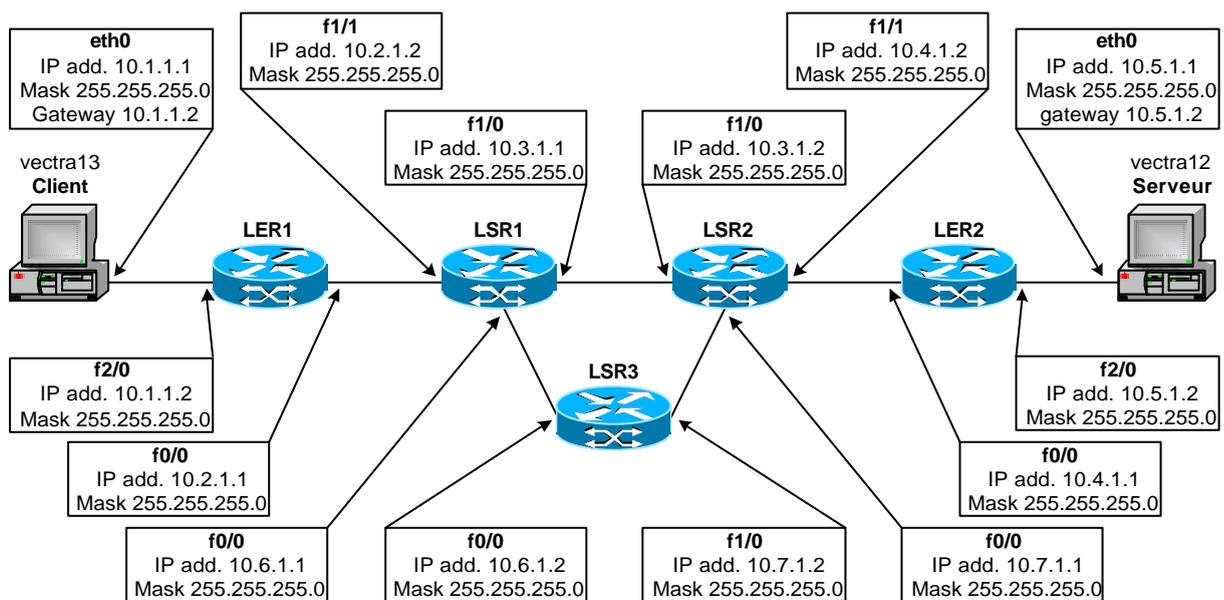


FIGURE 7.1 : Topologie du réseau de tests

7.2. Mise en œuvre d'*OSPF*

Le protocole *OSPF* (*Open Shortest Path First*) est utilisé sur les routeurs afin de créer dynamiquement les tables de routage IP. Grâce à ce protocole, le routeur connaît la topologie du réseau et utilise le chemin le plus direct pour envoyer des paquets IP vers n'importe quelle destination de ce réseau. Ce qui suit explique comment activer ce protocole sur les routeurs.

Pour commencer, il faut configurer les interfaces *fastethernet* selon la FIGURE 7.1. Les commandes sont identiques à celles utilisées pour la configuration du routeur Cisco 2600 (voir partie *RAS*).

- Pour *LER1* :

- router(config)# hostname LER1
- LER1(config)# interface f2/0
- LER1(config-int)# **ip address 10.1.1.2 255.255.255.0**
- LER1(config-int)# **no shutdown**
- LER1(config)# interface f0/0
- LER1(config-int)# **ip address 10.2.1.1 255.255.255.0**
- LER1(config-int)# **no shutdown**

La commande suivante active le *process OSPF* sur le routeur. Il faut donner un numéro d'identification à ce *process*, car plusieurs *process OSPF* peuvent être activés sur un même routeur.

router ospf process-id

- *Process-id* : Numéro d'identification du *process OSPF*. Ce numéro est local au routeur.
- LER1(config)# **router ospf 10**

La commande suivante permet de définir la zone du réseau couverte par *OSPF*. Dans notre cas, cette zone représente l'ensemble du réseau de tests.

network address wildcard-mask area area-id

- *address* : Adresse IP indiquant la plage d'adresse recherchée par *OSPF*.
- *wildcard-mask* : Mask de la plage d'adresse.
- *area-id* : Identificateur de zone associé à la plage d'adresses *OSPF*. Cet identificateur doit être commun à tous les routeurs d'une même zone.
- LER1(config-router)# **network 10.0.0.0 0.255.255.255 area 0**

Tous les routeurs du réseau doivent être configurés de la même manière.

- Pour *LSR1* :

- router(config)# hostname LSR1
- LSR1(config)# interface f1/1
- LSR1(config-int)# **ip address 10.2.1.2 255.255.255.0**
- LSR1(config-int)# **no shutdown**
- LSR1(config)# interface f1/0
- LSR1(config-int)# **ip address 10.3.1.1 255.255.255.0**
- LSR1(config-int)# **no shutdown**
- LSR1(config)# interface f0/0
- LSR1(config-int)# **ip address 10.6.1.1 255.255.255.0**
- LSR1(config-int)# **no shutdown**
- LSR1(config)# **router ospf 10**
- LSR1(config-router)# **network 10.0.0.0 0.255.255.255 area 0**

- Pour *LSR2* :

- router(config)# hostname LSR2
- LSR2(config)# interface f1/0
- LSR2(config-int)# **ip address 10.3.1.2 255.255.255.0**
- LSR2(config-int)# **no shutdown**
- LSR2(config)# interface f1/1
- LSR2(config-int)# **ip address 10.4.1.2 255.255.255.0**
- LSR2(config-int)# **no shutdown**
- LSR2(config)# interface f0/0
- LSR2(config-int)# **ip address 10.7.1.1 255.255.255.0**
- LSR2(config-int)# **no shutdown**
- LSR2(config)# **router ospf 10**
- LSR2(config-router)# **network 10.0.0.0 0.255.255.255 area 0**

- Pour *LSR3* :

- router(config)# hostname LSR3
- LSR3(config)# interface f0/0
- LSR3(config-int)# **ip address 10.6.1.2 255.255.255.0**
- LSR3(config-int)# **no shutdown**
- LSR3(config)# interface f1/0
- LSR3(config-int)# **ip address 10.7.1.2 255.255.255.0**
- LSR3(config-int)# **no shutdown**
- LSR3(config)# **router ospf 10**
- LSR3(config-router)# **network 10.0.0.0 0.255.255.255 area 0**

- Pour *LER2* :

- router(config)# hostname LER2
- LER2(config)# interface f0/0
- LER2(config-int)# **ip address 10.4.1.1 255.255.255.0**
- LER2(config-int)# **no shutdown**
- LER2(config)# interface f2/1
- LER2(config-int)# **ip address 10.5.1.2 255.255.255.0**
- LER2(config-int)# **no shutdown**
- LER2(config)# **router ospf 10**
- LER2(config-router)# **network 10.0.0.0 0.255.255.255 area 0**

7.3. Activation d'*MPLS* sur les routeurs

La mise en oeuvre du protocole *MPLS* sur les routeurs Cisco est très simple. La création des tables de *forwarding* est faite dynamiquement grâce au protocole de distribution de label.

Pour pouvoir utiliser *MPLS* sur les routeurs Cisco, il faut d'abord activer le *Cisco Express Forwarding (CEF)*.

- router(config)# **ip cef**

La commande suivante active *MPLS* et démarre le processus de distribution des labels.

```
➤ router(config)# tag-switching advertise-tags
```

La commande suivante permet de sélectionner le protocole utilisé pour la distribution des labels. Par défaut, c'est le protocole propriétaire Cisco qui est utilisé (*TDP : Tag Distribution Protocol*). Pour notre réseau, nous utilisons *LDP*.

```
mpls label protocol { LDP | TDP }
```

```
➤ router(config)# mpls label protocol ldp
```

Les interfaces doivent encapsuler les paquets avant de les envoyer sur le réseau *MPLS*. Pour cela, la commande suivante doit être ajoutée dans la configuration de chaque interface. Cette configuration n'est pas utilisée pour les interfaces des *LER* connectées au PCs, car ces machines utilisent des paquets IP standards.

```
➤ router(config-int)# mpls ip
```

7.4. Tests du réseau

On peut vérifier que les tables de routage des routeurs sont bien remplies par *OSPF*, en utilisant la commande *show ip route*. La FIGURE 7.2 correspond à la table de routage de *LSR1*. Les routes précédées de la lettre « O » ont été ajoutées par *OSPF*.

```
LSR1#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR

Gateway of last resort is not set

    10.0.0.0/24 is subnetted, 7 subnets
C       10.3.1.0 is directly connected, FastEthernet1/0
C       10.2.1.0 is directly connected, FastEthernet1/1
O       10.1.1.0 [110/2] via 10.2.1.1, 00:01:39, FastEthernet1/1
O       10.7.1.0 [110/2] via 10.3.1.2, 00:01:39, FastEthernet1/0
        [110/2] via 10.6.1.2, 00:01:39, FastEthernet0/0
C       10.6.1.0 is directly connected, FastEthernet0/0
O       10.5.1.0 [110/3] via 10.3.1.2, 00:01:39, FastEthernet1/0
O       10.4.1.0 [110/2] via 10.3.1.2, 00:01:39, FastEthernet1/0
```

FIGURE 7.2 : Table de routage de LSR1

De même, la commande *show mpls forwarding-table* permet de voir le *LFIB* de *LSR1* constitué dynamiquement grâce au protocole *LDP*.

```
LSR1#show mpls forwarding-table
Local  Outgoing  Prefix          Bytes tag  Outgoing  Next Hop
tag    tag or VC  or Tunnel Id   switched  interface
16     Pop tag    10.1.1.0/24    9521      Fa1/1     10.2.1.1
17     Pop tag    10.7.1.0/24    0         Fa1/0     10.3.1.2
18     Pop tag    10.7.1.0/24    0         Fa0/0     10.6.1.2
19     19         10.5.1.0/24    768      Fa1/0     10.3.1.2
19     Pop tag    10.4.1.0/24    2106     Fa1/0     10.3.1.2
```

FIGURE 7.3 : Table de forwarding de LSR1 (LFIB)

Les captures des tables de routage et de *forwarding* des autres routeurs se trouvent dans l'annexe *Captures des tables des routeurs Cisco 1*.

La figure ci-dessous montre la correspondance des *LFIB* pour un chemin donné. Le label *Local* correspond au label attribué à chaque préfixe (*FEC*) de la table de routage par le routeur. Ce label local est transmis, grâce au message *LDP label mapping*, aux autres routeurs afin qu'ils puissent compléter leur champ *Outgoing label*. Ainsi chaque routeur connaît le label à utiliser pour *forwarder* un label.

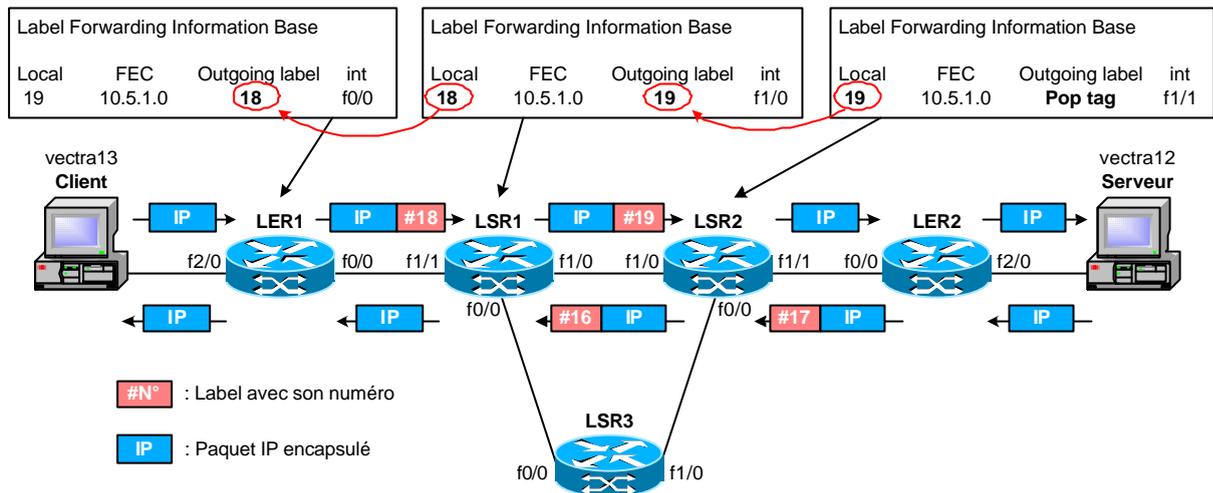


FIGURE 7.4 : Chemin emprunté par les paquets lors d'un ping

Afin de vérifier le bon fonctionnement du réseau, un *ping* est effectué, depuis le client, sur le serveur (FIGURE 7.4). Cela permet de constater les choses suivantes :

- Les paquets envoyés sur le réseau *MPLS* sont bien labellisés.
- Les paquets empruntent le chemin le plus direct vers la destination.
- Les labels sont déjà retirés des paquets par le dernier *LSR* se trouvant avant le *LER*.

L'opération, décrite dans cette dernière constatation, se nomme *Penultimate Hop Poping (PHP)*. Elle permet un gain de performance dans les *LER*. En effet, lorsque *PHP* n'est pas utilisé, le *LER* doit effectuer deux opérations pour *forwarder* les paquets. La première est la consultation de la table de *forwarding* qui indique que le label doit être retiré du paquet et l'entête IP consulté. La deuxième est la recherche de l'adresse IP du paquet dans la table de routage, pour l'envoyer vers sa destination. Avec *PHP*, le *LSR*, se trouvant avant le *LER*, *forward* les paquets sans ajouter de nouveau label. Ainsi, le *LER* n'a plus qu'à effectuer le *forwarding* IP. L'utilisation de *PHP* est demandée, par le *LER*, en utilisant une valeur de label de 3, lors de l'envoi du message *LDP label mapping*.

8. Traffic engineering sur le réseau MPLS

8.1. Introduction

Le but des tests présenté ci-dessous est de mettre en œuvre certains mécanismes offerts par *MPLS* pour effectuer du *traffic engineering* sur un réseau. Les scénarios étudiés sont :

- Création d'un *path* statique pour répartir la charge sur un réseau.
- Création d'un *backup path* pour sécuriser un *path*.
- Création d'un *path* dynamique avec contrainte.

Ces scénarios sont implémentés sur le réseau décrit dans le chapitre 7.

8.2. Création d'un *path* statique

8.2.1. Objectif

Le but de ce scénario est de définir explicitement un chemin (*path*) reliant *LER1* à *LER2* en passant par *LSR3* (FIGURE 8.1). Comme nous l'avons vu au paragraphe 7.4, un paquet allant de *LER1* à *LER2* emprunte le chemin le plus direct, c'est-à-dire qu'il ne passe pas par *LSR3*. Sur un réseau réel, les chemins les plus courts sont parfois congestionnés alors que d'autres sont sous-utilisés. *MPLS* permet de répartir la charge sur le réseau en obligeant certains flux à emprunter les chemins moins utilisés.

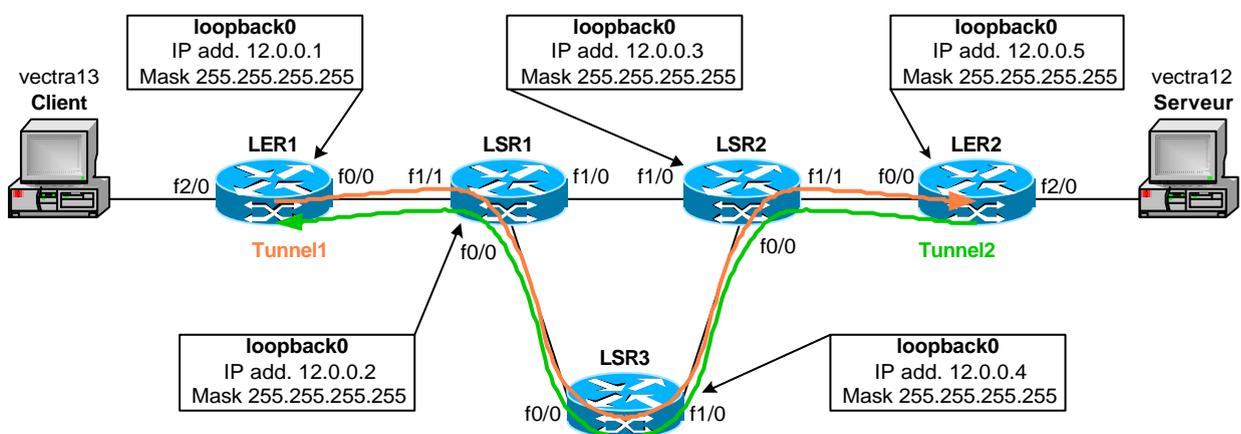


FIGURE 8.1 : Création d'un *path* passant par *LSR3*

8.2.2. Autoriser le *traffic engineering* sur les routeurs

Pour commencer, il faut configurer tous les routeurs afin qu'ils supportent le *traffic engineering*. Pour cela, il faut procéder comme suit :

La commande *mpls traffic-eng tunnels* active la fonction tunnel du routeur.

```
➤ router(config)# mpls traffic-eng tunnels
```

Afin que chaque routeur puisse être identifié par une adresse IP unique, nous utilisons une interface virtuelle appelée *loopback 0*. L'adresse IP de cette interface se configure de la même manière qu'une interface physique classique. Cette adresse IP est utilisée comme *host address* du routeur et doit donc posséder un masque de 32 bits.

- `router(config)# interface loopback 0`
- `router(config-int)# ip address 12.0.0.x 255.255.255.255`

Voici les adresses IP utilisées pour chaque routeur :

- *LER1* : 12.0.0.1
- *LSR1* : 12.0.0.2
- *LSR2* : 12.0.0.4
- *LSR3* : 12.0.0.3
- *LER2* : 12.0.0.5

Pour faire du *traffic engineering*, le routeur doit connaître la topologie du réseau. Pour cela, on peut lui dire d'utiliser *OSPF* grâce à la commande *mpls traffic-eng area*. L'*area* est toujours la même que celle configurée dans le paragraphe 7.2.

- `router(config)# router ospf 10`
- `router(config-router)# mpls traffic-eng area 0`

La commande suivante permet de configurer l'identificateur du routeur utilisé pour le *traffic engineering*. Nous identifions les routeurs avec leur adresse de *loopback*.

- `router(config-router)# mpls traffic-eng router-id loopback 0`

Pour que *OSPF* ajoute les adresses d'identifications des routeurs dans les tables, il faut ajouter la plage d'adresse 12.0.0.0 dans l'*area 0*.

- `router(config-router)# network 12.0.0.0 0.0.0.255 area 0`

Pour autoriser les tunnels sur une interface, la commande *mpls traffic-eng tunnels* doit être ajoutée dans la configuration de cette interface. Chaque interface utilisée par le tunnel doit être configurée ainsi.

- `router(config-int)# mpls traffic-eng tunnels`

8.2.3. Création des paths

Un *path* est unidirectionnel. Il faut donc créer un tunnel pour le chemin aller et un autre pour le chemin retour (FIGURE 8.1). Un tunnel est défini sur le routeur se trouvant à son entrée. Dans notre cas, cela correspond aux deux *LER*.

- Configuration du tunnel 1 sur *LER1* :

Le routeur voit un tunnel comme une interface. Il suffit de taper la commande suivante pour le créer et entrer dans son mode de configuration :

interface tunnel *id*

- *id* : numéro permettant d'identifier le tunnel sur le réseau.

➤ LER1(config)# **interface tunnel 1**

Il faut configurer l'adresse IP de cette interface. Un tunnel *MPLS* doit être *unnumbered* car il représente une connexion unidirectionnelle. On l'associe à l'identificateur du routeur.

➤ LER1(config-int)# **ip unnumbered loopback 0**

Ensuite, il faut définir la destination du tunnel. Dans notre cas, cette destination est *LER2* (12.0.0.5).

➤ LER1(config-int)# **tunnel destination 12.0.0.5**

La commande suivante indique que le mode d'encapsulation des paquets utilisé pour le tunnel est *MPLS*.

➤ LER1(config-int)# **tunnel mode mpls traffic-eng**

Le tunnel doit être signalé aux *IGP* afin qu'il soit pris en compte lors de la création des tables de routage. Pour cela, il faut utiliser la commande suivante :

➤ LER1(config-int)# **tunnel mpls traffic-eng autoroute announce**

La commande ci-dessous indique au tunnel qu'il doit suivre la route explicitée par l'identificateur 1.

➤ LER1(config-int)# **tunnel mpls traffic-eng path-option 1 explicit identifier 1**

Ci-dessous est définie la route explicitée par *idntifier 1*. Cette route est indiquée en faisant la liste des routeur devant être traversés par le tunnel. Selon la FIGURE 8.1, le tunnel 1 doit passer par *LSR1*, *LSR3*, *LSR2* pour finir à *LER2*.

➤ LER1(config)# **ip explicit-path identifier 1**
➤ LER1(cfg-ip-expl-path)# **next-address 12.0.0.2**
➤ LER1(cfg-ip-expl-path)# **next-address 12.0.0.4**
➤ LER1(cfg-ip-expl-path)# **next-address 12.0.0.3**
➤ LER1(cfg-ip-expl-path)# **next-address 12.0.0.5**

- Configuration du tunnel 2 sur *LER2* :

La configuration du second tunnel sur *LER2* se fait de la même manière que celle décrite ci-dessus :

- LER2(config)# **interface tunnel 2**
- LER2(config-int)# **ip unnumbered loopback 0**
- LER2(config-int)# **tunnel destination 12.0.0.1**
- LER2(config-int)# **tunnel mode mpls traffic-eng**
- LER2(config-int)# **tunnel mpls traffic-eng autoroute announce**
- LER2(config-int)# **tunnel mpls traffic-eng path-option 1 explicit identifier 1**
- LER2(config)# **ip explicit-path identifier 1**
- LER2(cfg-ip-expl-path)# **next-address 12.0.0.3**
- LER2(cfg-ip-expl-path)# **next-address 12.0.0.4**
- LER2(cfg-ip-expl-path)# **next-address 12.0.0.2**
- LER2(cfg-ip-expl-path)# **next-address 12.0.0.1**

Le listing de configuration de chaque routeur se trouve dans l'annexe *Listings de configuration 1*

Remarque : Pour que la route explicitée soit valide, il faut parfois *reloader* les routeurs.

8.2.4. Test et analyse

Le protocole *MPLS* propose deux protocoles permettant d'obtenir des labels, afin de créer des chemins avec contraintes. Le premier protocole est une extension de *LDP* nommée *CR-LDP* (*Constraint-based Routing Label Distribution Protocol*). Le second protocole est *RSVP* (*Resource Reservation Protocol*). Les routeurs Cisco implémentent uniquement cette deuxième solution. Comme nous l'avons dit plus haut, le but de ce travail n'est pas d'étudier ce protocole *RSVP*. Toutefois, afin de comprendre comment les labels utilisés pour le tunnel sont distribués nous illustrons le principe utilisé par ce protocole :

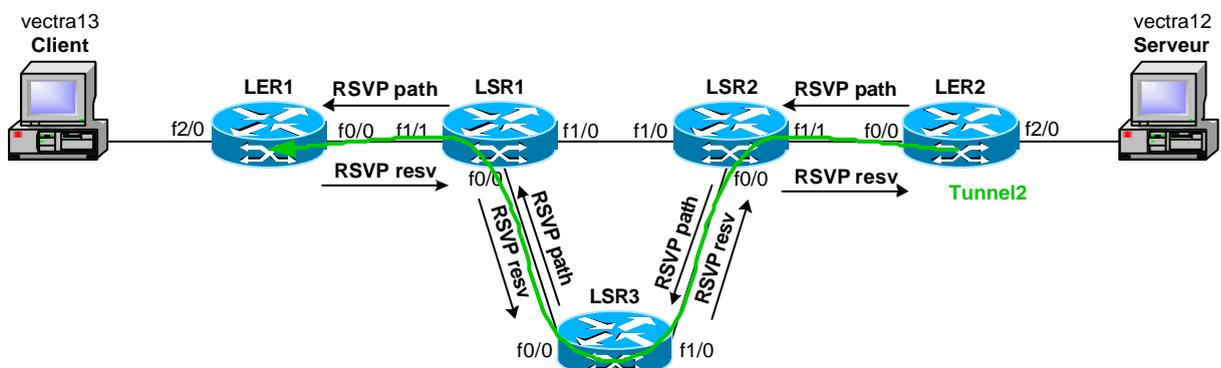


FIGURE 8.2 : Création du path par RSVP

La figure ci-dessus montre comment le tunnel 2 est réalisé. *LER2* envoie un paquet *RSVP path* (FIGURE 8.3) contenant toutes les adresses à une interface utilisée par le tunnel (champ **EXPLICIT ROUTE**). Ce paquet traverse tous les routeurs concernés en

leur indiquant qu'un label doit être retourné (champ **LABEL REQUEST**) pour ce tunnel (champ **Extended Tunnel ID**). Chaque routeur répond à ce paquet en envoyant, au routeur le précédent, un message *RSVP resv* (FIGURE 8.4). Ce message contient le label à utiliser pour encapsuler les paquets devant circuler dans le tunnel (champ **LABEL**).

```

Resource ReserVation Protocol (RSVP)
  RSVP Header
    RSVP Version: 1
    Flags: 00
    Message Type: PATH Message (1)
    Message Checksum: 0x9086 (correct)
    Sending TTL: 254
    Message length: 232
  SESSION: IPv4-LSP, 12.0.0.1, 2, c000005
    Length: 16
    Class number: 1 - SESSION object
    C-type: 7 - IPv4 LSP
    Destination address: 12.0.0.1 (12.0.0.1)
    Tunnel ID: 2
    Extended Tunnel ID: 201326597 (12.0.0.5)
  HOP: IPv4, 10.4.1.1
  TIME VALUES: 30000 ms
  EXPLICIT ROUTE
    Length: 68
    Class number: 20 - EXPLICIT ROUTE object
    C-type: 1
    IPv4 Subobject - 10.4.1.2, Strict
    IPv4 Subobject - 10.7.1.1, Strict
    IPv4 Subobject - 10.7.1.2, Strict
    IPv4 Subobject - 10.6.1.2, Strict
    IPv4 Subobject - 10.6.1.1, Strict
    IPv4 Subobject - 10.2.1.2, Strict
    IPv4 Subobject - 10.2.1.1, Strict
    IPv4 Subobject - 12.0.0.1, Strict
  LABEL REQUEST: IP (0x0800)
    Length: 8
    Class number: 19 - LABEL REQUEST object
    C-type: 1
    L3PID: IP (0x0800)
  SESSION ATTRIBUTE
  SENDER TEMPLATE: IPv4-LSP, 12.0.0.5, 6
  SENDER TSPEC: IntServ, 0 bytes/sec
  ADSPEC

```

FIGURE 8.3 : Paquet RSVP contenant le path

```

Resource ReserVation Protocol (RSVP)
  RSVP Header
    RSVP Version: 1
    Flags: 00
    Message Type: RESV Message (2)
    Message Checksum: 0xcf96 (correct)
    Sending TTL: 255
    Message length: 108
  SESSION: IPv4-LSP, 12.0.0.1, 2, c000005
    Length: 16
    Class number: 1 - SESSION object
    C-type: 7 - IPv4 LSP
    Destination address: 12.0.0.1 (12.0.0.1)
    Tunnel ID: 2
    Extended Tunnel ID: 201326597 (12.0.0.5)
  HOP: IPv4, 10.4.1.2
  TIME VALUES: 30000 ms
  STYLE: Shared-Explicit (18)
  FLOWSPEC: Controlled-Load, 0 bytes/sec
  FILTERSPEC: IPv4-LSP, 12.0.0.5, 6
  LABEL: 25
    Length: 8
    Class number: 16 - LABEL object
    C-type: 1
    Label: 25

```

FIGURE 8.4 : Paquet RSVP retournant le label

En regardant la table de routage de *LER1*, on constate que les paquets à destination de *LER2* sont bien envoyés par le tunnel 1.

```

10.0.0.0/24 is subnetted, 7 subnets
O   10.3.1.0 [110/2] via 10.2.1.2, 00:04:05, FastEthernet0/0
C   10.2.1.0 is directly connected, FastEthernet0/0
C   10.1.1.0 is directly connected, FastEthernet2/0
O   10.7.1.0 [110/3] via 10.2.1.2, 00:04:05, FastEthernet0/0
O   10.6.1.0 [110/2] via 10.2.1.2, 00:04:05, FastEthernet0/0
O   10.5.1.0 [110/4] via 0.0.0.0, 00:04:05, Tunnel1
O   10.4.1.0 [110/3] via 10.2.1.2, 00:04:05, FastEthernet0/0

```

FIGURE 8.5 : Table de routage de *LER1*

De même, dans la table de *forwarding* (*LFIB*), on constate que pour le *FEC* 10.5.1.0, le champ *Outgoing tag* ne contient pas de label, mais pointe sur le tunnel à utiliser.

Local tag	Outgoing tag or VC	Prefix or Tunnel Id	Bytes switched	tag	Outgoing interface	Next Hop
16	Pop tag	10.3.1.0/24	0		Fa0/0	10.2.1.2
17	16	10.4.1.0/24	0		Fa0/0	10.2.1.2
18	Untagged[T]	10.5.1.0/24	0	0	Tu1	point2point
19	Pop tag	10.6.1.0/24	0		Fa0/0	10.2.1.2
20	19	10.7.1.0/24	0		Fa0/0	10.2.1.2
[T]	Forwarding through a TSP tunnel.					

FIGURE 8.6 : Table de *forwarding* de *LER1*

La commande *show mpls traffic-eng tunnels* permet de voir les informations sur les tunnels présents sur le routeur. Voici ce qui est affiché pour le tunnel 1 sur *LER1*. On constate qu'il est bien du type explicite et traverse les interfaces correspondant à la route définie à la FIGURE 8.1. Le label à utiliser pour envoyer les paquets à travers le tunnel est défini dans le champ *OutLabel* de cette table. Ce label est le 25.

LER1#show mpls traffic-eng tunnels

```

Name: LER1_t1 (Tunnel1) Destination: 12.0.0.5
Status:
  Admin: up      Oper: up      Path: valid      Signalling: connected
  path option 1, type explicit 1 (Basis for Setup, path weight 4)
Config Parameters:
  Bandwidth: 0 kbps (Global) Priority: 7 7 Affinity: 0x0/0xFFFF
  Metric Type: TE (default)
  AutoRoute: enabled LockDown: disabled Loadshare: 0 bw-based
  auto-bw: disabled(0/67) 0 Bandwidth Requested: 0
InLabel : -
OutLabel : FastEthernet0/0, 25
RSVP Signalling Info:
  Src 12.0.0.1, Dst 12.0.0.5, Tun_Id 1, Tun_Instance 15
RSVP Path Info:
  My Address: 10.2.1.1
  Explicit Route: 10.2.1.2 10.6.1.1 10.6.1.2 10.7.1.2
                  10.7.1.1 10.4.1.2 10.4.1.1 12.0.0.5
.
.

```

FIGURE 8.7 : Extrait de la table des tunnels de *LER1*

Les captures des diverses tables des autres routeurs se trouvent dans l'annexe *Captures des tables des routeurs Cisco 2*.

En effectuant un ping depuis le client sur le serveur, on constate que les paquets empruntent bien les tunnels et non pas le chemin le plus direct comme mesuré au paragraphe 7.4. Ces paquets sont représentés sur la FIGURE 8.8. Les tables représentées ci-dessous contiennent uniquement les informations pour le tunnel 1. La totalité de ces tables se trouvent dans l'annexe *Captures des tables des routeurs Cisco 2*. On voit bien qu'une fois que le paquet est encapsulé avec le label 25, il suit le chemin explicité, car les labels, pour ce tunnel, ont été ajoutés dans les tables de *forwarding* par *RSVP*.

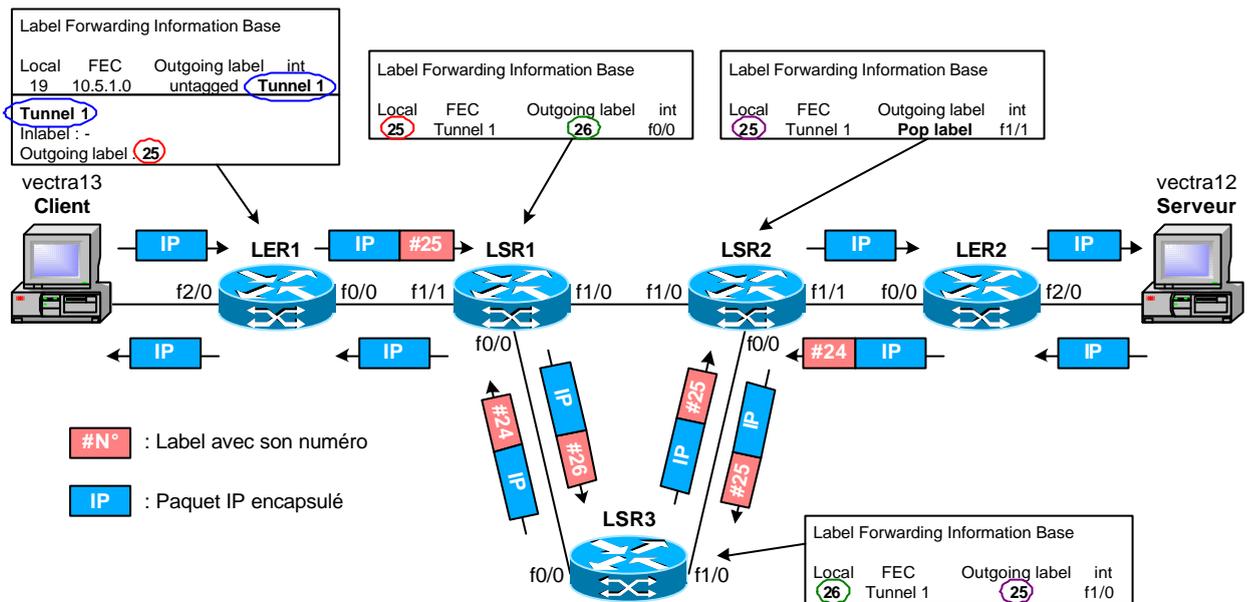


FIGURE 8.8 : Paquet empruntant les tunnels

8.3. Création d'un *backup path*

8.3.1. Objectif

Le but de ce scénario est de protéger des chemins définis statiquement (tunnel 1 et tunnel 2) contre une coupure de connexion (FIGURE 8.9). Dans ce test, c'est la liaison entre *LSR1* et *LSR2* qui est protégée. Pour cela, nous créons deux tunnels de réserve (*backup path*) qui permettent de relier ces deux routeurs en passant par *LSR3*. Lorsque la liaison entre *LSR1* et *LSR2* est coupée, les paquets sont envoyés sur les *backup path* (tunnel 10 et tunnel 20) afin de contourner la ligne défectueuse. Lorsqu'ils détectent la coupure, la décision de rerouter les paquets dans les *backup path* est directement prise par *LSR1*, pour les paquets à destination du serveur, et par *LSR2*, pour les paquets à destination du client. L'avantage de ce système est la rapidité de reroutage des paquets (*fast reroute*).

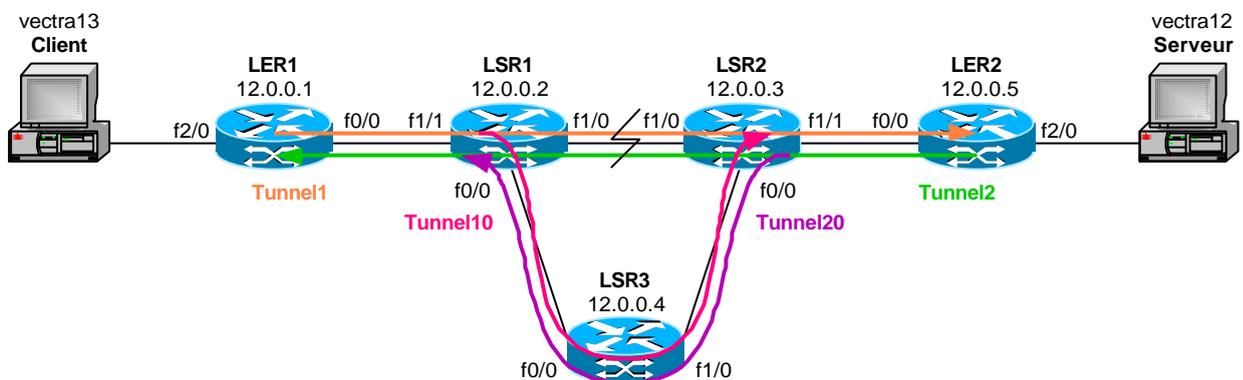


FIGURE 8.9 : Tunnels avec *backup path*

8.3.2. Création des path

La création des deux chemins reliant *LER1* et *LER2* (tunnel 1 et tunnel 2) est faite de la même manière que celle décrite dans le test précédent (voir § 8.2). Les changements à apporter à cette configuration sont indiqués ci-dessous.

- Configuration du tunnel 1 sur *LER1* :

La route explicitée pour le tunnel 1 doit être modifiée pour n'emprunter que *LSR1*, *LSR2* et *LER2*.

- LER1(config)# **ip explicit-path identifier 1**
- LER1(cfg-ip-expl-path)# **next-address 12.0.0.2**
- LER1(cfg-ip-expl-path)# **next-address 12.0.0.3**
- LER1(cfg-ip-expl-path)# **next-address 12.0.0.5**

En ajoutant la commande suivante dans l'interface tunnel 1, on autorise ce tunnel à utiliser un *backup path*, si son chemin « normal » est interrompu :

- LER1(config)# **interface tunnel 1**
- LER1(config-int)# **tunnel mpls traffic-eng fast-reroute**

- Configuration du tunnel 2 sur *LER2* :

Les mêmes changements sont effectués pour le tunnel 2.

- LER2(config)# **ip explicit-path identifier 1**
- LER2(cfg-ip-expl-path)# **next-address 12.0.0.3**
- LER2(cfg-ip-expl-path)# **next-address 12.0.0.2**
- LER2(cfg-ip-expl-path)# **next-address 12.0.0.1**

- LER2(config)# **interface tunnel 2**
- LER2(config-int)# **tunnel mpls traffic-eng fast-reroute**

8.3.3. Création des backup-path

A présent, il faut créer les tunnels de *backup* (tunnel 10 et tunnel 20) reliant *LSR1* à *LSR2* en passant par *LSR3*. Ces tunnels sont créés de la même manière que précédemment.

- Configuration du tunnel 10 sur *LSR1* :

Voici comment créer le tunnel 10 :

- LSR1(config)# **interface tunnel 10**
- LSR1(config-int)# **ip unnumbered loopback 0**
- LSR1(config-int)# **tunnel destination 12.0.0.3**
- LSR1(config-int)# **tunnel mode mpls traffic-eng**
- LSR1(config-int)# **tunnel mpls traffic-eng path-option 1 explicit identifier 1**

- LSR1(config)# **ip explicit-path identifier 1**
- LSR1(cfg-ip-expl-path)# **next-address 12.0.0.4**
- LSR1(cfg-ip-expl-path)# **next-address 12.0.0.3**

Ensuite, il faut configurer l'interface *fastethernet1/0* pour qu'elle envoie les paquets dans le tunnel de *backup* (tunnel 10), si la connexion avec *LSR2* est interrompue.

- LSR1(config)# **interface f1/0**
- LSR1(config-int)# **mpls traffic-eng tunnels**
- LSR1(config-int)# **mpls traffic-eng backup-path tunnel 10**

- Configuration du tunnel 20 sur *LER2* :

La configuration du tunnel 20 sur *LER2* se fait de la même manière que celle décrite ci-dessus.

- LSR2(config)# **interface tunnel 20**
- LSR2(config-int)# **ip unnumbered loopback 0**
- LSR2(config-int)# **tunnel destination 12.0.0.2**
- LSR2(config-int)# **tunnel mode mpls traffic-eng**
- LSR2(config-int)# **tunnel mpls traffic-eng path-option 1 explicit identifier 1**

- LSR2(config)# **ip explicit-path identifier 1**
- LSR2(cfg-ip-expl-path)# **next-address 12.0.0.4**
- LSR2(cfg-ip-expl-path)# **next-address 12.0.0.2**

Il ne reste plus qu'à configurer l'interface *fastethernet1/0* pour rerouter les paquets dans ce tunnel si une coupure de connexion avec *LSR1* est détectée.

- LSR2(config)# **interface f1/0**
- LSR2(config-int)# **mpls traffic-eng tunnels**
- LSR2(config-int)# **mpls traffic-eng backup-path tunnel 20**

Le listing de configuration de chaque routeur se trouve dans l'annexe *Listings de configuration 2*.

8.3.4. Test et analyse

La distribution des labels, pour chaque tunnel, est faite par le protocole *RSVP* selon le principe illustré à la FIGURE 8.2. Les captures des diverses tables des routeurs se trouvent dans l'annexe *Captures des tables des routeurs Cisco 3*.

En regardant la table des tunnels de *LSR1* (FIGURE 8.10), on voit les caractéristiques du tunnel 10 créé par ce routeur. On constate qu'il traverse bien les interfaces correspondant à la route définie à la FIGURE 8.9. Le label à utiliser pour envoyer les paquets à travers le tunnel est le 25. On voit aussi les caractéristiques du tunnel 1. C'est dans le champ *FRR OutLabel* que le tunnel 10 est défini comme *backup path*.

```
LSR1#sh mpls traffic-eng tunnels
```

```

Name: LSR1_t10                               (Tunnel10) Destination: 12.0.0.3
Status:
  Admin: up           Oper: up           Path: valid           Signalling: connected

  path option 1,(LOCKDOWN) type explicit 1 (Basis for Setup, path weight 2)

Config Parameters:
  Bandwidth: 0          kbps (Global) Priority: 7 7  Affinity: 0x0/0xFFFF
  Metric Type: TE (default)
  AutoRoute: disabled LockDown: enabled  Loadshare: 0          bw-based
  auto-bw: disabled(0/93) 0  Bandwidth Requested: 0

InLabel  : -
OutLabel : FastEthernet0/0, 25
RSVP Signalling Info:
  Src 12.0.0.2, Dst 12.0.0.3, Tun_Id 10, Tun_Instance 11
RSVP Path Info:
  My Address: 10.6.1.1
  Explicit Route: 10.6.1.2 10.7.1.2 10.7.1.1 12.0.0.3
.
.

LSP Tunnel LSR1_t1 is signalled, connection is up
InLabel  : FastEthernet1/1, 24
OutLabel : FastEthernet1/0, 24
FRR OutLabel : Tunnel10, 24
RSVP Signalling Info:
  Src 12.0.0.1, Dst 12.0.0.5, Tun_Id 1, Tun_Instance 8
RSVP Path Info:
  My Address: 10.3.1.1
  Explicit Route: 10.3.1.2 10.4.1.2 10.4.1.1 12.0.0.5
.
.

```

FIGURE 8.10 : Extrait de la table des tunnels de *LSR1*

Lorsque la connexion entre *LSR1* et *LSR2* est active, *LSR1* utilise le label 24 pour *forwarder* les paquets, dans le tunnel 1, vers *LSR2*. Si cette connexion est défectueuse, les paquets sont normalement encapsulés avec le label 24, puis envoyés dans le tunnel 10. Le label correspondant à ce tunnel est le 25. Ce label est ajouté au paquet avant de le *forwarder* dans ce tunnel (FIGURE 8.11). Le paquet possède donc une pile de labels (*label stack*). La FIGURE 8.12 montre un des ces paquets avec ses deux labels. *LSR3*, étant l'avant dernier routeur du tunnel 10, retire le label (*PHP*) se trouvant au sommet de la pile, puis le *forward* vers *LSR2*. Le label 24 étant toujours présent sur le paquet, *LSR2* sait, grâce à sa table de *forwarding*, que le paquet correspond au tunnel 1 et doit être *forwardé* vers *LER2*. Grâce à ce mécanisme, le *forwarding* des paquets dans le tunnel 1 est préservé de bout en bout.

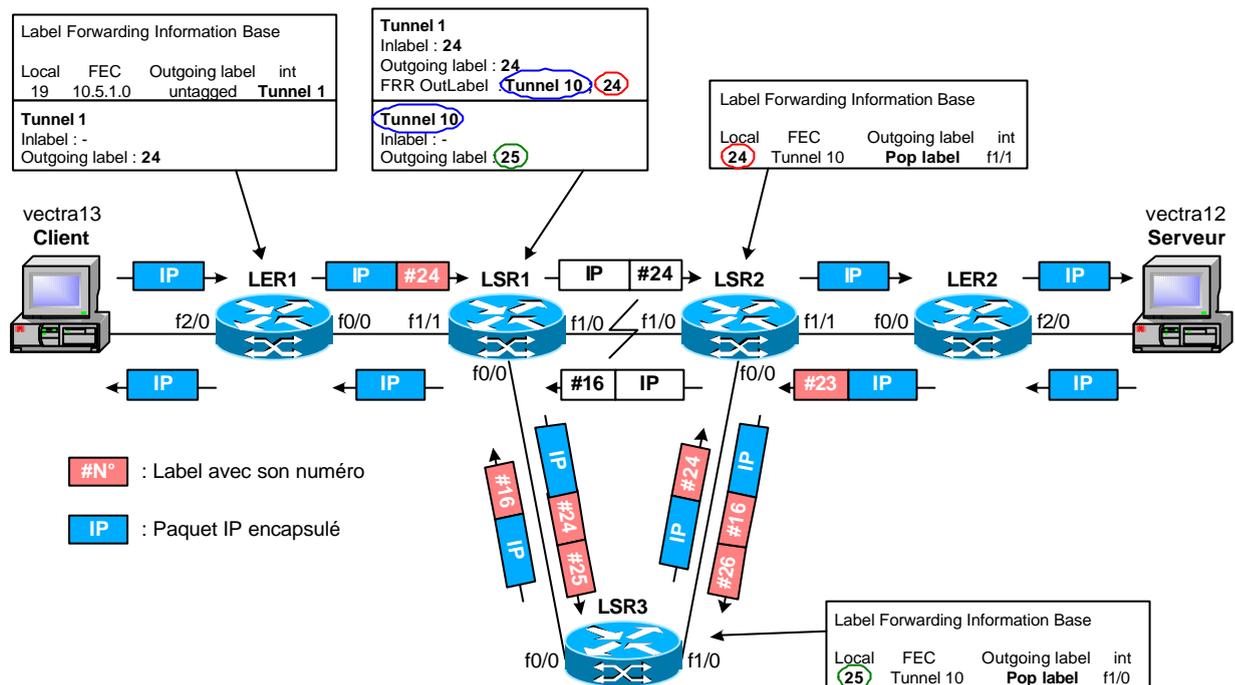


FIGURE 8.11 : Paquet empruntant le backup path

```

Frame 3 (82 on wire, 82 captured)
Ethernet II
MultiProtocol Label Switching Header
  MPLS Label: Unknown (25)      (label pour tunnel 10)
  MPLS Experimental Bits: 0
  MPLS Bottom Of Label Stack: 0
  MPLS TTL: 126
MultiProtocol Label Switching Header
  MPLS Label: Unknown (24)      (label pour tunnel 1)
  MPLS Experimental Bits: 0
  MPLS Bottom Of Label Stack: 1
  MPLS TTL: 126
Internet Protocol, Src Addr: 10.1.1.1 (10.1.1.1), Dst Addr: 10.5.1.1 (10.5.1.1)
Internet Control Message Protocol
  Type: 8 (Echo (ping) request)
  Code: 0
  Checksum: 0x765a (correct)
  Identifier: 0x0300
  Sequence number: d4:01
  Data (32 bytes)

```

FIGURE 8.12 : Paquet circulant de LSR1 vers LSR3

8.4. Création d'une route dynamique avec contrainte

8.4.1. Objectif

Dans les deux scénarios précédents, nous avons créé des tunnels statiquement. Le but de ce test est de créer, dynamiquement, deux tunnels, entre *LER1* et *LER2*, satisfaisant à une contrainte de bande passante de 25 Mbps. Afin de s'assurer que le choix de la route est bien effectué par rapport à la contrainte, le bande passante du chemin le plus direct est amené à une valeur inférieure à 25 Mbps.

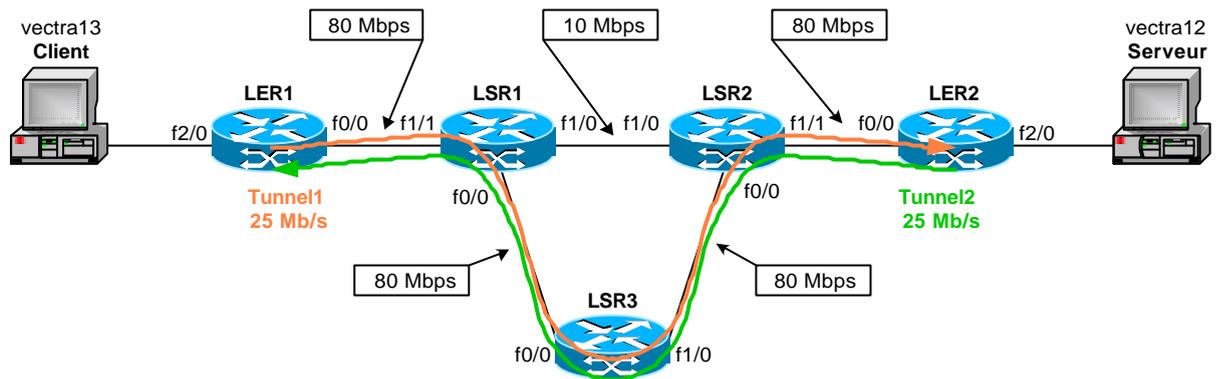


FIGURE 8.13 : Tunnels créés dynamiquement avec contrainte

8.4.2. Préparation des routeurs pour RSVP

Pour que les routeurs puissent créer dynamiquement une route avec une contrainte sur la bande passante, ils doivent connaître la topologie du réseau. Cette topologie est connue grâce à *OSPF*. Ensuite, les ressources nécessaires pour garantir cette bande passante doivent être réservées sur chaque routeur. Cela est fait par le protocole *RSVP* lors de la création des tunnels.

En reprenant la configuration décrite au paragraphe 8.2.2, préparant les routeurs pour la *traffic engineering*, il faut procéder comme suit pour configurer les routeurs :

Pour que *RSVP* puisse réserver les ressources sur chaque routeur, il faut configurer la bande passante maximale que peut allouer chaque interface à ce protocole. Voici la commande permettant cela :

```
ip rsvp bandwidth interface-kbps single-flow-kbps
```

- *interface-kbps* : Bande passante maximale allouée au protocole *RSVP*, en kbps.
- *single-flow-kbps* : Bande passante maximale allouée par flux de trafic, en kbps.

Pour notre test, toutes les interfaces autorisent *RSVP* à réserver une bande passante de 80 Mbps. Seules les interfaces f1/0 de *LSR1* et *LSR2*, limitent à 10 Mbps pour les raisons citées précédemment (FIGURE 8.12).

```
➤ routeur(config-int)# ip rsvp bandwidth 80000 80000
```

8.4.3. Création dynamique de tunnels avec contrainte

Pour la création dynamique de tunnels avec contrainte, la plupart des commandes utilisées sont identiques à celles utilisées précédemment. C'est pourquoi, seules les nouvelles commandes sont commentées.

- Configuration du tunnel 1 sur *LER1* :

Voici les commandes permettant de créer le tunnel 1 à destination de *LER2* :

```
➤ LER1(config)# interface tunnel 1
➤ LER1(config-int)# ip unnumbered loopback 0
➤ LER1(config-int)# tunnel destination 12.0.0.5
➤ LER1(config-int)# tunnel mode mpls traffic-eng
➤ LER1(config-int)# tunnel mpls traffic-eng autoroute
announce
```

La commande suivante permet de définir la bande passante requise pour le tunnel 1

Tunnel mpls traffic-eng bandwidth *bandwidth*

- *bandwidth* : Bande passante du tunnel en kbps.

Dans notre cas, le tunnel doit offrir une bande passante de 25 Mbps

```
➤ LER1(config-int)# tunnel mpls traffic-eng bandwidth 25000
```

La commande ci-dessous indique aux routeurs que le tunnel doit être créé dynamiquement en respectant la contrainte définie ci-dessus.

```
➤ LER1(config-int)# tunnel mpls traffic-eng path-option 1
dynamic
```

- Configuration du tunnel 2 sur *LER2* :

La configuration de *LER2* pour le tunnel 2 se fait de la même manière.

```
➤ LER2(config)# interface tunnel 2
➤ LER2(config-int)# ip unnumbered loopback 0
➤ LER2(config-int)# tunnel destination 12.0.0.1
➤ LER2(config-int)# tunnel mode mpls traffic-eng
➤ LER2(config-int)# tunnel mpls traffic-eng autoroute
announce
➤ LER2(config-int)# tunnel mpls traffic-eng bandwidth 25000
➤ LER2(config-int)# tunnel mpls traffic-eng path-option 1
dynamic
```

Le listing de configuration de chaque routeur se trouve dans l'annexe *Listings de configuration 3*.

8.4.4. Test et analyse

En effectuant un ping depuis le client sur le serveur, on constate que les paquets circulent à travers *LSR3* (FIGURE 8.14). Cela montre que *RSVP* a bien tenu compte de la contrainte de bande la passante, en contournant la connexion à 10 Mbps entre *LSR1* et *LSR2*. Une fois le tunnel constitué, le *forwarding* des paquets est effectué de la même manière que pour les tunnels créés statiquement. La figure ci-dessous montre le paquet circulant lors de ce ping. Les captures des diverses tables des routeurs se trouvent dans l'annexe *Captures des tables des routeurs Cisco 4*.

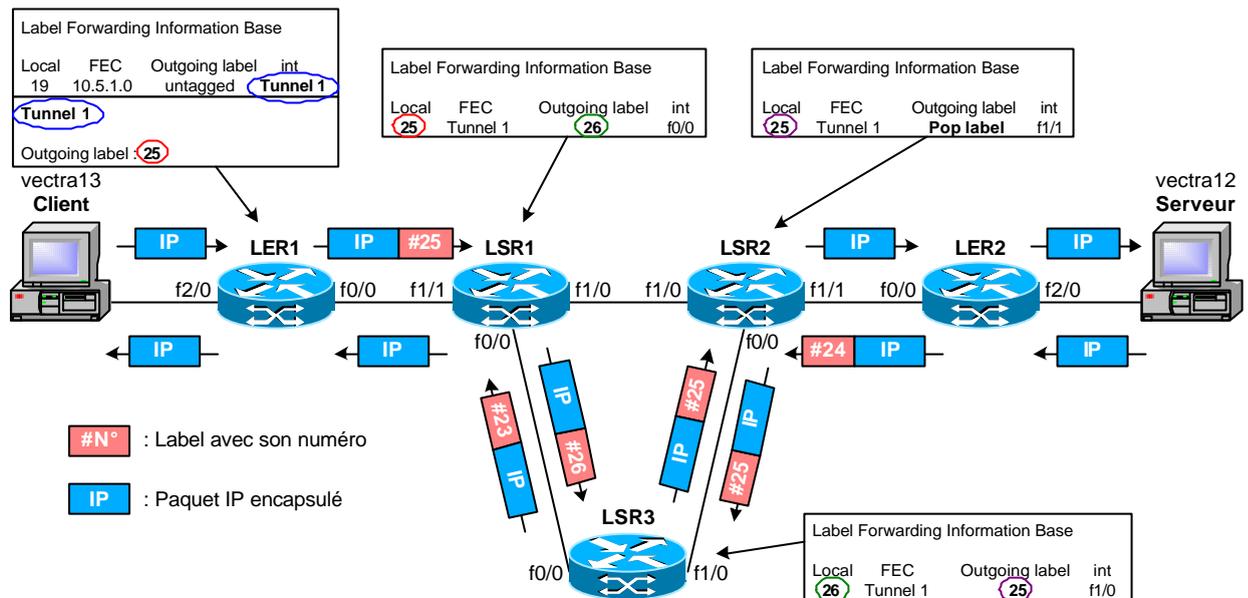


FIGURE 8.14 : Tunnels créés dynamiquement avec contrainte

Afin de comprendre les mécanismes utilisés pour déterminer la route respectant la contrainte, il faudrait étudier, dans le détail, les protocoles *OSPF* et *RSVP*. Malheureusement, le peu de temps disponible pour ce travail ne nous permet pas de commencer une telle étude. On peut simplement constater que dans les paquets *RSVP path* (FIGURE 8.15) et *RSVP resv* (FIGURE 8.16), la bande passante désirée est signalée aux différents routeurs traversés par ces paquets.

```
Resource ReserVation Protocol (RSVP)
  RSVP Header
    RSVP Version: 1
    Flags: 00
    Message Type: PATH Message (1)
    Message Checksum: 0x8aca (correct)
    Sending TTL: 254
    Message length: 232
    SESSION: IPv4-LSP, 12.0.0.5, 1, c000001
    HOP: IPv4, 10.2.1.1
    TIME VALUES: 30000 ms
    EXPLICIT ROUTE
    LABEL REQUEST: IP (0x0800)
    SESSION ATTRIBUTE
    SENDER TEMPLATE: IPv4-LSP, 12.0.0.1, 12
    SENDER TSPEC: IntServ, 3125000 bytes/sec (3125000 bytes/s = 25 Mbits/s)
    ADSPEC
```

FIGURE 8.15 : Paquet *RSVP path*

```
Resource ReserVation Protocol (RSVP)
  RSVP Header
    RSVP Version: 1
    Flags: 00
    Message Type: RESV Message (2)
    Message Checksum: 0xc938 (correct)
    Sending TTL: 255
    Message length: 108
    SESSION: IPv4-LSP, 12.0.0.5, 1, c000001
    HOP: IPv4, 10.2.1.2
    TIME VALUES: 30000 ms
    STYLE: Shared-Explicit (18)
    FLOWSPEC: Controlled-Load, 3125000 bytes/sec (3125000 bytes/s = 25 Mbits/s)
    FILTERSPEC: IPv4-LSP, 12.0.0.1, 12
    LABEL: 25
      Length: 8
      Class number: 16 - LABEL object
      C-type: 1
      Label: 25
```

FIGURE 8.16 : Paquet RSVP resv

8.5. Conclusion

Les scénarios de *traffic engineering* testé ci-dessus ont tous fonctionné selon nos attentes. Mais comme nous l'avons vu, le protocole *MPLS* nécessite plusieurs autres protocoles pour effectuer du *traffic engineering*. Dans nos tests, nous avons utilisé *OSPF*, afin que les routeurs puissent connaître la topologie du réseau, et *RSVP*, pour la distribution des labels sur des routes explicites ou avec contraintes. Le but de ce travail de diplôme n'est pas de décrire ces protocoles, mais de montrer les possibilités offertes grâce au protocole *MPLS*.

Dans le scénario du *backup path* (§ 8.3) nous avons pu mettre en pratique, avec succès, le concept présenté sur Linux concernant la création d'un *stack* de labels (§ 5.4). En effet, sur Linux, ce scénario ne fonctionne pas correctement.

9. Utilité de *MPLS* pour les *VPN*

9.1. Introduction

Le but de ce chapitre est d'illustrer la manière dont le protocole *MPLS* est utilisé pour créer des réseaux privés virtuels (*VPN* : *Virtual Privat Network*). Un *VPN* est un réseau sur lequel des clients (*customers*), se trouvant sur plusieurs sites reliés par un réseau partagé, peuvent se connecter avec les mêmes accès et la même sécurité que sur un réseau privé. Illustrons cela par un exemple (FIGURE 9.1) :

- Le réseau partagé (*core network*) est constitué des routeurs *Core*, *PE1* et *PE2* (*PE* : *Provider Edge router*).
- Il y a deux groupes de clients. *Customer A* (*CUST-A1* et *CUST-A2*) et *Customer B* (*CUST-B1* et *CUST-B2*).
- Sur chaque site, les clients possèdent un routeur connecté au *core network* (*CE* : *Customer Edge router*).

Le but du *VPN* est de relier les différents sites d'un groupe de clients, sans devoir posséder de ligne privée ou louée entre ces sites. Pour cela, chaque site est connecté à un réseau partagé (ex. Internet). Les flux de données de ces différents clients circulent donc sur un même réseau. Malgré cela, il ne faut pas que les clients du groupe A puissent accéder aux ressources du groupe B et inversement. Il faut donc pouvoir différencier ces différents flux. C'est ici qu'intervient le protocole *MPLS*. Comme nous le verrons plus bas, un label est utilisé pour distinguer les différents flux de données.

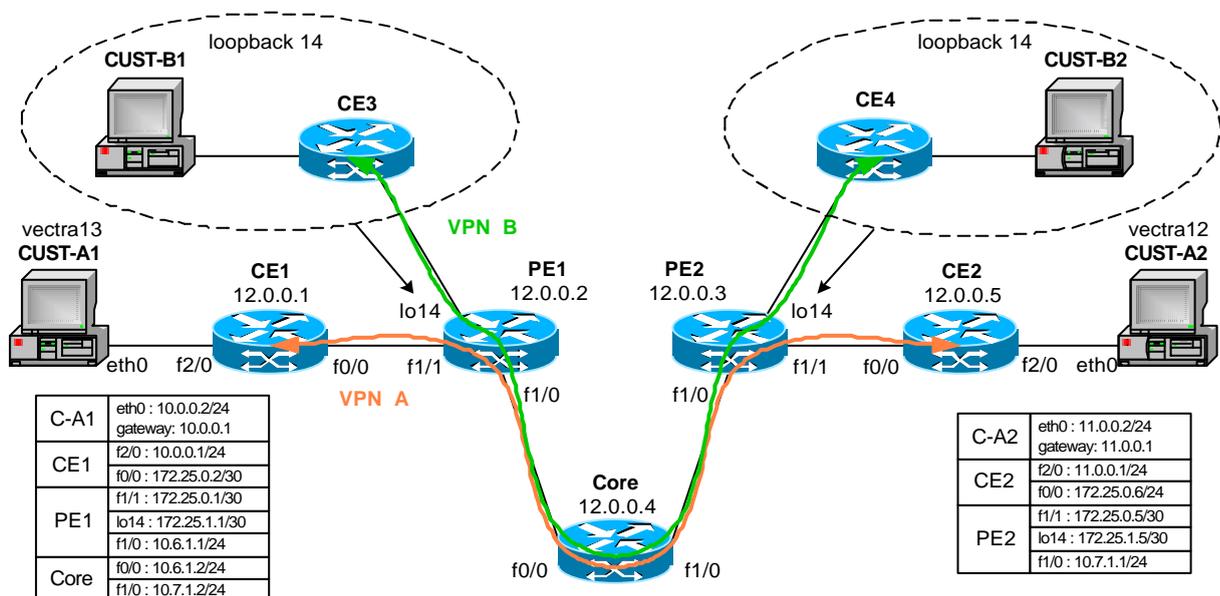


FIGURE 9.1 : Exemple de VPN

9.2. Configuration des routeurs

La configuration des *VPN* sur les routeurs Cisco demande la création de groupes d'utilisateurs sur les *PE* et la mise en place de différents protocoles de routage. Le but de ce chapitre n'est pas d'expliquer comment configurer les routeurs pour mettre en œuvre les *VPN*, mais de montrer l'utilité du protocole *MPLS* pour ces *VPN*. C'est pourquoi nous avons simplement appliqué et adapté une configuration trouvée sur le site Internet de Cisco (www.cisco.com). Le listing de la configuration utilisée, lors de ce test, pour chaque routeur, se trouve dans l'annexe *Listings de configuration 4*.

Dans cette configuration, le protocole *OSPF* est utilisé pour créer les tables de routage du *core network*. Le protocole *BGP* est employé pour distribuer les informations de routage entre les différents sites d'un même groupe de clients, afin que ces sites connaissent l'ensemble de la topologie du réseau privé virtuel. Le *forwarding* des paquets dans le *core network* est réalisé avec *MPLS*. Pour cela, *MPLS* est simplement activé sur les routeurs *PE1*, *PE2* et *Core* comme expliqué au paragraphe 7.3.

Ne disposant pas de 7 routeur pour réaliser la topologie de la FIGURE 9.1, les routeurs *CE3* et *CE4* sont virtuellement connecté sur des interfaces *loopback* (*loopback14*).

9.3. Fonctionnement du VPN grâce à MPLS

La figure ci-dessous illustre le fonctionnement d'un *VPN* en utilisant *MPLS*. Voici comment un paquet circule depuis *CUST-A1* vers *CUST-A2* :

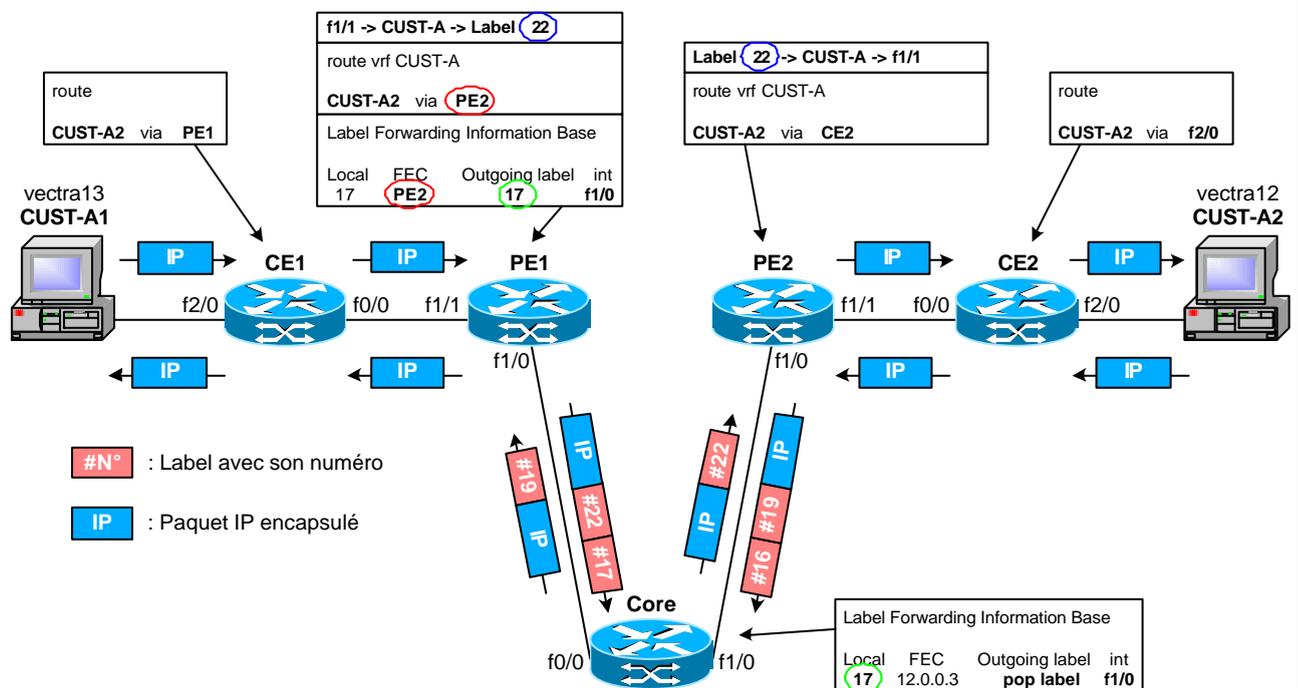


FIGURE 9.2 : Circulation des paquet sur un VPN

CUST-A1 envoie un paquet à *CE1*, car c'est son *default gateway*. Grâce à *BGP*, *CE1* connaît la topologie du réseau privé du groupe A. Dans sa table de routage, il voit qu'il doit envoyer le paquet vers *PE1*. Quand *PE1* reçoit un paquet sur l'interface *f1/1*, il sait que ce paquet appartient au groupe A. Chaque groupe est associé à un

label *MPLS*. Cela permet aux *PE* de distinguer les différents réseaux privés. Ici le label est le 22. Ensuite, *PE1* regarde dans la table de rouage du groupe A et constate que le paquet doit être envoyé à *PE2*. Selon sa table de *forwarding*, le label 17 doit être utilisé pour cette destination. *PE1* encapsule donc le paquet avec la pile de labels 22 et 17, avant de l'envoyer par l'interface f1/0. Le *forwarding* des paquets dans le *core network* utilise uniquement le label au sommet de la pile. Le routeur *Core* reçoit le paquet avec le label 17. Comme il est le dernier routeur avant *PE2*, il retire ce label (*PHP*) et envoie le paquet par l'interface f1/0. *PE2* reçoit donc un paquet avec le label 22. Grâce à ce label, il sait que le paquet appartient au groupe A. Il regarde donc dans la table de routage du groupe A et constate que le paquet doit être envoyé vers *CE2*. Enfin, *CE2* *forward* le paquet vers CUST-A2.

9.4. Conclusion

Pour un *VPN*, l'utilité du protocole *MPLS* dans le *core network* est double. D'abord, il sert à distinguer les différents réseaux privés traversant le *core network*, par l'ajout d'un premier label sur les paquets. Ensuite, il permet de *forwarder* ces paquets, dans ce réseau, par l'ajout d'un deuxième label. On constate, une fois de plus, l'utilité de la pile de labels (*label stack*) offerte par le protocole *MPLS*.

10. Remarques et conclusion finale

Les objectifs de ce travail de diplôme ont été atteints. En effet, nous avons pu réaliser une topologie de *RAS (Remote Access)* en utilisant un routeur Cisco 2600 comme serveur d'accès distant. Cela nous a permis de comprendre la philosophie utilisée par Cisco pour la configuration des routeurs. De plus, nous avons pu approfondir nos connaissances sur le protocole *PPP*. Ce protocole offre de nombreux mécanismes permettant la négociation de paramètres de connexion, l'authentification des équipements et l'établissement de la couche 3.

Une étude théorique du protocole *MPLS* a été effectuée. Cela comprend la description des divers équipements constituant un réseau *MPLS*, l'étude des mécanismes offerts par ce protocole pour *forwarder* des paquets à travers ce réseau, en utilisant le principe de *label switching*, et la description des mécanismes de distribution des labels par le protocole *LDP*.

Afin de mieux comprendre et de compléter les éléments décrits ci-dessus, des premiers essais ont été effectués sur des machines Linux. Dans cette partie du travail, de nombreux problèmes ont été rencontrés. Les patchs disponibles permettant l'implémentation de *MPLS* et de *LDP* sur ces machines n'existent encore que dans des versions en développement. C'est pourquoi la mise en oeuvre de ces patchs ne s'est pas passée sans difficultés. Heureusement, nous avons quand même réussi à mettre en évidence les mécanismes de *lable swapping*, de *label stacking* et de distribution de labels par *LDP*.

Les dernières étapes du travail de diplôme étaient l'étude des possibilités offertes par le protocole *MPLS* pour réaliser du *traffic engineering* et l'utilité de ce protocole pour la création de *VPN*. Ces mécanismes sont essentiellement prévus pour être utilisés par des *providers* sur les *core network*. C'est pourquoi ils sont uniquement implantés dans des équipements utilisés pour de tels réseaux. Notre laboratoire ne disposant pas de ces machines, la mise en oeuvre de ces mécanismes devait se faire à l'extérieur de l'école. Nous avons eu la chance de pouvoir effectuer ces essais chez Swisscom, au laboratoire IPSS de Bümplitz, sur des routeurs Cisco 7200. Ne disposant que d'une semaine pour faire les tests dans ce laboratoire, toutes les configurations ont été préparées à l'école.

MPLS est un protocole qui sera de plus en plus intégré dans les réseaux à grande vitesse, car le mécanisme de *lable switching* est plus rapide que le routage IP traditionnel. De plus, il permet de réaliser du *traffic engineering* et il est utilisable pour la création de *VPN*. Le travail réalisé pendant ce diplôme permet de comprendre les concepts de ce protocole. Toutefois, il reste de nombreux points à approfondir. Par exemple, il serait intéressant d'étudier en détail l'interaction des protocoles *RSVP* et *BGP* avec le protocole *MPLS*. De plus, il faudrait pouvoir mesurer les performances de ce protocole sur un réseau réel en charge. Comme notre laboratoire va bientôt acquérir des équipements permettant la mise en oeuvre de ces mécanismes, l'étude de ce protocole sera certainement poursuivie.

Ce travail de diplôme a été très enrichissant par la variété des sujets traités. En effet, nous avons étudié les protocoles *PPP*, *MPLS* et *LDP*. Nous avons travaillé avec des routeurs Cisco 2600, des machines Linux et des routeurs Cisco 7200. Je regrette que

le travail au laboratoire IPSS n'ait pu durer qu'une semaine. En effet, avec le matériel mis à ma disposition, des tests plus approfondis auraient pu être envisagés. De plus, l'équipe du laboratoire m'a énormément appris sur le fonctionnement d'un *core network*, en partageant ses connaissances avec moi.

Philippe LOGEAN

11. Remerciements

Je tiens à remercier toutes les personnes qui m'ont aidé à mener ce travail de diplôme à bien. Je remercie particulièrement :

- Toute ma famille et mes amis qui m'ont soutenu et encouragé tout au long de mes études et durant ce travail de diplôme.
- M. Eric Jenny et M. Gérald Litzistorf pour leur disponibilité et pour le travail qu'ils ont accompli afin de nous offrir des conditions de travail optimum.
- M. Nicolas Tinguely, M. Pascal Aebischer, M. Beat Frank, M. Gianni Del Vecchio et M. Bobby Raju du laboratoire IPSS de Swisscom à Bümplitz, pour m'avoir mis à disposition leur laboratoire et pour l'aide qu'ils m'ont fournie.
- M. Alain Hugentobler et tous les membres du jury pour le temps qu'ils nous ont consacré et pour l'intérêt qu'ils ont porté à notre travail.
- Tous mes collègues du laboratoire de transmission de données pour les très bons moments passés ensemble lors de ce travail de diplôme. Un merci particulier à Yann Souchon pour m'avoir aidé à régler les problèmes rencontrés sur les machines Linux.

12. Planning

Le travail de diplôme s'est déroulé selon le planning suivant :

Tâches	Semaine											
	39	40	41	42	43	44	45	46	47	48	49	50
RAS												
Etude des principes des routeurs Cisco	■											
Etude des commande Cisco 2600 pour le RAS		■	■	■								
Etablissement des topologies et tests		■	■	■	■							
Rédaction du mémoire				■	■							
MPLS												
Etude des protocoles MPLS et LDP					■	■						
Installation des patchs MPLS sur Linux						■	■					
Tests et mesures sur Linux							■	■				
Préparation des scénarios pour le traffic engineering								■	■			
Test des scénarios au laboratoire IPSS									■	■		
Rédaction du mémoire										■	■	
Préparation de la défense de diplôme											■	■

13. Annexes

Annexes *RAS*

- Configuration « routeur à routeur » Cisco 2500 HDLC
- Configuration « routeur à routeur » Cisco 2600 HDLC
- Configuration « routeur à routeur » Cisco 2500 PPP
- Configuration « routeur à routeur » Cisco 2600 PPP
- Configuration Cisco2600-TA avec adresse IP statique
- Configuration Cisco2600 pour attribution d'adresses dynamiques
- Configuration Cisco2600 pour accès au réseau du labo
- Configuration Cisco2600 pour accès au réseau du labo par plusieurs clients
- Capture de l'établissement entre un PC client utilisant la FRITZ!CARD USB et le routeur Cisco2600

Annexes *MPLS sur Linux*

- Installation des patches sur LINUX
- Captures 1
- Captures 2
- Captures 3
- Captures 4
- Captures 5

Annexes *MPLS sur Cisco*

- Captures des tables des routeurs Cisco 1
- Listings de configuration 1
- Captures des tables des routeurs Cisco 2
- Listings de configuration 2
- Captures des tables des routeurs Cisco 3
- Listings de configuration 3
- Captures des tables des routeurs Cisco 4
- Listings de configuration 4
- Captures des tables des routeurs Cisco 5

Liste du matériel

14. Bibliographie

14.1. Livres

- Introduction to Cisco router configuration
Laura Chappell
Cisco Press (1999)
- MPLS Technology and Applications
Bruce Davie, Yakov Rekhter
Academic Press (2000)
- MPLS and VPN architectures
Ivan Pepelnjak, Jim Guichard
Cisco Press (2001)

14.2. RFC

- The Point-to-Point Protocol (PPP) for the Transmission of Multi-protocol Datagrams over Point-to-Point Links
William Allen Simpson
RFC 1331 (1992)
- The PPP Internet Protocol Control Protocol (IPCP)
Glenn McGregor
RFC 1332 (1992)
- PPP Authentication Protocols
William Allen Simpson
RFC 1334 (1992)
- PPP over ISDN
William Allen Simpson
RFC 1618 (1994)
- The Point-to-Point Protocol (PPP)
William Allen Simpson
RFC 1661 (1994)
- The PPP Multilink Protocol (MP)
Keith Sklower, Brian Lloyd, Glenn McGregor, Dave Carr, Tom Coradetti
RFC 1990 (1996)
- PPP Challenge Handshake Authentication Protocol (CHAP)
William Allen Simpson
RFC 1994 (1996)

- Multiprotocol Label Switching Architecture
Eric C. Rosen, Arun Viswanathan, Ross Callon
RFC 3031 (2001)
- LDP Specification
Loa Andersson, Paul Doolan, Nancy Feldman, Andre Fredette, Bob Thomas
RFC 3036 (1991)

14.3. Web

- <http://www.cisco.com>
Site officiel de Cisco. Contient énormément d'informations et d'exemples pour la configuration des routeurs Cisco. Permet de télécharger des *IOS*.
- <http://www.ietf.org/html.charters/mpls-charter.html>
Page officielle de l'*IETF (Internet Engineering Task Force)* sur *MPLS*. Permet de suivre l'évolution du protocole *MPLS*. Contient les *Drafts* et les RFC en rapport avec ce protocole.
- <http://sourceforge.net/projects/mpls-linux/>
Site officiel du projet *MPLS* sur Linux. Permet de télécharger les patches nécessaires à l'installation de *MPLS* sur Linux
- <http://www.geocrawler.com/lists/3/SourceForge/7968/0/>
Archives du forum de discussions de « sourceforge » sur *MPLS*. Permet de trouver des informations sur le fonctionnement des patches *MPLS* sur Linux.
- <http://sunsite.cnlab-switch.ch/cgi-bin/search/standard/nph-findstd>
Site permettant la recherche des RFC