Groupe de travail Réseau
Request for Comments : 3209
Catégorie : Sur la voie de la normalisation
décembre 2001
Traduction Claude Brière de L'Isle

D. Awduche, Movaz Networks, Inc.
L. Berger, D. Gan, Juniper Networks, Inc.
T. Li, Procket Networks, Inc.
V. Srinivasan, Cosine Communications, Inc.
G. Swallow, Cisco Systems, Inc.

RSVP-TE : Extensions à RSVP pour LSP tunnels

Statut du présent mémoire

Notice de Copyright

Résumé

Le présent document décrit l'utilisation du protocole de réservation de ressource (RSVP, *Resource Reservation Protocol*) y compris de toutes ses extensions nécessaires, pour établir des chemins de commutation d'étiquette (LSP, *label-switched path*) en commutation d'étiquette multi protocoles (MPLS, *Multi-Protocol Label Switching*). Comme le flux le long d'un LSP est complètement identifié par l'étiquette appliquée au nœud d'entrée du chemin, ces chemins peuvent être traités comme des tunnels. Une application clé des LSP tunnels est l'ingénierie du trafic avec MPLS comme spécifié dans la RFC2702.

On propose plusieurs objets supplémentaires qui étendent RSVP, permettant l'établissement de chemins de commutation d'étiquette à acheminement explicite en utilisant RSVP comme protocole de signalisation. Le résultat est l'instanciation de tunnels de commutation d'étiquette qui peuvent être automatiquement acheminés loin des défaillances, encombrements et goulets d'étranglement du réseau.

Table des Matières

1. Introduction....2
1.1 Fondements....2
1.2 Terminologie....3
2. Généralités....4
2.1 LSP tunnels et tunnels à ingénierie du trafic....4
2.2 Fonctionnement des LSP tunnels....4
2.3 Classes de service....5
2.4 Styles de réservation....5
2.5 Réacheminement des tunnels à ingénierie du trafic....6
2.6 MTU de chemin....7
3. Formats des messages en rapport avec les LSP tunnels....8
3.1 Message Path....8
3.2 Message Resv....8
4. Objets en rapport avec le LS tunnelP....9
4.1 Objet Label....9
4.2 Objet Label Request....10
4.3 Objet Explicit Route....12
4.4 Objets Record Route....17
4.5 Codes d'erreur pour ERO et RRO....20
4.6 Objets Session, Gabarit d'envoyeur et Spéc de filtre....21
4.7 Objet Session Attribute....23
5. Extension de Hello....26
5.1 Format du message Hello....27
5.2 Formats d'objet HELLO....27
5.3 Usage du message Hello....28
5.4 Considérations de multi-liaison....29
5.5 Compatibilité....29

6. Considérations pour la sécurité....29
7. Considérations relatives à l'IANA....29
7.1 Types de message....29
7.2 Numéros de classe et C-Types....30
7.3 Codes d'erreur et sous codes de valeur d'erreur à définition mondiale....30
7.4 Définitions de sous-objet....31
8. Considérations sur la propriété intellectuelle....31
9. Remerciements....31
10. Références....31
11. Adresse des auteurs....32
12. Déclaration de droits de reproduction....32

1. Introduction

Le paragraphe 2.9 de l'architecture MPLS [RFC3031] définit un protocole de distribution d'étiquettes comme un ensemble de procédures par lesquelles un routeur de commutation d'étiquettes (LSR, *Label Switched Router*) informe un autre routeur de la signification des étiquettes utilisées pour transmettre le trafic entre et à travers eux. L'architecture MPLS ne postule pas l'unicité du protocole de distribution d'étiquettes. Le présent document est une spécification des extensions à RSVP pour l'établissement des chemins de commutation d'étiquettes (LSP, *Label Switched Path*) dans les réseaux MPLS.

Plusieurs des nouvelles caractéristiques décrites dans le présent document étaient motivées par les exigences d'ingénierie du trafic sur MPLS (voir la [RFC2702]). En particulier, le protocole RSVP étendu prend en charge la réalisation de LSP à acheminement explicite, avec ou sans réservation de ressource. Il prend aussi en charge le réacheminement en douceur des LSP, la préemption, et la détection des boucles.

Les LSP créés avec RSVP peuvent être utilisés pour porter les "circuits de trafic" décrits dans la [RFC2702]. Le LSP qui porte un circuit de trafic et un circuit de trafic sont des concepts distincts bien qu'en rapport étroit. Par exemple, deux LSP entre la même source et destination pourraient partager la charge pour porter un seul circuit de trafic. À l'inverse, plusieurs circuits de trafic pourraient être portés dans le même LSP si, par exemple, le LSP était capable de porter plusieurs classes de service. L'applicabilité de ces extensions est discutée plus avant dans la [RFC3210].

Comme le trafic qui s'écoule le long d'un chemin à commutation d'étiquettes est défini par l'étiquette appliquée au nœud d'entrée du LSP, ces chemins peuvent être traités comme des tunnels, tunnelant sous les mécanismes normaux d'acheminement et de filtrage IP. Lorsque un LSP est utilisé de cette façon, on l'appelle un LSP tunnel.

Les LSP tunnels permettent la mise en œuvre de diverses politiques se rapportant à l'optimisation des performances du réseau. Par exemple, les LSP tunnels peuvent être acheminés automatiquement ou manuellement à partir de défaillances du réseau, d'encombrements, et de goulets d'étranglement. De plus, plusieurs LSP tunnels en parallèle peuvent être établis entre deux nœuds, et le trafic entre les deux nœuds peut être transposé sur les LSP tunnels conformément à la politique locale. Bien que l'ingénierie du trafic (c'est-à-dire, l'optimisation des performances des réseaux de fonctionnement) soit supposée être une importante application de la présente spécification, le protocole RSVP étendu peut être utilisé dans un contexte beaucoup plus large.

L'objet du présent document est de décrire l'utilisation de RSVP pour établir des LSP tunnels. L'intention est de décrire de façon complète tous les objets, formats de paquets et procédures, requis pour réaliser des mises en œuvre interopérables. Quelques nouveaux objets sont aussi définis qui améliorent la gestion et le diagnostic des LSP tunnels.

Le document décrit aussi un moyen de détection rapide des défaillances des nœuds via un nouveau message HELLO.

Tous les objets et messages décrits dans la présente spécification sont facultatifs par rapport à RSVP. Le présent document expose ce qui se passe lorsque un objet décrit ici n'est pas pris en charge par un nœud.

Tout au long de ce document, la discussion se restreint aux chemins de commutation d'étiquette en envoi individuel. Les LSP en diffusion groupée feront l'objet d'études ultérieures.

1.1 Fondements

Les hôtes et routeurs qui prennent en charge à la fois RSVP [RFC2205] et la commutation d'étiquettes multi-protocoles [RFC3031] peuvent associer des étiquettes aux flux RSVP. Lorsque MPLS et RSVP sont combinés, la définition d'un flux

peut être rendue plus flexible. Une fois qu'est établi un chemin de commutation d'étiquette (LSP, *label switched path*) le trafic à travers le chemin est défini par l'étiquette appliquée au nœud d'entrée du LSP. La transposition des étiquettes en trafic peut être accomplie en utilisant un certain nombre de critères différents. La même valeur d'étiquette est allouée à l'ensemble des paquets par un nœud spécifique et ils sont dits appartenir à la même classe d'équivalence de transmission (FEC, *Forwarding Equivalence Class*) (voir la [RFC3031]) et définissent effectivement le "flux RSVP". Lorsque le trafic est transposé de cette façon sur un chemin de commutation d'étiquette, on appelle le LSP un "LSP tunnel". Lorsque les étiquettes sont associées à des flux de trafic, il devient possible à un routeur d'identifier l'état de réservation approprié pour un paquet sur la base de la valeur de l'étiquette.

Le modèle de protocole de signalisation utilise la distribution d'étiquettes vers l'aval à la demande. Une demande de liaison des étiquettes à un LSP tunnel spécifique est initiée par un nœud d'entrée au moyen du message RSVP Path. À cette fin, le message RSVP Path est augmenté d'un objet LABEL_REQUEST. Les étiquettes sont allouées vers l'aval et distribuées (propagées vers l'amont) au moyen du message RSVP Resv. À cette fin, le message RSVP Resv est étendu avec un objet spécial LABEL. Les procédures d'allocation, distribution, lien, et mise en pile des étiquettes sont décrites dans les sections suivantes du présent document.

Le modèle de protocole de signalisation prend aussi en charge la capacité d'acheminement explicite. Cela est accompli en incorporant un simple objet EXPLICIT_ROUTE dans les messages RSVP Path. L'objet EXPLICIT_ROUTE encapsule un enchaînement de bonds qui constitue le chemin à acheminement explicite. En utilisant cet objet, les chemins pris par les flux RSVP-MPLS à commutation d'étiquettes peuvent être prédéterminés, indépendamment de l'acheminement IP conventionnel. Le chemin à acheminement explicite peut être spécifié administrativement, ou calculé automatiquement par une entité convenable sur la base de la QS et des exigences de politique, en prenant en considération l'état prévalant du réseau. En général, le calcul du chemin peut être piloté par commande ou piloté par les données. Les mécanismes, processus, et algorithmes utilisés pour calculer les chemins à acheminement explicite sortent du domaine d'application de la présente spécification.

Une application utile de l'acheminement explicite est l'ingénierie du trafic. En utilisant des LSP à acheminement explicite, un nœud à la bordure d'entrée d'un domaine MPLS peut contrôler le chemin par lequel le trafic traverse depuis lui-même, à travers le réseau MPLS, jusqu'au nœud de sortie. L'acheminement explicite peut être utilisé pour optimiser l'utilisation des ressources du réseau et améliorer les caractéristiques de performances du trafic.

Le concept de chemins de commutation d'étiquette à acheminement explicite peut être généralisé grâce à la notion de nœud abstrait. Un nœud abstrait est un groupe de nœuds dont la topologie interne est opaque au nœud d'entrée du LSP. Un nœud abstrait est dit simple si il contient seulement un nœud physique. En utilisant ce concept d'abstraction, un LSP à acheminement explicite peut être spécifié comme une séquence de préfixes IP ou une séquence de systèmes autonomes.

Le modèle de protocole de signalisation prend en charge la spécification d'un chemin explicite comme une séquence de chemins stricts et lâches. La combinaison de nœuds abstraits et de chemins stricts et lâches améliore de façon significative la souplesse des définitions de chemin.

Un avantage de l'utilisation de RSVP pour établir les LSP tunnels est qu'il permet l'allocation de ressources le long du chemin. Par exemple, la bande passante peut être allouée à un LSP tunnel en utilisant les réservations standard RSVP et les classes de service intégré de la [RFC2211].

Bien que les réservations de ressources soient utiles, elles ne sont pas obligatoires. Bien sûr, un LSP peut être établi sans aucune réservation de ressource. De tels LSP sans réservation de ressource peuvent être utilisés, par exemple, pour porter du trafic au mieux. Ils peuvent aussi être utilisés dans de nombreux autres contextes, y compris la mise en œuvre de politiques de repli et de récupération dans des conditions de faute, et ainsi de suite.

1.2 Terminologie

Dans le présent document, les mots clés "DOIT", "NE DOIT PAS", "EXIGE", "DEVRA", "NE DEVRA PAS", "DEVRAIT", "NE DEVRAIT PAS", "RECOMMANDE", "PEUT", et "FACULTATIF" sont à interpréter comme décrit dans le BCP 14, [RFC2119].

Le lecteur est supposé familier avec la terminologie des [RFC2205], [RFC2702] et [RFC3031].

Nœud abstrait
Groupe de nœuds dont la topologie interne est opaque au nœud d'entrée du LSP. Un nœud abstrait est dit simple si il contient un seul nœud physique.

LSP à acheminement explicite
LSP dont le chemin est établi par un moyen autre que l'acheminement IP normal.

Chemin à commutation d'étiquette
Chemin créé par enchaînement d'un ou plusieurs bonds de commutation d'étiquette, permettant qu'un paquet soit transmis par échange d'étiquettes d'un nœud MPLS à un autre nœud MPLS. Voir une définition plus précise dans la [RFC3031].

LSP
Chemin de commutation d'étiquette

LSP tunnel
LSP qui est utilisé pour tunneler en dessous de l'acheminement IP normal et/ou des mécanismes de filtrage.

Tunnel à ingénierie du trafic (Tunnel TE)
Ensemble d'un ou plusieurs LSP tunnels qui portent un circuit de trafic.

Circuit de trafic
Ensemble de flux agrégés par leur classe de service puis placés sur un LSP ou ensemble de LSP appelé tunnel à ingénierie du trafic. Voir un exposé plus complet dans la [RFC2702].

2. Généralités

2.1 Tunnels LSP et tunnels à ingénierie du trafic

Selon la [RFC2205], "RSVP définit une 'session' comme étant un flux de données avec une destination particulière et un protocole de couche transport". Cependant, quand RSVP et MPLS sont combinés, un flux ou une session peut être défini avec une souplesse et une généralité plus grandes. Le nœud d'entrée d'un LSP peut utiliser divers moyens pour déterminer quels paquets sont assignés à une étiquette particulière. Une fois qu'une étiquette est assignée à un ensemble de paquets, l'étiquette définit effectivement le "flux" à travers le LSP. On se réfère à un tel LSP sous le nom de "LSP tunnel" parce que le trafic à travers lui est opaque aux nœuds intermédiaires le long du chemin de commutation d'étiquettes.

De nouveaux objets RSVP SESSION, SENDER_TEMPLATE, et FILTER_SPEC, appelés LSP_TUNNEL_IPv4 et LSP_TUNNEL_IPv6 ont été définis pour prendre en charge le dispositif de LSP tunnel. La sémantique de ces objets, du point de vue d'un nœud le long du chemin de commutation d'étiquettes, est que le trafic qui appartient au LSP tunnel est identifié seulement sur la base des paquets qui arrivent du PHOP *(previous hop)* ou "bond précédent" (voir la [RFC2205]) avec la ou les valeurs d'étiquettes particulières assignées par ce nœud aux envoyeurs amont de la session. En fait, le IPv4(v6) qui apparaît dans le nom de l'objet note seulement que l'adresse de destination est une adresse IPv4(v6). Quand on se réfère de façon générique à ces objets, on utilise le qualificatif LSP_TUNNEL.

Dans certaines applications il est utile d'associer des ensembles de LSP tunnels. Cela peut être utile durant des opérations de réacheminement ou pour étaler un circuit de trafic sur plusieurs chemins. Dans l'application d'ingénierie du trafic, de tels ensembles sont appelés des tunnels à ingénierie du trafic (tunnels TE). Pour permettre l'identification et l'association de tels LSP tunnels, deux identifiants sont portés. Un identifiant de tunnel fait partie de l'objet SESSION. L'objet SESSION définit de façon univoque un tunnel à ingénierie du trafic. Les objets SENDER_TEMPLATE et FILTER_SPEC portent un identifiant de LSP. L'objet SENDER_TEMPLATE (ou FILTER_SPEC) conjointement avec l'objet SESSION identifie de façon univoque un LSP tunnel

2.2 Fonctionnement des LSP tunnels

Ce paragraphe récapitule certaines des caractéristiques prises en charge par RSVP, tel qu'étendu par le présent document, qui se rapportent au fonctionnement des LSP tunnels. Cela inclut : (1) la capacité d'établir des LSP tunnels avec ou sans exigences de qualité de service (QS), (2) la capacité de réacheminer de façon dynamique un LSP tunnel établi, (3) la capacité d'observer le chemin réel traversé par un LSP tunnel établi, (4) la capacité d'identifier et diagnostiquer les LSP tunnels, (5) la capacité de préempter un LSP tunnel établi sous contrôle administratif de politique, et (6) la capacité d'effectuer l'allocation, la distribution, et la liaison d'étiquettes vers l'aval à la demande. Dans les paragraphes qui suivent, ces caractéristiques sont brièvement décrites. Des descriptions plus détaillées se trouvent dans les sections suivantes du présent document.

Pour créer un LSP tunnel, le premier nœud MPLS sur le chemin – c'est-à-dire, le nœud envoyeur par rapport au chemin – crée un message RSVP Path avec un type de session de LSP_TUNNEL_IPv4 ou LSP_TUNNEL_IPv6 et insère un objet LABEL_REQUEST dans le message Path. L'objet LABEL_REQUEST indique qu'est demandée une liaison d'étiquette pour ce chemin, et il donne aussi une indication du protocole de couche réseau qui va être porté sur ce chemin. La raison en est que le protocole de couche réseau qui descend un LSP ne peut pas être supposé être IP et ne peut pas être déduit de l'en-tête de couche 2, qui identifie simplement le protocole de couche supérieure comme MPLS.

Si le nœud envoyeur a connaissance d'un chemin qui a une forte probabilité de satisfaire aux exigences de QS du tunnel, ou qui fait une utilisation efficace des ressources du réseau, ou qui satisfait à certains critères de politique, le nœud peut décider d'utiliser le chemin pour certaines ou pour toutes ses sessions. Pour ce faire, le nœud envoyeur ajoute un objet EXPLICIT_ROUTE au message RSVP Path. L'objet EXPLICIT_ROUTE spécifie le chemin comme une séquence de nœuds abstraits.

Si, après la réussite de l'établissement d'une session, le nœud envoyeur découvre un meilleur chemin, l'envoyeur peut réacheminer dynamiquement la session en changeant simplement l'objet EXPLICIT_ROUTE. Si on rencontre des problèmes avec un objet EXPLICIT_ROUTE, soit parce qu'il cause une boucle d'acheminement, soit parce que des routeurs intermédiaires ne le prennent pas en charge, cela est notifié au nœud envoyeur.

En ajoutant un objet RECORD_ROUTE au message Path, le nœud envoyeur peut recevoir des informations sur le chemin réel que traverse le LSP tunnel. Le nœud envoyeur peut aussi utiliser cet objet pour demander une notification de la part du réseau concernant les changements au chemin d'acheminement. L'objet RECORD_ROUTE est analogue à un vecteur de chemin, et donc peut être utilisé pour la détection des boucles.

Finalement, un objet SESSION_ATTRIBUTE peut être ajouté aux messages Path pour aider à l'identification et au diagnostic de session. Des informations de contrôle supplémentaires, telles que les priorités d'établissement et de garde, les affinités de ressources (voir la [RFC2702]) et la protection locale, sont aussi incluses dans cet objet.

Les routeurs le long du chemin peuvent utiliser les priorités d'établissement et de garde ainsi que tout objet SENDER_TSPEC et POLICY_DATA contenu dans les messages Path comme entrées de contrôle de politique. Par exemple, dans une application d'ingénierie du trafic, il est très utile d'utiliser le message Path comme moyen de vérifier que la bande passante existe avec une priorité particulière tout le long d'un chemin avant de préempter des réservations de priorité inférieure. Si il est permis à un message Path de progresser alors qu'il y a des ressources insuffisantes, il y a alors danger que des réservations de priorité inférieure vers l'aval de ce point soient inutilement préemptées dans une tentative futile de service de cette demande.

Lorsque l'objet EXPLICIT_ROUTE (ERO, *EXPLICIT_ROUTE object*) est présent, le message Path est transmis vers sa destination le long d'un chemin spécifié par le ERO. Chaque nœud le long du chemin enregistre le ERO dans son bloc d'état de chemin. Les nœuds peuvent aussi modifier le ERO avant de transmettre le message Path. Dans ce cas, le ERO modifié DEVRAIT être mémorisé dans le bloc d'état de chemin en plus du ERO reçu.

L'objet LABEL_REQUEST demande aux routeurs et nœuds receveurs intermédiaires de fournir un lien d'étiquette pour la session. Si un nœud est incapable de fournir un lien d'étiquette, il envoie un message PathErr avec une erreur "classe d'objet inconnue". Si l'objet LABEL_REQUEST n'est pas pris en charge de bout en bout, le nœud envoyeur en sera notifié par le premier nœud qui n'assure pas cette prise en charge.

Le nœud de destination d'un chemin de commutation d'étiquette répond à une LABEL_REQUEST en incluant un objet LABEL dans sa réponse au message Resv RSVP. L'objet LABEL est inséré dans la liste des spécifications de filtre immédiatement à la suite de la spécification de filtre à laquelle il appartient.

Le message Resv est renvoyé en amont vers l'envoyeur, en suivant l'état de chemin créé par le message Path, en ordre inverse. Noter que si l'état de chemin a été créé par l'utilisation d'un ERO, le message Resv va alors suivre le chemin inverse de l'ERO.

Chaque nœud qui reçoit un message Resv contenant un objet LABEL utilise cette étiquette pour le trafic sortant associé à ce LSP tunnel. Si le nœud n'est pas l'envoyeur, il alloue une nouvelle étiquette et la place dans l'objet LABEL correspondant du message Resv qu'il envoie en amont au PHOP. L'étiquette envoyée en amont dans l'objet LABEL est l'étiquette que ce nœud va utiliser pour identifier le trafic entrant associé à ce LSP tunnel. Cette étiquette sert aussi de raccourci pour la spécification de filtre. Le nœud peut maintenant mettre à jour sa transposition d'étiquette entrante (ILM, *Incoming Label Map*) qui est utilisée pour transposer les paquets étiquetés entrants en une entrée de transmission d'étiquette pour le prochain bond (NHLFE, *Next Hop Label Forwarding Entry*) voir la [RFC3031].

Lorsque le message Resv est propagé en amont jusqu'au nœud envoyeur, un chemin de commutation d'étiquettes est effectivement établi.

2.3 Classes de service

Le présent document ne restreint pas le type de demandes de service intégré pour les réservations. Cependant, une mise en œuvre DEVRAIT prendre en charge le service à charge contrôlée [RFC2211] et le service Nul [RFC2997].

2.4 Styles de réservation

Le nœud receveur peut choisir parmi un ensemble de styles de réservation possibles pour chaque session, et chaque session RSVP doit avoir un style particulier. Les envoyeurs n'ont pas d'influence sur le choix du style de réservation. Le receveur peut choisir différents styles de réservation pour des LSP différents.

Une session RSVP peut résulter en un ou plusieurs LSP, selon le style de réservation choisi.

Certains styles de réservation, comme de filtre fixe (FF), dédient une réservation particulière à un nœud envoyeur individuel. D'autres styles de réservation, comme de filtre générique (WF) et partagé explicite (SE), peuvent partager une réservation entre plusieurs nœuds envoyeurs. Les paragraphes qui suivent exposent les différents styles de réservation et leurs avantages et inconvénients. On trouvera un exposé plus détaillé sur les styles de réservation dans la [RFC2205].

2.4.1 Style Filtre fixe (FF)

Le style de réservation Filtre fixe (FF, *Fixed Filter*) crée une réservation distincte pour le trafic provenant de chaque envoyeur qui n'est pas partagé par d'autres envoyeurs. Ce style est courant pour les applications dans lesquelles le trafic provenant de chaque envoyeur va vraisemblablement être concurrent et indépendant. La quantité totale de bande passante réservée sur une liaison pour les sessions qui utilisent FF est la somme des réservations pour les envoyeurs individuels.

Comme chaque envoyeur a sa propre réservation, une étiquette unique est allouée à chaque envoyeur. Il peut en résulter un LSP point à point entre chaque paire d'envoyeur/receveur.

2.4.2 Style Filtre générique (WF)

Avec le style de réservation Filtre générique (WF, *Wildcard Filter*) une seule réservation partagée est utilisée pour tous les envoyeurs d'une session. La réservation totale sur une liaison reste la même sans considération du nombre d'envoyeurs.

Un seul chemin de commutation d'étiquette en multipoint à point est créé pour tous les envoyeurs de la session. Sur les liaisons que partagent les envoyeurs de la session, une seule valeur d'étiquette est allouée à la session. Si il n'y a qu'un seul envoyeur, le LSP ressemble à une connexion point à point normale. Lorsque plusieurs envoyeurs sont présents, un LSP multipoint à point (un arbre renversé) est créé.

Ce style est utile pour les applications dans lesquelles tous les envoyeurs n'envoient pas de trafic en même temps. Une conférence téléphonique, par exemple, est une application dans laquelle tous les locuteurs ne parlent pas en même temps. Si cependant, tous les envoyeurs envoient simultanément, il n'y a alors aucun moyen d'avoir la bonne réservation. Soit la bande passante réservée sur les liaisons proches de la destination sera inférieure à ce qui est requis, soit la bande passante réservée sur les liaisons proches de certains envoyeurs sera supérieure à ce qui est requis. Cela restreint l'applicabilité de WF pour les besoins de l'ingénierie du trafic.

De plus, à cause des règles de fusion de WF, les objets EXPLICIT_ROUTE ne peuvent pas être utilisés avec les réservations WF. Par suite de ce problème et du manque d'applicabilité à l'ingénierie du trafic, l'utilisation de WF n'est pas prise en considération dans le présent document.

2.4.3 Style Partagé explicite (SE)

Le style Partagé explicite (SE, *Shared Explicit*) permet à un receveur de spécifier explicitement les envoyeurs à inclure dans une réservation. Il y a une seule réservation sur une liaison pour tous les envoyeurs énumérés. Parce que chaque envoyeur figure explicitement dans le message Resv, différentes étiquettes peuvent être allouées aux différents envoyeurs, créant par là des LSP distincts.

Les réservations de style SE peuvent être fournies en utilisant un chemin de commutation d'étiquette en multipoint à point ou LSP par envoyeur. Les LSP en multipoint à point peuvent être utilisés lorsque les messages Path ne portent pas l'objet EXPLICIT_ROUTE, ou quand les messages Path ont des objets EXPLICIT_ROUTE identiques. Dans aucun de ces cas une étiquette commune ne peut être allouée.

Les messages Path provenant d'envoyeurs différents peuvent chacun porter leur propre ERO, et les chemins pris par les envoyeurs peuvent converger et diverger en tout point de la topologie du réseau. Lorsque les messages Path ont des objets EXPLICIT_ROUTE différents, des LSP séparés doivent être établis pour chaque objet EXPLICIT_ROUTE.

2.5 Réacheminement des tunnels à ingénierie du trafic

Une des exigences de l'ingénierie du trafic est la capacité de réacheminer un tunnel TE établi sous un certain nombre de conditions, sur la base de la politique administrative. Par exemple, dans certains contextes, une politique administrative peut imposer qu'un certain tunnel TE soit réacheminé lorsque un chemin plus "optimal" devient disponible. Un autre contexte important qui impose habituellement un réacheminement de tunnel TE est lors de la défaillance d'une ressource le long du chemin établi du tunnel TE. Avec certaines politiques, il peut aussi être nécessaire de faire revenir le tunnel TE à son chemin d'origine lorsque la ressource défaillante redevient active.

En général, il est très souhaitable de ne pas interrompre le trafic, ou contrecarrer le fonctionnement du réseau lorsque le réacheminement d'un tunnel TE est en cours. Cette exigence de réacheminement adaptatif et en douceur nécessite l'établissement d'un nouveau LSP tunnel et d'y transférer le trafic du vieux LSP avant de supprimer le vieux LSP tunnel. Ce concept est appelé "faire avant de couper". Un problème peut survenir parce que le vieux LSP tunnel et le nouveau peuvent entrer en compétition l'un avec l'autre pour les ressources sur les segments de réseau qu'ils ont en commun. Selon la disponibilité des ressources, cette compétition peut amener le contrôle d'admission à empêcher le nouveau LSP tunnel d'être établi. Un avantage de l'utilisation de RSVP pour établir des LSP tunnels est qu'il résout ce problème très élégamment.

Pour prendre en charge en douceur le "faire avant de couper", il est nécessaire que sur les liaisons qui sont communes au vieux et au nouveau LSP, les ressources utilisées par le vieux LSP tunnel ne devraient pas être libérées avant que le trafic ait été passé au nouveau LSP tunnel, et les réservations ne devraient pas être comptées deux fois parce que cela pourrait être cause de rejet du nouveau LSP tunnel par le contrôle d'admission.

Une situation similaire peut survenir lorsque on veut augmenter la bande passante d'un tunnel TE. La nouvelle réservation sera pour la totalité de la quantité nécessaire, mais l'allocation réelle nécessaire est seulement du delta entre la nouvelle et l'ancienne bande passante. Si une régulation est appliquée aux messages Path par les nœuds intermédiaires, un message Path qui demande trop de bande passante sera alors rejeté. Dans cette situation, simplement augmenter la demande de bande passante sans changer le SENDER_TEMPLATE pourrait résulter en la suppression d'un tunnel en fonction de la politique locale.

La combinaison de l'objet SESSION LSP_TUNNEL et du style SE de réservation s'accommode naturellement des transitions en douceur de bande passante et d'acheminement. L'idée est que le vieux et le nouveau LSP tunnel partagent les ressources le long des liaisons qu'ils ont en commun. L'objet SESSION LSP_TUNNEL est utilisé pour restreindre la portée de la session RSVP au tunnel TE particulier en question. Pour identifier de façon univoque un tunnel TE, on utilise la combinaison de l'adresse IP de destination (une adresse du nœud qui est la sortie du tunnel) un identifiant de tunnel, et l'adresse IP du nœud d'entrée du tunnel, qui est placée dans le champ Identifiant de tunnel étendu.

Durant l'opération de réacheminement ou d'augmentation de bande passante, l'entrée du tunnel doit apparaître comme deux envoyeurs différents à la session RSVP. Ceci est réalisé par l'inclusion du "ID de LSP", qui est porté dans les objets SENDER_TEMPLATE et FILTER_SPEC. Comme la sémantique de ces objets a changé, de nouveaux C-Types sont alloués.

Pour effectuer un réacheminement, le nœud d'entrée prend un nouvel identifiant de LSP et forme un nouveau SENDER_TEMPLATE. Le nœud d'entrée crée alors un nouvel ERO pour définir le nouveau chemin. Après cela, le nœud envoie un nouveau message Path en utilisant l'objet SESSION original et les nouveaux SENDER_TEMPLATE et ERO. Il continue d'utiliser le vieux LSP et rafraîchit le vieux message Path. Sur les liaisons qui ne sont pas en commun, le nouveau message Path est traité comme un établissement de nouveau LSP tunnel conventionnel. Sur les liaisons détenues en commun, l'objet SESSION partagé et le style SE permettent au LSP d'être établi en partageant les ressources avec le vieux LSP. Une fois que le nœud d'entrée a reçu un message Resv pour le nouveau LSP, il peut faire transiter le trafic sur lui et supprimer le vieux LSP.

Pour effectuer une augmentation de bande passante, un nouveau message Path avec un nouveau LSP_ID peut être utilisé pour tenter une plus grande réservation de bande passante tandis que le LSP_ID actuel continue d'être rafraîchi pour assurer que la réservation n'est pas perdue si la plus grande réservation échoue.

2.6 MTU de chemin

Le RSVP standard [RFC2205] et Int-Serv [RFC2210] fournissent à l'envoyeur RSVP la MTU minimum disponible entre l'envoyeur et le receveur. Cette capacité d'identification de la MTU de chemin est aussi fournie pour les LSP établis via RSVP.

Les informations sur la MTU de chemin sont portées, selon ce qui est présent, dans les objets Services Intégré ou Service Nul. Lorsque on utilise les objets Services intégré, la MTU de chemin est fournie sur la base des procédures définies dans la [RFC2210]. L'identification de la MTU de chemin lorsque on utilise les objets Service Nul est définie dans la [RFC2997].

Avec RSVP standard, les informations de MTU de chemin sont utilisées par l'envoyeur pour vérifier quels paquets IP excèdent la MTU de chemin. Pour les paquets qui excèdent la MTU de chemin, l'envoyeur fragmente les paquets ou, quand le datagramme IP a le bit "Ne pas fragmenter" (DF, *Do not Fragment*) établi, produit un message ICMP Destination injoignable. Ce traitement de la MTU de chemin est aussi exigé des LSP établis via RSVP.

L'algorithme suivant s'applique à tous les datagrammes IP non étiquetés et à tout paquet étiqueté dont le nœud sait qu'il est un datagramme IP, auxquels des étiquettes doivent être ajoutées avant transmission. Pour les paquets étiquetés, le fond de la pile est trouvé, l'en-tête IP est examiné.

En utilisant la terminologie définie dans la [RFC3032], un LSR DOIT exécuter l'algorithme suivant :

1. Soit N le nombre d'octets dans la pile d'étiquettes (c'est-à-dire, 4 fois le nombre d'entrées de la pile d'étiquettes) y compris les étiquettes à ajouter par ce nœud.

2. Soit M le plus petit de "taille maximum de datagramme IP initialement étiqueté" ou de "MTU de chemin - N".

Lorsque la taille d'un datagramme IPv4 (sans étiquettes) excède la valeur de M,
Si le bit DF n'est pas établi *(pas 1)* dans l'en-tête IPv4, alors
(a) le datagramme DOIT être cassé en fragments, dont la taille d'aucun n'est supérieure à M, et
(b) chaque fragment DOIT être étiqueté avant d'être transmis.

Si le bit DF est établi *(à 1)* dans l'en-tête IPv4, alors
(a) Le datagramme NE DOIT PAS être transmis.
(b) Créer un message ICMP Destination injoignable :
i. régler son champ Code [RFC0792] à "Fragmentation exigée et DF établi",
ii. régler son champ MTU du prochain bond [RFC1191] à M.
(c) Si possible, transmettre le message ICMP Destination injoignable à la source du datagramme éliminé.

Lorsque la taille d'un datagramme IPv6 (sans étiquette) excède la valeur de M,
(a) le datagramme NE DOIT PAS être transmis.
(b) Créer un message ICMP Paquet trop gros avec le champ MTU de liaison du prochain bond [RFC2463] réglé à M.
(c) Si possible, transmettre le message ICMP Paquet trop gros à la source du datagramme éliminé.

3. Formats des messages en rapport avec les LSP tunnels

Cinq nouveaux objets sont définis dans cette section :

Nom de l'objet	**messages RSVP applicables**
LABEL_REQUEST	Path
LABEL	Resv
EXPLICIT_ROUTE	Path
RECORD_ROUTE	Path, Resv
SESSION_ATTRIBUTE	Path

De nouveaux C-Types sont aussi alloués pour les objets SESSION, SENDER_TEMPLATE, et FILTER_SPEC.

Les descriptions détaillées des nouveaux objets sont données dans les paragraphes qui suivent. Tous les nouveaux objets sont FACULTATIFS pour ce qui concerne RSVP. Une mise en œuvre peut choisir de prendre en charge un sous-ensemble de ces objets. Cependant, les objets LABEL_REQUEST et LABEL sont obligatoires pour ce qui concerne la présente spécification.

Les objets LABEL et RECORD_ROUTE sont spécifiques de l'envoyeur. Dans les messages Resv, ils DOIVENT apparaître après la FILTER_SPEC associée et avant toute FILTER_SPEC suivante.

Le placement relatif des objets EXPLICIT_ROUTE, LABEL_REQUEST, et SESSION_ATTRIBUTE est simplement une recommandation. L'ordre de ces objets n'est pas important, de sorte qu'une mise en œuvre DOIT être prête à accepter les objets dans n'importe quel ordre.

3.1 Message Path

Le format du message Path est le suivant :

<Message Path> ::= <En-tête commun> [<INTEGRITY>] <SESSION> <RSVP_HOP> <TIME_VALUES>
[<EXPLICIT_ROUTE>] <LABEL_REQUEST> [<SESSION_ATTRIBUTE>]
[<POLICY_DATA> ...] <descripteur de l'envoyeur>

<descripteur de l'envoyeur> ::= <SENDER_TEMPLATE> <SENDER_TSPEC> [<ADSPEC>] [<RECORD_ROUTE>]

3.2 Message Resv

Le format du message Resv est le suivant :

<Message Resv> ::= <En-tête commun> [<INTEGRITY>] <SESSION> <RSVP_HOP>
<TIME_VALUES> [<RESV_CONFIRM>] [<SCOPE>]
[<POLICY_DATA> ...] <STYLE> <liste de descripteurs de flux>

<liste de descripteurs de flux> ::= <liste de descripteurs de flux FF> | <descripteur de flux SE>

<liste de descripteurs de flux FF> ::= <FLOWSPEC> <FILTER_SPEC> <LABEL> [<RECORD_ROUTE>]
| <liste de descripteurs de flux FF> <descripteur de flux FF>

<descripteur de flux FF> ::= [<FLOWSPEC>] <FILTER_SPEC> <LABEL> [<RECORD_ROUTE>]

<descripteur de flux SE> ::= <FLOWSPEC> <liste de spec de filtre SE>

<liste de spec de filtre SE> ::= < spec de filtre SE> | <liste de spec de filtre SE> < spec de filtre SE>

< spec de filtre SE> ::= <FILTER_SPEC> <LABEL> [<RECORD_ROUTE>]

Note : LABEL et RECORD_ROUTE (s'ils sont présents) sont liés à la FILTER_SPEC précédente. Un seul LABEL et/ou RECORD_ROUTE peut suivre chaque FILTER_SPEC.

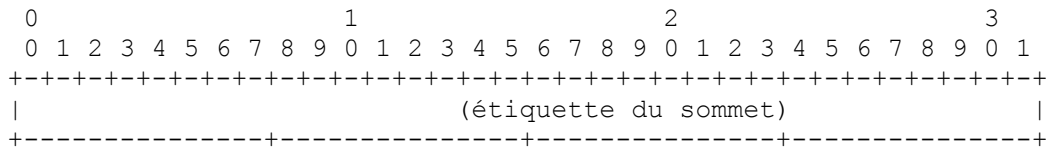
4. Objets en rapport avec le LSP tunnel

4.1 Objet LABEL

Les objets LABEL PEUVENT être portés dans les messages Resv. Pour les styles FF et SE, une étiquette est associée à chaque envoyeur. L'étiquette pour un envoyeur DOIT suivre immédiatement la FILTER_SPEC pour cet envoyeur dans le message Resv.

L'objet LABEL a le format suivant

LABEL : classe = 16, C_Type = 1



Le contenu de LABEL est une seule étiquette, codée sur 4 octets. Chaque étiquette MPLS générique est un entier non signé dans la gamme de 0 à 1 048 575. Les étiquettes MPLS génériques et les étiquettes FR sont codées en alignement à droite sur 4 octets. Les étiquettes ATM sont codées avec le VPI justifié à droite sur les bits 0-15 et le VCI justifié à droite sur les bits 16-31.

4.1.1 Traitement des objets LABEL dans les messages Resv

En MPLS, un nœud peut prendre en charge plusieurs espaces d'étiquettes, associant peut-être un espace unique à chaque interface entrante. Pour les besoins de l'exposé qui suit, le terme "même étiquette" signifie la valeur d'étiquette identique tirée de l'espace d'étiquettes identique. De plus, ce qui suit ne s'applique qu'aux sessions en envoi individuel.

Les étiquettes reçues dans les messages Resv sur des interfaces différentes sont toujours considérées comme différentes même si la valeur de l'étiquette est la même.

4.1.1.1 Vers l'aval

Le nœud aval choisit une étiquette pour représenter le flux. Si une gamme d'étiquettes a été spécifiée dans la demande d'étiquette, l'étiquette DOIT être tirée de cette gamme. Si aucune étiquette n'est disponible, le nœud envoie un message PathErr avec un code d'erreur de "problème d'acheminement" et une valeur d'erreur de "échec d'allocation d'étiquette".

Si un nœud reçoit un message Resv qui a alloué la même valeur d'étiquette à plusieurs envoyeurs, ce nœud PEUT alors aussi allouer une seule valeur à ces mêmes envoyeurs ou à tout sous-ensemble de ces envoyeurs. Noter que si un nœud prévoit de réguler les envoyeurs individuels pour une session, il DOIT allouer des étiquettes uniques à ces envoyeurs.

Dans le cas d'ATM, une condition supplémentaire s'applique. Certains nœuds ATM ne sont pas capables de fusionner les flux. Ces nœuds PEUVENT indiquer cela en réglant un bit à zéro dans la demande d'étiquette. Dans l'objet LABEL_REQUEST de C-Type 2, dans une demande d'étiquette dans la gamme d'étiquettes ATM, le bit M *(Merge)* sert à cela. Le bit M DEVRAIT être établi par les nœuds qui ont la capacité de fusion. Si pour un envoyeur, le bit M n'est pas établi, le nœud aval DOIT allouer une étiquette unique à cet envoyeur.

Une fois qu'une étiquette est allouée, le nœud formate un nouvel objet LABEL. Le nœud envoie alors au bond précédent le nouvel objet LABEL au titre du message Resv. Le nœud DEVRAIT être prêt à transmettre des paquets qui portent l'étiquette allouée avant d'envoyer le message Resv. L'objet LABEL DEVRAIT être gardé dans le bloc d'état de réservation. Il sera alors utilisé dans le prochain événement de rafraîchissement de réservation pour formater le message Resv.

Un nœud est supposé envoyer un message Resv avant que son temporisateur de rafraîchissement n'arrive à expiration si le contenu de l'objet LABEL change.

4.1.1.2 Vers l'amont

Un nœud utilise l'étiquette portée dans l'objet LABEL comme étiquette sortante associée à l'envoyeur. Le routeur alloue une nouvelle étiquette et la lie à l'interface entrante de cette session/envoyeur. C'est la même interface qu'utilise le routeur pour transmettre les messages Resv aux bonds précédents.

Plusieurs circonstances peuvent conduire à une étiquette inacceptable :

1. le nœud est un commutateur ATM incapable de fusion mais le nœud aval a alloué la même étiquette à deux envoyeurs,
2. l'étiquette nulle implicite a été allouée mais le nœud n'est pas capable de faire un avant-dernier saut pour le L3PID associé,
3. l'étiquette allouée est en dehors de la gamme d'étiquettes demandée.

Dans ces trois cas, le nœud envoie un message ResvErr avec un code d'erreur de "problème d'acheminement" et une valeur d'erreur de "valeur d'étiquette inacceptable".

4.1.2 Non prise en charge de l'objet LABEL

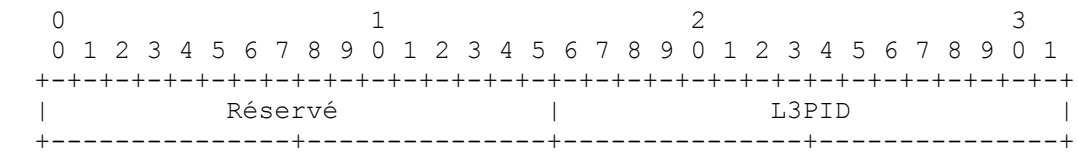
Dans des circonstances normales, un nœud ne devrait jamais recevoir un objet LABEL dans un message Resv sauf si il a inclus un objet LABEL_REQUEST dans le message Path correspondant. Cependant, un routeur RSVP qui ne reconnaît pas l'objet LABEL envoie une ResvErr avec le code d'erreur "Classe d'objet inconnue" au receveur. Cela cause l'échec de la réservation.

4.2 Objet LABEL_REQUEST

La classe de LABEL_REQUEST est 19. Actuellement, il y a trois C_Types possibles. Le type 1 est une demande d'étiquette sans gamme d'étiquettes. Le type 2 est une demande d'étiquette dans une gamme d'étiquettes ATM. Le type 3 est une demande d'étiquette avec une gamme d'étiquettes de relais de trame. Les formats d'objet LABEL_REQUEST sont donnés ci-dessous.

4.2.1 Demande d'étiquette sans gamme d'étiquettes

Classe = 19, C_Type = 1

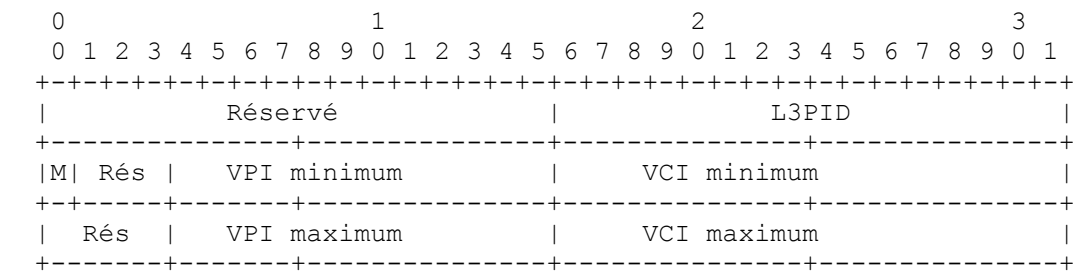


Réservé
Ce champ est réservé. Il DOIT être réglé à zéro à l'émission et DOIT être ignoré à réception.

L3PID *(Layer 3 Protocol Identifier)*
C'est l'identifiant du protocole de couche 3 qui utilise ce chemin. Des valeurs d'Ethertype standard sont utilisées.

4.2.2 Demande d'étiquette avec gamme d'étiquette ATM

Classe = 19, C_Type = 2



Réservé (Rés)
Ce champ est réservé. Il DOIT être réglé à zéro à l'émission et DOIT être ignoré à réception.

L3PID
C'est l'identifiant du protocole de couche 3 qui utilise ce chemin. Des valeurs d'Ethertype standard sont utilisées.

M
Ce bit réglé à un indique que le nœud est capable de fusion dans le plan des données.

VPI minimum (12 bits)
Ce champ de 12 bits spécifie la limite inférieure d'un bloc d'identifiants de chemins virtuels qui est pris en charge sur le commutateur d'origine. Si le VPI fait moins de 12 bits, il DOIT être justifié à droite dans ce champ et les bits précédents DOIVENT être réglés à zéro.

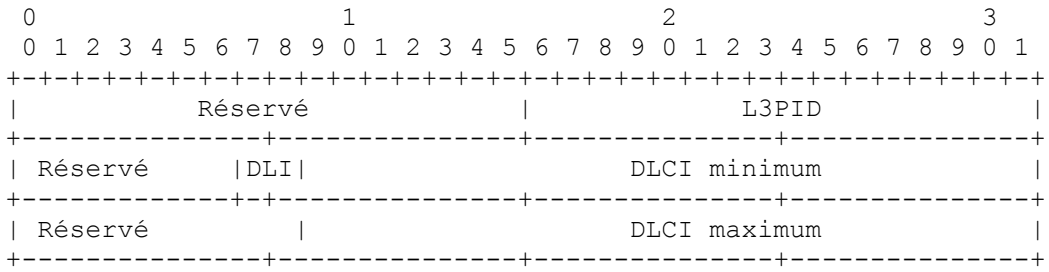
VCI minimum (16 bits)
Ce champ de 16 bits spécifie la limite inférieure d'un bloc d'identifiants de connexion virtuelle qui est pris en charge sur le commutateur d'origine. Si le VCI fait moins de 16 bits, il DOIT être justifié à droite dans ce champ et les bits précédents DOIVENT être mis à zéro.

VPI maximum (12 bits)
Ce champ de 12 bits spécifie la limite supérieure d'un bloc d'identifiants de chemin virtuel qui est pris en charge sur le commutateur d'origine. Si le VPI fait moins de 12 bits, il DOIT être justifié à droite dans ce champ et les bits précédents DOIVENT être mis à zéro.

VCI maximum (16 bits)
Ce champ de 16 bits spécifie la limite supérieure d'un bloc d'identifiants de connexion virtuelle qui est pris en charge par le commutateur d'origine. Si le VCI fait moins de 16 bits, il DOIT être justifié à droite dans ce champ et les bits précédents DOIVENT être réglés à zéro.

4.2.3 Demande d'étiquette avec gamme d'étiquettes de relais de trame

Classe = 19, C_Type = 3



Réservé
Ce champ est réservé. Il DOIT être réglé à zéro à l'émission et ignoré à réception.

L3PID
C'est l'identifiant du protocole de couche 3 qui utilise ce chemin. Des valeurs d'Ethertype standard sont utilisées.

DLI *(DLCI Length Indicator)*
Indicateur de longueur de DLCI. C'est le nombre de bits du DLCI. Les valeurs suivantes sont acceptées :

DLI	**bits du DLCI**
0	10
2	23

DLCI minimum
Ce champ de 23 bits spécifie la limite inférieure d'un bloc d'identifiants de connexion de liaison de données (DLCI, *Data Link Connection Identifier*) qui est pris en charge sur le commutateur d'origine. Le DLCI DOIT être justifié à droite dans ce champ et les bits inutilisés DOIVENT être réglés à 0.

DLCI maximum
Ce champ de 23 bits spécifie la limite supérieure d'un bloc d'identifiants de connexion de liaison de données (DLCI) qui est pris en charge sur le commutateur d'origine. Le DLCI DOIT être justifié à droite dans ce champ et les bits inutilisés DOIVENT être réglés à 0.

4.2.4 Traitement de LABEL_REQUEST

Pour établir un LSP tunnel, l'envoyeur crée un message Path avec un objet LABEL_REQUEST. L'objet LABEL_REQUEST indique qu'un lien d'étiquette est demandé pour ce chemin et donne une indication du protocole de couche réseau qui sera porté sur ce chemin. Cela permet aux protocoles de couche réseau non IP d'être envoyés sur un LSP. Ces informations peuvent aussi être utiles pour l'allocation réelle d'étiquette, parce que certaines étiquettes réservées sont spécifiques du protocole, voir la [RFC3032].

L'objet LABEL_REQUEST DEVRAIT être mémorisé dans le bloc d'état de chemin, afin que les messages de rafraîchissement de chemin contiennent aussi l'objet LABEL_REQUEST. Lorsque le message Path atteint le receveur, la présence de l'objet LABEL_REQUEST déclenche l'allocation d'une étiquette par le receveur et le placement de l'étiquette dans l'objet LABEL pour le message Resv correspondant. Si une gamme d'étiquettes était spécifiée, l'étiquette DOIT être allouée dans cette gamme. Un receveur qui accepte un objet LABEL_REQUEST DOIT inclure un objet LABEL dans les messages Resv qui relèvent de ce message Path. Si un objet LABEL_REQUEST n'était pas présent dans le message Path, un nœud NE DOIT PAS inclure un objet LABEL dans un message Resv pour la session et le PHOP de ce message Path.

Un nœud qui envoie un objet LABEL_REQUEST DOIT être prêt à accepter et traiter correctement un objet LABEL dans les messages Resv correspondants.

Un nœud qui reconnaît un objet LABEL_REQUEST, mais qui est incapable de le prendre en charge (éventuellement parce qu'il ne réussit pas à allouer des étiquettes) DEVRAIT envoyer un PathErr avec le code d'erreur "Problème d'acheminement" et la valeur d'erreur "Échec d'allocation d'étiquette MPLS". Cela inclut le cas où une gamme d'étiquettes a été spécifiée et où une étiquette ne peut pas être allouée dans cette gamme.

Un nœud qui reçoit et transmet un message Path avec un objet LABEL_REQUEST DOIT copier le L3PID de l'objet LABEL_REQUEST reçu dans l'objet LABEL_REQUEST transmis.

Si le receveur ne peut pas prendre en charge le L3PID du protocole, il DEVRAIT envoyer un PathErr avec le code d'erreur "Problème d'acheminement" et la valeur d'erreur "L3PID non pris en charge". Cela cause l'échec de la session RSVP.

4.2.5 Non prise en charge de l'objet LABEL_REQUEST

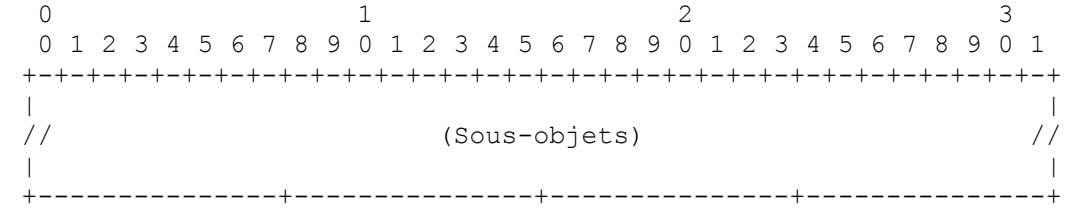
Un routeur RSVP qui ne reconnaît pas l'objet LABEL_REQUEST envoie un PathErr avec le code d'erreur "Classe d'objet inconnue" à l'envoyeur. Un routeur RSVP qui reconnaît l'objet LABEL_REQUEST mais ne reconnaît pas le C_Type envoie un PathErr avec le code d'erreur "C_Type d'objet inconnu" à l'envoyeur. Cela cause l'échec de l'établissement du chemin. L'envoyeur devrait notifier au gestionnaire qu'un LSP ne peut pas être établi et éventuellement prendre des mesures pour continuer la réservation sans la LABEL_REQUEST.

RSVP est conçu pour traiter en douceur les routeurs non RSVP partout entre les envoyeurs et les receveurs. Cependant, il est évident que les routeurs non RSVP ne peuvent pas porter les étiquettes via RSVP. Cela signifie que si un routeur a un voisin dont il sait qu'il n'est pas à capacité RSVP, ce routeur NE DOIT PAS annoncer l'objet LABEL_REQUEST quand il envoie des messages qui passent à travers des routeurs non RSVP. Le routeur DEVRAIT renvoyer un PathErr à l'envoyeur, avec le code d'erreur "Problème d'acheminement" et la valeur d'erreur "MPLS en négociation, mais un routeur non RSVP est sur le chemin". Ce même message DEVRAIT être envoyé si un routeur reçoit un objet LABEL_REQUEST dans un message provenant d'un routeur sans capacité RSVP. Voir dans la [RFC2205] une description de la façon dont un routeur aval peut déterminer la présence de routeurs non RSVP.

4.3 Objet EXPLICIT_ROUTE

Les chemins explicites sont spécifiés via l'objet EXPLICIT_ROUTE (ERO). La classe de EXPLICIT_ROUTE est 20. Actuellement un C_Type est défini : le chemin explicite de type 1. L'objet EXPLICIT_ROUTE a le format suivant :

Classe = 20, C_Type = 1



Sous-objets
Le contenu d'un objet EXPLICIT_ROUTE est une série d'éléments de données de longueur variable appelés sous-objets. Les sous-objets sont définis au paragraphe 4.3.3.

Si un message Path contient plusieurs objets EXPLICIT_ROUTE, seul le premier objet est significatif. Les objets EXPLICIT_ROUTE suivants PEUVENT être ignorés et NE DEVRAIENT PAS être propagés.

4.3.1 Applicabilité

L'objet EXPLICIT_ROUTE est destiné à n'être utilisé que pour des situations d'envoi individuel. Les applications d'acheminement explicite en diffusion groupée sont un sujet pour des recherches futures.

L'objet EXPLICIT_ROUTE est à n'utiliser que lorsque tous les routeurs le long du chemin explicite prennent en charge RSVP et l'objet EXPLICIT_ROUTE. L'objet EXPLICIT_ROUTE a reçu une valeur de classe de la forme 0bbbbbbb. Les routeurs RSVP qui ne prennent pas l'objet en charge répondront donc par une erreur "Classe d'objet inconnue".

4.3.2 Sémantique de l'objet EXPLICIT_ROUTE

Un chemin explicite est un chemin particulier dans la topologie du réseau. Normalement, le chemin explicite est déterminé par un nœud, avec l'intention de diriger le trafic le long de ce chemin. Un chemin explicite est décrit comme une liste de groupes de nœuds le long du chemin explicite. En plus de la capacité à identifier des nœuds spécifiques le long du chemin, un chemin explicite peut identifier un groupe de nœuds qui doivent être traversés le long du chemin. Cette capacité permet au système d'acheminement une souplesse locale significative pour satisfaire une demande de chemin explicite. Cette capacité permet au générateur du chemin explicite d'avoir des informations imparfaites sur les détails du chemin.

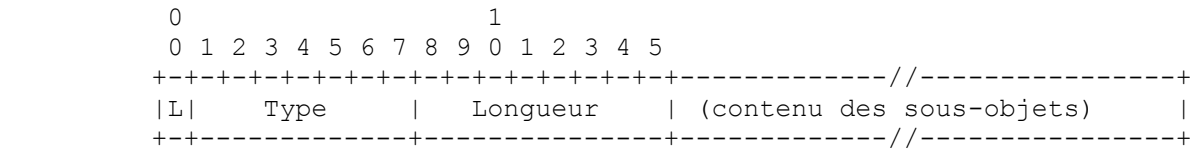
Le chemin explicite est codé comme une série de sous-objets contenus dans un objet EXPLICIT_ROUTE. Chaque sous-objet identifie un groupe de nœuds dans le chemin explicite. Un chemin explicite est donc une spécification des groupes de nœuds à traverser.

Pour formaliser la discussion, on appelle chaque groupe de nœud un nœud abstrait. Donc, on dit qu'un chemin explicite est une spécification d'un ensemble de nœuds abstraits à traverser. Si un nœud abstrait consiste en seulement un nœud, on dit que c'est un nœud abstrait simple.

Comme exemple du concept de nœud abstrait, considérons un chemin explicite qui consiste seulement en un certain nombre d'objets Système autonome. Chaque sous-objet correspond à un système autonome dans la topologie globale. Dans ce cas, chaque système autonome est un nœud abstrait, et le chemin explicite est un chemin qui inclut chacun des systèmes autonomes spécifiés. Il peut y avoir plusieurs bonds au sein de chaque système autonome, mais ils sont opaques au nœud source pour le chemin explicite.

4.3.3 Sous-objets

Le contenu d'un objet EXPLICIT_ROUTE est une série d'éléments de données de longueur variable appelés des sous-objets. Chaque sous-objet a la forme suivante.



L : Le bit L est un attribut du sous-objet. Le bit L est établi si le sous-objet représente un bond lâche dans le chemin explicite. Si le bit n'est pas établi, le sous-objet représente un bond strict dans le chemin explicite.

Type : Le type indique le type de contenu du sous-objet. Les valeurs définies actuellement sont :

1 Préfixe IPv4
2 Préfixe IPv6
32 Numéro de système autonome

Longueur : La longueur contient la longueur totale du sous-objet en octets, incluant les champs L, Type et Longueur. La longueur DOIT être au moins 4, et DOIT être un multiple de 4.

4.3.3.1 Sous-objets stricts et lâches

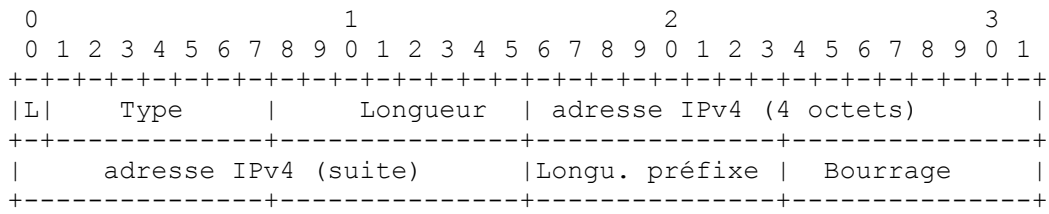
Le bit L dans le sous-objet est un attribut d'un bit. Si le bit L est établi, alors la valeur de l'attribut est 'lâche'. Autrement, la valeur de l'attribut est 'strict'. Pour faire court, on dira que si la valeur de l'attribut du sous-objet est 'lâche' alors c'est un sous-objet 'lâche'. Autrement, c'est un sous-objet 'strict'. De plus, on dira que le nœud abstrait d'un sous-objet strict ou

lâche est un nœud respectivement strict ou lâche. Les nœuds stricts sont toujours interprétés par rapport à leur nœud abstrait précédent.

Le chemin entre un nœud strict et son nœud précédent DOIT inclure seulement les nœuds de réseau du nœud strict et de son nœud abstrait précédent.

Le chemin entre un nœud lâche et son nœud précédent PEUT inclure d'autres nœuds de réseau qui ne font pas partie du nœud lâche ou de son nœud abstrait précédant.

4.3.3.2 Sous-objet 1 : préfixe IPv4



L : Le bit L est un attribut du sous-objet. Il est établi si le sous-objet représente un bond lâche dans le chemin explicite. Si le bit est à zéro, le sous-objet représente un bond strict dans le chemin explicite.

Type : 0x01 : adresse IPv4

Longueur : Longueur contient la longueur totale du sous-objet en octets, incluant les champs Type et Longueur. La longueur est toujours 8.

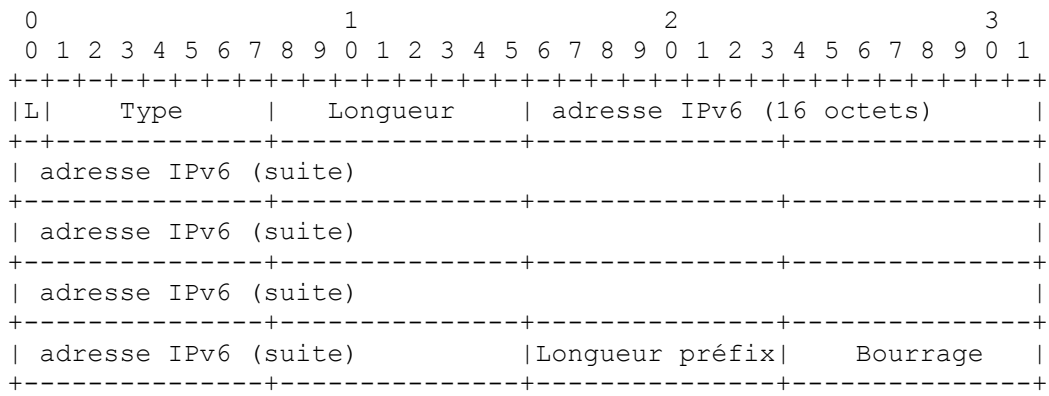
Adresse IPv4 : Une adresse IPv4. Cette adresse est traitée comme un préfixe sur la base de la valeur de longueur de préfixe ci-dessous. Les bits au-delà du préfixe sont ignorés à la réception et DEVRAIENT être réglés à zéro à l'émission.

Longueur de préfixe : Longueur en bits du préfixe IPv4

Bourrage : Zéro à l'émission. Ignoré à réception.

Le contenu d'un sous-objet préfixe IPv4 est une adresse IPv4 de 4 octets, une longueur de préfixe de 1 octet, et un bourrage de 1 octet. Le nœud abstrait représenté par ce sous-objet est l'ensemble des nœuds qui ont une adresse IP qui se tient dans ce préfixe. Noter qu'une longueur de préfixe de 32 indique un seul nœud IPv4.

4.3.3.3 Sous objet 2 : Préfixe IPv6



L : Le bit L est un attribut du sous-objet. Le bit L est mis à 1 si le sous-objet représente un bond lâche dans le chemin explicite. Si le bit est à zéro, le sous-objet représente un bond strict dans le chemin explicite.

Type : 0x02 : adresse IPv6

Longueur : Longueur contient la longueur totale du sous-objet en octets, incluant les champs Type et Longueur. La longueur est toujours 20.

Adresse IPv6 : Une adresse IPv6. Cette adresse est traitée comme un préfixe sur la base de la valeur de longueur de préfixe ci-dessous. Les bits au-delà du préfixe sont ignorés à réception et DEVRAIENT être mis à zéro à l'émission.

Longueur de préfixe : Longueur en bits du préfixe IPv6.

Bourrage : Zéro à l'émission. Ignoré à réception.

Le contenu d'un sous-objet préfixe IPv6 est une adresse IPv6 de 16 octets, une longueur de préfixe de 1 octet, et un octet de bourrage. Le nœud abstrait représenté par ce sous-objet est l'ensemble des nœuds qui ont une adresse IP qui se tient dans ce préfixe. Noter qu'une longueur de préfixe de 128 indique un seul nœud IPv6.

4.3.3.4 Sous-objet 32 : Numéro de système autonome

Le contenu d'un sous-objet Numéro de système autonome (AS) est un numéro d'AS de 2 octets. Le nœud abstrait représenté par ce sous-objet est l'ensemble des nœuds qui appartiennent au système autonome.

La longueur du sous-objet Numéro d'AS est 4 octets.

4.3.4 Traitement de l'objet EXPLICIT_ROUTE

4.3.4.1 Choix du prochain bond

Un nœud qui reçoit un message Path contenant un objet EXPLICIT_ROUTE doit déterminer le prochain bond pour ce chemin. Cela est nécessaire parce que le prochain nœud abstrait le long du chemin explicite pourrait être un sous réseau IP ou un système autonome. Donc, le choix de ce prochain bond peut impliquer une décision de choix parmi un ensemble de solutions possibles. Les critères utilisés pour faire un choix parmi plusieurs solutions possibles dépendent de la mise en œuvre et peuvent aussi être impactés par la politique locale, et cela sort du domaine d'application de la présente spécification. Cependant, on suppose que chaque nœud va faire un effort au mieux pour déterminer un chemin sans boucle. Noter que les chemins ainsi déterminés peuvent être outrepassés par la politique locale.

Pour déterminer le prochain bond pour le chemin, un nœud suit les étapes ci-après :

1) Le nœud qui reçoit le message RSVP DOIT d'abord évaluer le premier sous-objet. Si le nœud ne fait pas partie du nœud abstrait décrit par le premier sous-objet, il a reçu le message par erreur et DEVRAIT retourner une erreur "Mauvais sous-objet initial". Si il n'y a pas de premier sous-objet, le message est aussi une erreur et le système DEVRAIT retourner "Mauvais objet EXPLICIT_ROUTE".

2) Si il n'y a pas de second sous-objet, cela indique la fin du chemin explicite. L'objet EXPLICIT_ROUTE DEVRAIT être retiré du message Path. Ce nœud peut être ou non la fin du chemin. Le traitement continue au paragraphe 4.3.4.2 où un nouvel objet EXPLICIT_ROUTE PEUT être ajouté au message Path.

3) Ensuite, le nœud évalue le second sous-objet. Si le nœud fait aussi partie du nœud abstrait décrit par le second sous-objet, alors le nœud supprime le premier sous-objet et continue le traitement à l'étape 2 ci-dessus. Noter que cela fait du second sous-objet le premier sous-objet de la prochaine itération et permet au nœud d'identifier le prochain nœud abstrait sur le chemin du message après une éventuelle répétition de l'application des étapes 2 et 3.

4) Cas de la bordure du nœud abstrait : le nœud détermine si il est topologiquement adjacent au nœud abstrait décrit par le second sous-objet. Si il l'est, le nœud choisit un prochain bond particulier qui est membre du nœud abstrait. Le nœud supprime alors le premier sous-objet et continue le traitement avec le paragraphe 4.3.4.2.

5) Cas de l'intérieur du nœud abstrait : autrement, le nœud choisit un prochain bond au sein du nœud abstrait du premier sous-objet (auquel appartient le nœud) qui est le long du chemin vers le nœud abstrait du second sous-objet (qui est le prochain nœud abstrait). Si un tel chemin n'existe pas, il y a alors deux cas :
   5a) Si le second sous-objet est un sous-objet strict, il y a une erreur et le nœud DEVRAIT retourner l'erreur "Mauvais nœud strict ".
   5b) Autrement, si le second sous-objet est un sous-objet lâche, le nœud choisit un prochain bond qui se trouve sur le chemin du prochain nœud abstrait. Si un tel chemin n'existe pas, il y a une erreur, et le nœud DEVRAIT retourner une erreur "Mauvais nœud lâche".

6) Finalement, le nœud remplace le premier sous-objet par tout sous-objet qui affiche un nœud abstrait contenant le prochain bond. Cela est nécessaire afin que lorsque le chemin explicite sera reçu par le prochain bond, il soit accepté.

4.3.4.2 Ajout de sous-objets à l'objet EXPLICIT_ROUTE

Après le choix d'un prochain bond, le nœud PEUT modifier le chemin explicite de la façon suivante.

Si, au titre de l'exécution de l'algorithme du paragraphe 4.3.4.1, l'objet EXPLICIT_ROUTE est retiré, le nœud PEUT ajouter un nouvel objet EXPLICIT_ROUTE.

Autrement, si le nœud est membre du nœud abstrait pour le premier sous-objet, une série de sous-objets PEUT être insérée avant le premier sous-objet ou PEUT remplacer le premier sous-objet. Chaque sous-objet dans cette série DOIT afficher un nœud abstrait qui soit un sous ensemble du nœud abstrait actuel.

Autrement, si le premier sous-objet est un sous-objet lâche, une série arbitraire de sous-objets PEUT être insérée avant le premier sous-objet.

4.3.5 Boucles

Alors que l'objet EXPLICIT_ROUTE est de longueur finie, l'existence de nœuds lâches implique qu'il soit possible de construire des boucles de transmission durant des périodes transitoires dans le protocole d'acheminement sous-jacent. Cela peut être détecté par l'origine du chemin explicite grâce à l'utilisation d'un autre objet de chemin opaque appelé l'objet RECORD_ROUTE. L'objet RECORD_ROUTE est utilisé pour collecter des informations détaillées de chemin et il est utile pour la détection des boucles et pour les diagnostics.

4.3.6 Compatibilité de transmission

Il est prévu que de nouveaux sous-objets puissent être définis plus tard. Un nœud qui rencontre un sous-objet non reconnu durant son traitement d'ERO normal envoie une PathErr avec le code d'erreur "Erreur d'acheminement" et la valeur d'erreur de "Mauvais objet EXPLICIT_ROUTE" à l'envoyeur. L'objet EXPLICIT_ROUTE est inclus, tronqué (à gauche) au niveau du sous-objet en cause. La présence d'un sous-objet non reconnu qui n'est pas rencontré dans le traitement d'ERO d'un nœud DEVRAIT être ignoré. Il est passé vers l'avant avec le reste de la pile d'ERO restante.

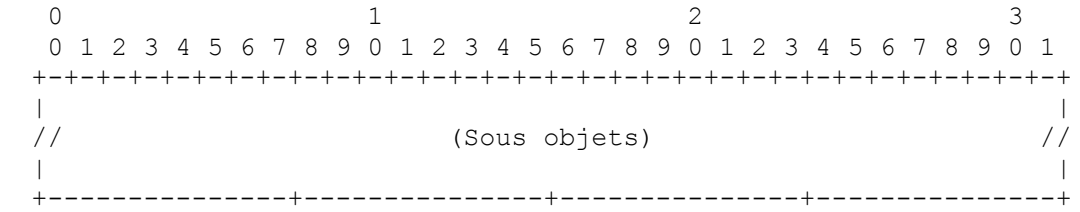
4.3.7 Non prise en charge de l'objet EXPLICIT_ROUTE

Un routeur RSVP qui ne reconnaît pas l'objet EXPLICIT_ROUTE envoie une PathErr avec le code d'erreur "Classe d'objet inconnue" à l'envoyeur. Cela cause l'échec de l'établissement du chemin. L'envoyeur devrait notifier au gestionnaire qu'un LSP ne peut pas être établi et éventuellement prendre des mesures pour continuer la réservation sans le EXPLICIT_ROUTE ou via un chemin explicite différent.

4.4 Objets RECORD_ROUTE

Les chemins peuvent être enregistrés via l'objet RECORD_ROUTE (RRO, *RECORD_ROUTE object*). Facultativement, les étiquettes peuvent aussi être enregistrées. La classe de RECORD_ROUTE est 21. Un C_Type est actuellement défini : Record Route Type 1. L'objet RECORD_ROUTE a le format suivant :

Classe = 21, C_Type = 1



Sous-objets : Le contenu d'un objet RECORD_ROUTE est une série d'éléments de données de longueur variable appelés sous-objets. Les sous-objets sont définis au paragraphe 4.4.1.

Le RRO peut être présent aussi bien dans le message RSVP Path que Resv. Si un message Path contient plusieurs RRO, seul le premier RRO est significatif. Les RRO suivants DEVRAIENT être ignorés et NE DEVRAIENT PAS être propagés.

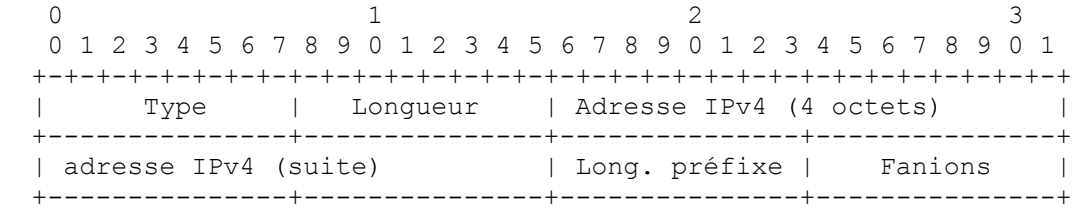
De même, si dans un message Resv plusieurs RRO sont rencontrés à la suite d'une FILTER_SPEC avant qu'une autre FILTER_SPEC soit rencontrée, seul le premier RRO est significatif. Les RRO suivants DEVRAIENT être ignorés et NE DEVRAIENT PAS être propagés.

4.4.1 Sous-objets

Le contenu d'un objet RECORD_ROUTE est une série d'éléments de données de longueur variable appelés sous-objets. Chaque sous-objet a son propre champ Longueur. La longueur contient la longueur totale du sous-objet en octets, incluant les champs Type et Longueur. La longueur DOIT toujours être un multiple de 4, et au moins 4.
Les sous-objets sont organisés comme une pile au dernier entré premier sorti. Le premier sous-objet par rapport au début d'un RRO est considéré comme le sommet. Le dernier sous-objet est considéré comme le bas. Lorsque un nouveau sous-objet est ajouté, il est toujours ajouté au sommet.
Un RRO vide sans sous-objet est considéré comme illégal.

Trois sortes de sous-objets sont actuellement définis.

4.4.1.1 Sous-objet 1 : adresse IPv4



Type : 0x01 : adresse IPv4

Longueur : Le champ Longueur contient la longueur totale du sous-objet en octets, incluant les champs Type et Longueur. Longueur est toujours 8.

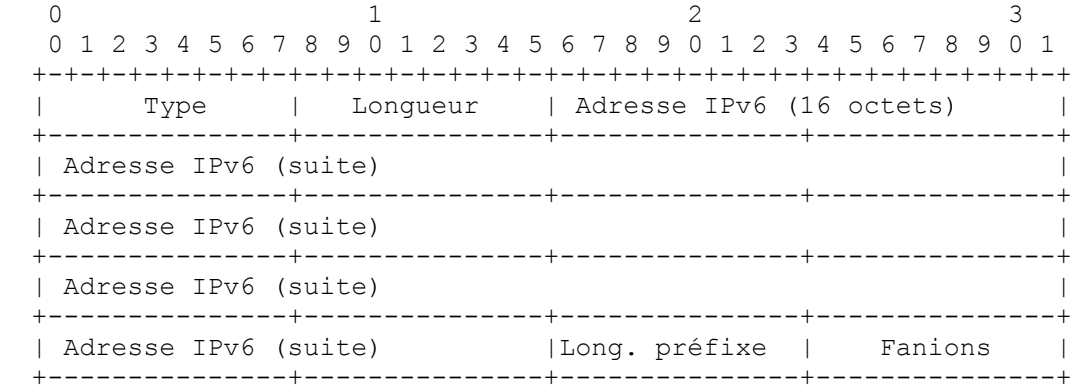
Adresse IPv4 : Adresse d'hôte en envoi individuel de 32 bits. Toute adresse d'interface accessible par le réseau est admise ici. Les adresses illégales, telles que certaines adresses de repli, NE DEVRAIENT PAS être utilisées.

Longueur de préfixe : 32

Fanions
0x01 : Protection locale disponible. Indique que la liaison aval de ce nœud est protégée via un mécanisme de réparation local. Ce fanion ne peut être établi que si le fanion Protection locale était établi dans l'objet SESSION_ATTRIBUTE du message Path correspondant.

0x02 : Protection locale utilisée. Indique qu'un mécanisme local de réparation est utilisé pour entretenir ce tunnel (normalement en présence d'une panne de la liaison sur laquelle il était précédemment acheminé).

4.4.1.2 Sous-objet 2 : Adresses IPv6



Type
0x02 : Adresse IPv6

Longueur : Longueur contient la longueur totale du sous-objet en octets, incluant les champs Type et Longueur. Longueur est toujours 20.

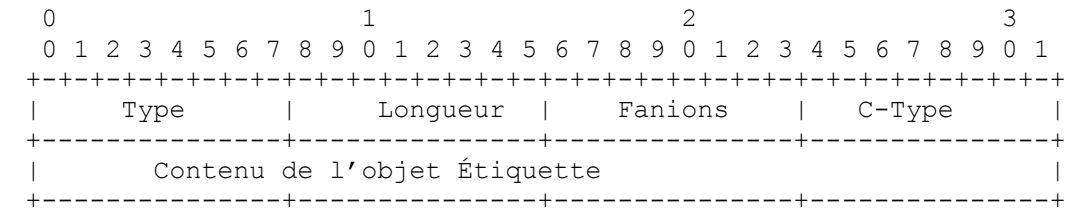
Adresse IPv6 : Adresse d'hôte en envoi individuel de 128 bits.

Longueur de préfixe : 128

Fanions
0x01 : Protection locale disponible. Indique que la liaison aval de ce nœud est protégée via un mécanisme de réparation local. Ce fanion ne peut être établi que si le fanion Protection locale était établi dans l'objet SESSION_ATTRIBUTE du message Path correspondant.

0x02 : Protection locale utilisée. Indique qu'un mécanisme local de réparation est utilisé pour entretenir ce tunnel (normalement en présence d'une panne de la liaison sur laquelle il était précédemment acheminé).

4.4.1.3 Sous-objet 3 : Étiquette



Type
0x03 : Étiquette

Longueur
Longueur contient la longueur totale du sous-objet en octets, incluant les champs Type et Longueur.

Fanions
0x01 = Étiquette globale
Ce fanion indique que l'étiquette sera comprise si elle est reçue sur n'importe quelle interface.

C-Type : Le C-Type de l'objet Étiquette inclus. Copié de l'objet Étiquette.

Contenu de l'objet Étiquette
C'est le contenu de l'objet Étiquette. Copié de l'objet Étiquette.

4.4.2 Applicabilité

Seules les procédures à utiliser dans les sessions en envoi individuel sont définie ici.

Il y a trois utilisations possibles de RRO dans RSVP. D'abord, un RRO peut fonctionner comme mécanisme de détection de boucle pour découvrir des boucles d'acheminement de couche 3, ou des boucles inhérentes au chemin explicite. La procédure exacte pour faire cela est décrite plus loin dans ce document.

Ensuite, un RRO collecte des informations détaillées et à jour sur le chemin bond par bond concernant les sessions RSVP, qui fournissent des informations précieuses pour l'envoyeur ou le receveur. Tout changement de chemin (dû à des changements de la topologie du réseau) va faire l'objet d'un rapport.

Enfin, la syntaxe de RRO est conçue de telle sorte que, avec des changements mineurs, la totalité de l'objet peut être utilisée comme entrée à l'objet EXPLICIT_ROUTE. Cela est utile si l'envoyeur reçoit un RRO du receveur dans un message Resv, l'applique à l'objet EXPLICIT_ROUTE dans le prochain message Path afin de "faire un marquage du chemin de session".

4.4.3 Traitement du RRO

Normalement, un nœud initie une session RSVP en ajoutant le RRO au message Path. Le RRO initial ne contient qu'un sous-objet – l'adresse IP de l'envoyeur. Si le nœud désire aussi l'enregistrement d'étiquette, il établit le fanion Label_Recording dans l'objet SESSION_ATTRIBUTE.

Lorsque un message Path contenant un RRO est reçu par un routeur intermédiaire, celui-ci en mémorise une copie dans le bloc d'état de chemin. Le RRO est alors utilisé dans le prochain événement de rafraîchissement de chemin pour formater les messages Path. Lorsque un nouveau message Path est à envoyer, le routeur ajoute un nouveau sous-objet au RRO et ajoute le RRO résultant au message Path avant la transmission.

Le nouveau sous-objet ajouté DOIT être l'adresse IP de ce routeur. L'adresse à ajouter DEVRAIT être l'adresse de l'interface des messages Path sortants. Si il y a plusieurs adresses entre lesquelles choisir, la décision est une affaire locale. Cependant, il est RECOMMANDÉ que la même adresse soit choisie de façon cohérente.

Lorsque le fanion Label_Recording est établi dans l'objet SESSION_ATTRIBUTE, les nœuds qui font l'enregistrement de chemin DEVRAIENT inclure un sous-objet Enregistrement d'étiquette. Si le nœud utilise un espace d'étiquette mondial, il DEVRAIT alors établir le fanion Global Label.

Le sous-objet Label Record est poussé sur l'objet RECORD_ROUTE avant de pousser l'adresse IP du nœud. Un nœud NE DOIT PAS pousser un sous-objet Label Record sans pousser aussi un sous-objet IPv4 ou IPv6.

Noter qu'à réception du message Path initial, un nœud n'aura probablement pas d'étiquette à inclure. Une fois qu'une étiquette est obtenue, le nœud DEVRAIT inclure l'étiquette dans le RRO dans le prochain événement de rafraîchissement de chemin.

Si le nouveau sous-objet ajouté est cause que le RRO est trop gros pour tenir dans un message Path (ou Resv) l'objet RRO DEVRA être ôté du message et le traitement du message continuera comme d'habitude. Un message PathErr (ou ResvErr) DEVRAIT être renvoyé à l'envoyeur (ou au receveur). Un code d'erreur de "Notifier" et une valeur d'erreur de "RRO trop gros pour la MTU" est utilisé. Si le receveur reçoit une telle ResvErr, il DEVRAIT envoyer un message PathErr avec un code d'erreur de "Notifier" et une valeur d'erreur de "Notification de RRO".

Un envoyeur qui reçoit une de ces valeurs d'erreur DEVRAIT retirer le RRO du message Path.

Les nœuds DEVRAIENT envoyer à nouveau le message PathErr ou ResvErr ci-dessus toutes les n secondes où n est le plus grand de 15 et de l'intervalle de rafraîchissement pour le message Path ou RESV associé. Le nœud PEUT appliquer des limites et/ou des temporisateurs de retardement pour limiter le nombre de messages envoyés.

Un routeur RSVP peut décider d'envoyer des messages Path avant son délai de rafraîchissement si le RRO dans le prochain message Path est différent du précédent. Cela peut arriver si le contenu du RRO reçu du routeur du bond précédent change ou si ce RRO vient d'être ajouté au (ou supprimé du) message Path.

Lorsque le nœud de destination d'une session RSVP reçoit un message Path avec un RRO, cela indique que le nœud envoyeur a besoin d'enregistrement de chemin. Le nœud de destination initie le traitement du RRO en ajoutant un RRO au messages Resv. Le traitement reflète celui des messages Path. La seule différence est que le RRO dans un message Resv enregistre les informations de chemin dans la direction inverse.

Noter que chaque nœud le long du chemin va maintenant avoir le chemin complet de la source à la destination. Le RRO Path aura le chemin de la source à ce nœud ; le RRO Resv aura le chemin de ce nœud à la destination. Cela est utile pour la gestion du réseau.

Un message Path reçu sans RRO indique que le nœud envoyeur n'a plus besoin de l'enregistrement de chemin. Les messages Resv suivants NE DEVRONT PAS contenir de RRO.

4.4.4 Détection de boucle

Au titre du traitement d'un RRO entrant, un routeur intermédiaire regarde dans tous les sous-objets contenus au sein du RRO. Si le routeur détermine qu'il est déjà dans la liste, il existe une boucle de transmission.

Une session RSVP est sans boucle si les nœuds en aval reçoivent les messages Path ou si les nœuds en amont reçoivent les messages Resv sans que des boucles soient détectées dans le RRO contenu.

Il y a deux grandes classifications des boucles de transmission. La première classe est la boucle transitoire, qui survient au titre du fonctionnement normal lorsque l'acheminement de couche 3 essaye de converger sur un chemin de transmission cohérent pour toutes les destinations. La seconde classe de boucle de transmission est la boucle permanente, qui résulte normalement d'une mauvaise configuration du réseau.

L'action effectuée par un nœud à réception d'un RRO dépend du type de message dans lequel le RRO est reçu.

Pour les messages Path qui contiennent une boucle de transmission, le routeur construit et envoie un message PathErr "Problème d'acheminement", avec la valeur d'erreur "Boucle détectée", et supprime le message Path. Jusqu'à ce que la boucle soit éliminée, cette session ne convient pas pour transmettre des paquets de données. La façon d'éliminer la boucle sort du domaine d'application du présent document.

Pour les messages Resv qui contiennent une boucle de transmission, le routeur élimine simplement le message. Les messages Resv ne devraient pas être en boucle si les messages Path ne le sont pas.

4.4.5 Compatibilité future

De nouveaux sous-objets peuvent être définis pour le RRO. Lors du traitement d'un RRO, les sous-objets non reconnus DEVRAIENT être ignorés et transmis. Lors du traitement d'un RRO pour une détection de boucle, un nœud DEVRAIT soumettre à l'analyse tous les objets non reconnus. La détection de boucle se fait par la détection de sous-objets qui ont été insérés par le nœud lui-même lors d'un passage précédent de l'objet. Cela assure que les sous-objets nécessaires pour la détection de boucle sont toujours compris.

4.4.6 Non prise en charge du RRO

Le RRO est à utiliser seulement lorsque tous les routeurs le long du chemin prennent en charge RSVP et le RRO. Le RRO reçoit une valeur de classe de la forme 0bbbbbbb. Les routeurs RSVP qui ne prennent pas en charge l'objet vont donc répondre par une erreur "Classe d'objet inconnue".

4.5 Codes d'erreur pour ERO et RRO

Dans le traitement décrit ci-dessus, certaines erreurs doivent être rapportées comme "Problème d'acheminement" ou "Notifier". La valeur du code d'erreur "Problème d'acheminement" est 24 ; la valeur du code d'erreur "Notifier" est 25.

Le tableau suivant définit les valeurs d'erreur pour le code d'erreur Problème d'acheminement :

Valeur	Erreur
1	Mauvais objet EXPLICIT_ROUTE
2	Mauvais nœud strict
3	Mauvais nœud lâche
4	Mauvais sous-objet initial
5	Pas de chemin disponible vers la destination
6	Valeur d'étiquette inacceptable
7	Le RRO indique des boucles d'acheminement
8	MPLS en négociation, mais un routeur non RSVP se trouve sur le chemin
9	Échec d'allocation d'étiquette MPLS
10	L3PID non pris en charge

Pour le code d'erreur Notifier, les 16 bits du champ Valeur d'erreur sont : ss00 cccc cccc cccc

Les bits de poids fort sont définis sous le code d'erreur 1. (Voir la [RFC2205]).

Lorsque ss = 00, les sous codes suivants sont définis :

1 RRO trop grand pour la MTU
2 Notification de RRO
3 Tunnel réparé en local

4.6 Objets Session, Gabarit d'envoyeur et Spéc de filtre

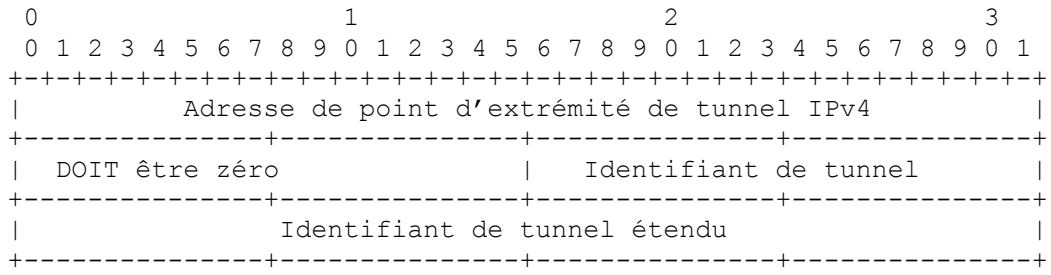
De nouveaux C-Types sont définis pour les objets SESSION, SENDER_TEMPLATE et FILTER_SPEC.

Les objets LSP_TUNNEL ont le format décrit dans les paragraphes suivants.

4.6.1 Objet Session

4.6.1.1 Objet Session LSP_TUNNEL_IPv4

Classe = SESSION, C-Type de LSP_TUNNEL_IPv4 = 7



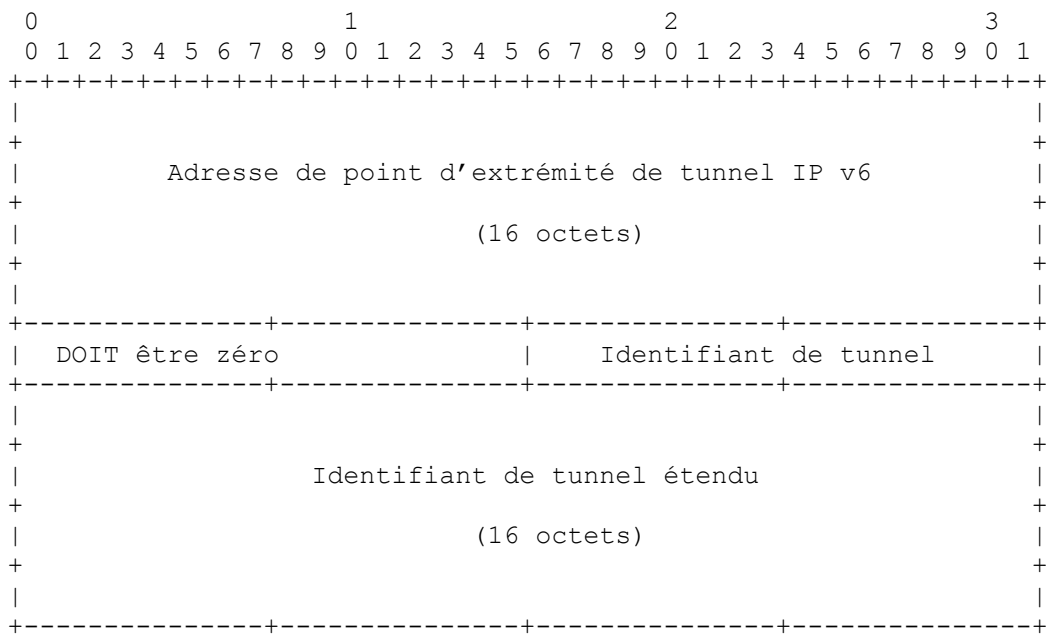
Adresse de point d'extrémité de tunnel IPv4
C'est l'adresse IPv4 du nœud de sortie pour le tunnel.

Identifiant de tunnel
C'est un identifiant de 16 bits utilisé dans la session qui reste constant sur la durée de vie du tunnel.

Identifiant de tunnel étendu
C'est un identifiant de 32 bits utilisé dans la session qui reste constant sur la durée de vie du tunnel. Réglé normalement tout à zéro. Les nœuds d'entrée qui souhaitent rétrécir la portée d'une session à la paire entrée-sortie peuvent placer leur adresse IPv4 ici comme identifiant mondialement univoque.

4.6.1.2 Objet Session LSP_TUNNEL_IPv6

Classe = SESSION, C_Type de LSP_TUNNEL_IPv6 = 8



Adresse de point d'extrémité de tunnel IPv6
C'est l'adresse IPv6 du nœud de sortie pour le tunnel.

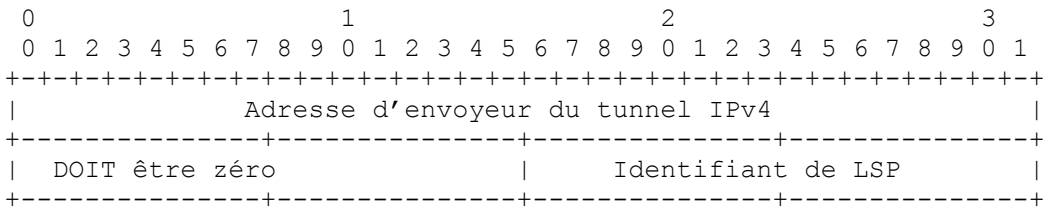
Identifiant de tunnel
Un identifiant de 16 bits utilisé dans la session qui reste constant sur la durée de vie du tunnel.

Identifiant de tunnel étendu
C'est un identifiant de 16 octets utilisé dans la session qui reste constant sur la durée de vie du tunnel. Réglé normalement tout à zéro. Les nœuds d'entrée qui souhaitent rétrécir la portée d'une session à la paire entrée-sortie peuvent placer leur adresse IPv6 ici comme identifiant mondialement univoque.

4.6.2 Objet Gabarit d'envoyeur

4.6.2.1 Objet Gabarit d'envoyeur LSP_TUNNEL_IPv4

Classe = SENDER_TEMPLATE, C-Type de LSP_TUNNEL_IPv4 = 7

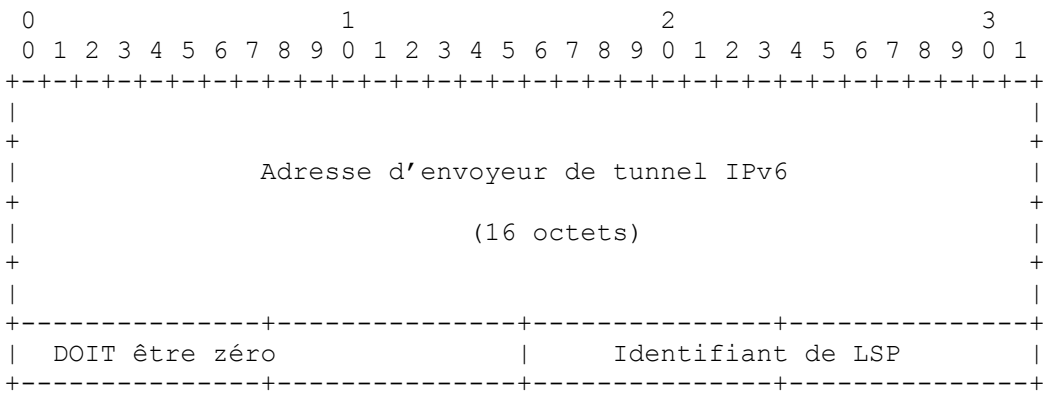


Adresse d'envoyeur de tunnel IPv4
C'est l'adresse IPv4 du nœud envoyeur.

Identifiant de LSP
C'est un identifiant de 16 bits utilisé dans le SENDER_TEMPLATE et la FILTER_SPEC qui peut être changé pour permettre à un envoyeur de partager des ressources avec lui-même.

4.6.2.2 Objet Gabarit d'envoyeur LSP_TUNNEL_IPv6

Classe = SENDER_TEMPLATE, C_Type de LSP_TUNNEL_IPv6 = 8



Adresse d'envoyeur de tunnel IPv6
C'est l'adresse IPv6 du nœud envoyeur

Identifiant de LSP
C'est un identifiant de 16 bits utilisé dans le SENDER_TEMPLATE et la FILTER_SPEC qui peuvent être changés pour permettre à un envoyeur de partager des ressources avec lui-même.

4.6.3 Objet Spécification de filtre

4.6.3.1 Objet Spécification de filtre LSP_TUNNEL_IPv4

Classe = FILTER_SPECIFICATION, C-Type de LSP_TUNNEL_IPv4 = 7

Le format de l'objet FILTER_SPEC LSP_TUNNEL_IPv4 est identique à celui de l'objet SENDER_TEMPLATE LSP_TUNNEL_IPv4.

4.6.3.2 Objet Spécification de filtre LSP_TUNNEL_IPv6

Classe = FILTER_SPECIFICATION, C_Type de LSP_TUNNEL_IPv6 = 8

Le format de l'objet FILTER_SPEC LSP_TUNNEL_IPv6 est identique à celui de l'objet SENDER_TEMPLATE LSP_TUNNEL_IPv6.

4.6.4 Réacheminement et procédure d'augmentation de la bande passante

Ce paragraphe décrit comment établir un tunnel qui soit capable de conserver les réservations de ressources (sans double comptage) lorsque il est réacheminé ou lorsque il tente d'augmenter sa bande passante. Dans le message Path initial, le nœud d'entrée forme un objet SESSION, alloue un identifiant de tunnel, et place son adresse IPv4 dans le Extended_Tunnel_ID. Il forme aussi un SENDER_TEMPLATE et alloue un LSP_ID. L'établissement de tunnel se poursuit alors suivant la procédure normale.

À réception du message Path, le nœud de sortie envoie un message Resv avec le STYLE Explicite partagé au nœud d'entrée.

Lorsque un nœud d'entrée avec un chemin établi veut changer ce chemin, il forme un nouveau message Path comme suit. L'objet SESSION existant est utilisé. En particulier, l'identifiant de tunnel et l'identifiant de tunnel étendu restent inchangés. Le nœud d'entrée choisit un nouvel identifiant de LSP pour former un nouveau Gabarit d'envoyeur. Il crée un objet EXPLICIT_ROUTE pour le nouveau chemin. Le nouveau message Path est envoyé. Le nœud d'entrée rafraîchit les deux messages Path, l'ancien et le nouveau.

Le nœud de sortie répond par un message Resv avec un descripteur de flux SE formaté comme suit :

<FLOWSPEC><vieille_FILTER_SPEC><vieux_LABEL_OBJECT><nouvelle_FILTER_SPEC>
<nouveau_LABEL_OBJECT>

(Noter que si les PHOP sont différents, deux messages sont alors envoyés, chacun avec la FILTER_SPEC et le LABEL_OBJECT appropriés.)

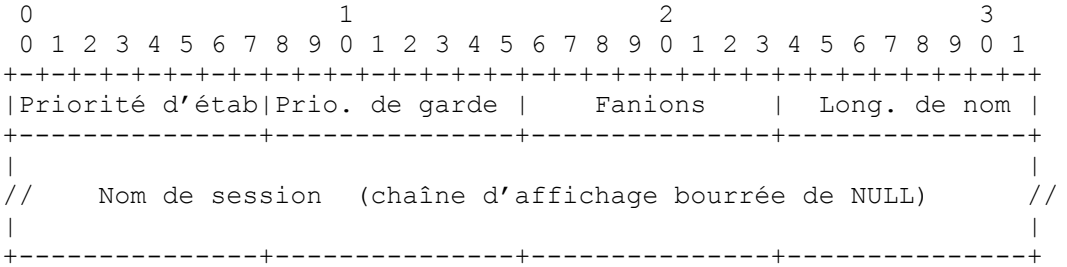
Lorsque le nœud d'entrée reçoit le ou les messages Resv, il peut commencer à utiliser le nouveau chemin. Il DEVRAIT envoyer un message PathTear pour le vieux chemin.

4.7 Objet Attribut de session

La classe de Attribut de session est 207. Deux C_Types sont définis, LSP_TUNNEL, C-Type = 7 et LSP_TUNNEL_RA, C-Type = 1. Le C-Type de LSP_TUNNEL_RA inclut tous les mêmes champs que le C-Type de LSP_TUNNEL. De plus, il porte les informations d'affinité de ressources. Les formats sont les suivants :

4.7.1 Format sans affinités de ressource

Classe de SESSION_ATTRIBUTE = 207, C-Type de LSP_TUNNEL = 7



Priorité d'étab : C'est la priorité de la session par rapport à la prise de ressources, dans la gamme de 0 à 7. La valeur 0 est la plus forte priorité. La priorité d'établissement est utilisée pour décider si cette session peut préempter une autre session.

Priorité de garde : C'est la priorité de la session par rapport à la garde des ressources, dans la gamme de 0 à 7. La valeur 0 est la plus forte priorité. La priorité de garde est utilisée pour décider si cette session peut être préemptée par une autre session.

Fanions

0x01 : La protection locale est désirée. Ce fanion permet aux routeurs de transit d'utiliser un mécanisme de réparation local qui peut résulter en une violation de l'objet Chemin explicite. Lorsque une faute est détectée sur une liaison adjacente vers l'aval, un routeur de transit peut réacheminer le trafic pour une restauration rapide du service .

0x02 : L'enregistrement d'étiquette est désiré. Ce fanion indique que les informations d'étiquette devraient être incluses lorsque on fait un enregistrement de chemin.

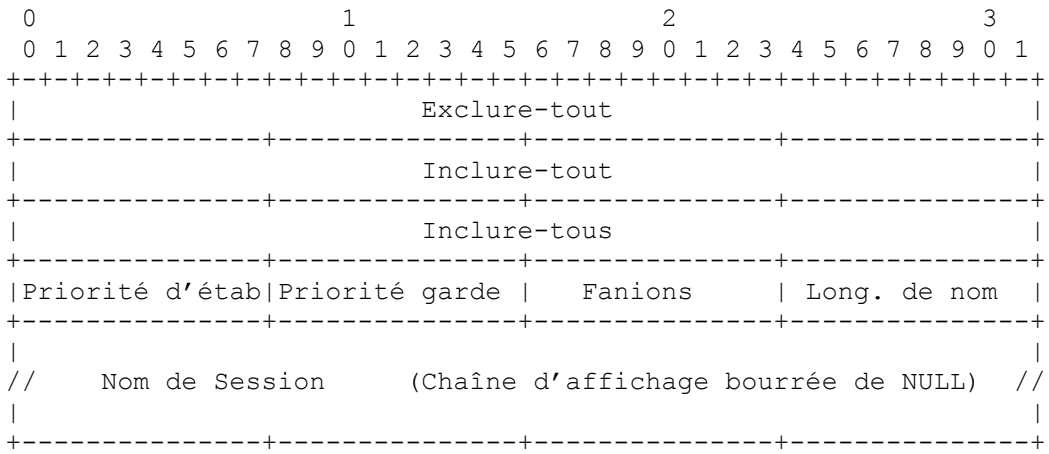
0x04 : Le style SE est désiré. Ce fanion indique que le nœud d'entrée du tunnel peut choisir de réacheminer ce tunnel sans le supprimer. Un nœud d'entrée de tunnel DEVRAIT utiliser le style SE lorsque il répond par un message Resv.

Longueur de nom : C'est la longueur de la chaîne affichée avant bourrage, en octets.

Nom de session : C'est une chaîne de caractères bourrée avec des zéros.

4.7.2 Format avec affinités de ressource

Classe de SESSION_ATTRIBUTE = 207, C-Type de LSP_TUNNEL_RA = 1



Exclure-tout : Vecteur de 32 bits qui représente un ensemble de filtres d'attributs associés à un tunnel dont tous rendent une liaison inacceptable.

Inclure-tout : Vecteur de 32 bits représentant un ensemble de filtres d'attribut associés à un tunnel dont tous rendent une liaison acceptable (par rapport à ce test). Un ensemble nul (tous les bits sont à zéro) réussit automatiquement.

Inclure-tous : Vecteur de 32 bits qui représente un ensemble de filtres d'attribut associés à un tunnel dont tous doivent être présents pour qu'une liaison soit acceptable (par rapport à ce test). Un ensemble nul (tous les bits sont à zéro) réussit automatiquement.

Priorité d'établissement : C'est la priorité de la session à l'égard de la prise de ressources, dans la gamme de 0 à 7. La valeur 0 est la plus forte priorité. La priorité d'établissement est utilisée pour décider si cette session peut préempter une autre session.

Priorité de garde : C'est la priorité de la session par rapport à la garde des ressources, dans la gamme de 0 à 7. La valeur 0 est la plus forte priorité. Priorité de garde est utilisé pour décider si cette session peut être préemptée par une autre session.

Fanions

0x01 : La protection locale est désirée. Ce fanion permet aux routeurs de transit d'utiliser un mécanisme local de réparation qui peut résulter en une violation de l'objet EXPLICIT_ROUTE. Lorsque une faute est détectée sur une liaison ou nœud aval adjacent, un routeur de transit peut réacheminer le trafic pour une restauration rapide du service.

0x02 : L'enregistrement d'étiquette est désiré. Ce fanion indique que les informations d'étiquettes devraient être incluses lorsque on fait un enregistrement de chemin.

0x04 : Le style SE est désiré. Ce fanion indique que le nœud d'entrée du tunnel peut choisir de réacheminer ce tunnel sans le supprimer. Un nœud d'entrée de tunnel DEVRAIT utiliser le style SE lorsque il répond par un message Resv.

Longueur de nom : C'est la longueur de la chaîne d'affichage avant bourrage, en octets.

Nom de session : C'est une chaîne de caractères bourrée de zéros.

4.7.3 Procédures applicables aux deux C-Types

La prise en charge des priorités d'établissement et de garde est FACULTATIVE. Un nœud peut reconnaître ces informations mais être incapable d'effectuer les opérations requises. Le nœud DEVRAIT passer les informations inchangées vers l'aval.

Comme noté ci-dessus, la préemption est mise en œuvre par deux priorités : la priorité d'établissement pour prendre les ressources, la priorité de garde pour conserver une ressource. Précisément, la priorité de garde est la priorité avec laquelle des ressources assignées à cette session seront réservées. La priorité d'établissement NE DEVRAIT jamais être supérieure à la priorité de garde pour une même session.

Les priorités d'établissement et de garde sont directement analogues aux priorités de préemption et défense telles que définies dans la [RFC2751]. Bien que l'interaction de ces deux objets soit en fin de compte une question de politique, les interactions par défaut suivantes sont RECOMMANDÉES.

Lorsque les deux objets sont présents, l'élément de politique Priorité de préemption est utilisé. Une transposition entre les espaces de priorité est définie comme suit. Une priorité d'attribut de session S est transposée en une priorité de préemption P par la formule P = 2^(14-2S). La transposition inverse est donnée dans le tableau suivant.

Priorité de préemption	**Priorité d'attribut de session**
0 – 3	7
4 – 15	6
16 – 63	5
64 – 255	4
256 – 1023	3
1024 – 4095	2
4096 – 16383	1
16384 – 65535	0

Lorsque un nouveau message Path est considéré pour admission, la bande passante demandée est comparée avec la bande passante disponible à la priorité spécifiée dans la priorité d'établissement.

Si la bande passante demandée n'est pas disponible, un message PathErr est retourné avec un code d'erreur de 01, Échec de contrôle d'admission, et une valeur d'erreur de 0x0002. Le premier 0 dans la valeur d'erreur indique un sous-code à définition mondiale et ne donne pas d'information. Le 002 indique "Bande passante demandée indisponible".

Si la bande passante demandée est inférieure à la bande passante inutilisée, le traitement est alors achevé. Si la bande passante demandée est disponible, mais est utilisée par des sessions de priorité inférieure, les sessions de priorité inférieure (en commençant par la plus faible priorité) PEUVENT être préemptées pour libérer la bande passante nécessaire.

Lorsque la préemption est prise en charge, chaque réservation préemptée déclenche un appel TC_Preempt() aux clients locaux, en passant un sous-code qui indique la raison. Une ResvErr et/ou PathErr avec le code "Échec de contrôle de politique" DEVRAIT être envoyé vers les receveurs en aval et vers les envoyeurs en amont.

La prise en charge de la protection locale est FACULTATIVE. Un nœud peut reconnaître le fanion Protection locale mais être incapable d'effectuer l'opération demandée. Dans ce cas, le nœud DEVRAIT passer les informations inchangées vers l'aval.

L'enregistrement du sous-objet Étiquette dans l'objet ROUTE_RECORD est contrôlé par le fanion Enregistrement d'étiquette désiré dans l'objet SESSION_ATTRIBUTE. Comme le sous-objet Étiquette n'est pas nécessaire pour toutes les applications, il n'est pas automatiquement enregistré. Le fanion permet aux applications de demander cela seulement quand c'est nécessaire.

Le contenu du champ Nom de session est une chaîne, normalement de caractères affichables. La Longueur DOIT toujours être un multiple de 4 et DOIT être au moins 8. Pour un objet Longueur qui n'est pas multiple de 4, l'objet est bourré avec des caractères NUL en queue. Le champ Longueur de nom contient la longueur réelle de la chaîne.

4.7.4 Procédures d'affinité de ressource

Les classes de ressource et les affinités de classe de ressource sont décrites dans la [RFC2702]. Dans le présent document on utilise le terme plus bref d'affinités de ressources pour ce dernier terme. Les classes de ressources peuvent être associées à des liaisons et annoncées dans les protocoles d'acheminement. Les affinités de classes de ressources sont utilisées par RSVP de deux façons. Afin d'être validée, une liaison DOIT réussir les trois essais ci-dessous. Si l'essai échoue, une PathErr avec le code "Échec de contrôle de politique" DEVRAIT être envoyé.

Lorsque une nouvelle réservation est considérée pour l'admission sur un nœud strict dans un ERO, un nœud PEUT valider les affinités de ressources avec les classes de ressources de cette liaison. Lorsque un nœud choisit des liaisons afin d'étendre un nœud lâche d'un ERO, le nœud DOIT valider les classes de ressources de ces liaisons par rapport aux affinités de ressources. Si aucune liaison acceptable ne peut être trouvée pour étendre l'ERO, le nœud DEVRAIT envoyer un message PathErr avec un code d'erreur de "Problème d'acheminement" et une valeur d'erreur de "Pas de chemin disponible vers la destination".

Pour être validée, une liaison DOIT réussir les trois essais suivants :
Pour décrire précisément les essais on utilise les définitions des descriptions d'objet données plus haut. On définit aussi Link-attr : Vecteur de 32 bits qui représente les attributs associés à une liaison.

Les trois essais sont

1. Exclure-tout
Cet essai exclut une liaison si elle porte un des attributs de l'ensemble (link-attr & exclure-tout) == 0

2. Inclure-tout
Cet essai accepte une liaison si elle porte un des attributs de l'ensemble (inclure-tout == 0) | ((link-attr & inclure-tout) != 0)

3. Inclure-tous
Cet essai n'accepte une liaison que si elle porte tous les attributs dans l'ensemble (inclure-tous == 0) | (((link-attr & inclure-tous) ^ inclure- tous) == 0)

Pour qu'une liaison soit acceptable, les trois essais DOIVENT réussir. Si l'essai échoue, le nœud DEVRAIT envoyer un message PathErr avec un code d'erreur de "Problème d'acheminement" et une valeur d'erreur de "Pas de chemin disponible vers la destination".

Si un message Path contient plusieurs objets SESSION_ATTRIBUTE, seul le premier objet SESSION_ATTRIBUTE est significatif. Les objets SESSION_ATTRIBUTE suivants peuvent être ignorés et n'ont pas besoin d'être transmis.

Tous les routeurs RSVP, qu'ils prennent ou non en charge l'objet SESSION_ATTRIBUTE, DEVRONT transmettre l'objet non modifié. La présence de routeurs non RSVP, n'importe où entre les envoyeurs et les receveurs, n'a pas d'impact sur cet objet.

5. Extension de Hello

L'extension du Hello RSVP permet aux nœuds RSVP de détecter quand un nœud du voisinage n'est pas accessible. Le mécanisme fournit la détection des défaillances nœud par nœud. Lorsque une telle défaillance est détectée, elle est traitée à peu près de la même façon qu'une défaillance de communication de couche liaison. Ce mécanisme est destiné à être utilisé lorsque la notification des défaillances de couche liaison n'est pas disponible et qu'on n'utilise pas de liaison non

numérotée, ou lorsque les mécanismes de détection de défaillance fournis par la couche liaison ne sont pas suffisants pour détecter à temps la défaillance du nœud.

On notera que la détection de la défaillance du nœud n'est pas la même que le mécanisme de détection de la défaillance de la liaison, particulièrement dans le cas de multiples liaisons non numérotées parallèles.

L'extension de Hello est spécifiquement conçue pour qu'un côté puisse utiliser le mécanisme alors que l'autre ne l'utilise pas. La détection de la défaillance du voisin peut être initiée à tout moment. Cela inclut quand les voisins se découvrent l'un l'autre, ou juste quand les voisins partagent l'état Resv ou Path.

L'extension de Hello est composée d'un message Hello, d'un objet HELLO_REQUEST et d'un objet HELLO_ACK. Le traitement du Hello entre deux voisins prend en charge le choix indépendant d'intervalles de détection de défaillance, normalement configurés. Chaque voisin peut de façon autonome produire des objets HELLO_REQUEST. Chaque demande est acquittée par un accusé de réception. Les messages Hello contiennent aussi assez d'informations pour qu'un voisin puisse supprimer la production de demandes Hello tout en continuant d'effectuer la détection de défaillance de voisin. Un message Hello peut être inclus comme un sous-message au sein d'un faisceau de messages.

La détection de défaillance de voisin est réalisée par la collecte et la mémorisation d'une valeur "d'instance" du voisin. Si un changement de cette valeur est vu ou si le voisin ne fait pas correctement rapport de la valeur annoncée en local, le voisin est alors présumé avoir réamorcé. Lorsqu'on voit un changement de la valeur d'un voisin ou lorsque la communication avec un voisin est perdue, la valeur d'instance annoncée à ce voisin est aussi changée. Les objets HELLO fournissent un mécanisme pour interroger sur, et fournir, une valeur d'instance. Une demande d'interrogation inclut aussi la valeur d'instance de l'envoyeur. Cela permet au receveur d'une interrogation de traiter facultativement l'interrogation comme une réponse implicite d'interrogation. Ce traitement facultatif est une optimisation qui peut réduire le nombre total d'interrogations et réponses traitées par une paire de voisins. Dans tous les cas, lorsque les deux côtés prennent en charge l'optimisation, le résultat sera un seul ensemble d'interrogations et réponses par intervalle de détection de défaillance. Selon les intervalles choisis, le même bénéfice peut être retiré même lorsque un seul voisin prend en charge l'optimisation.

5.1 Format du message Hello

Les messages Hello sont toujours envoyés entre deux voisins RSVP. L'adresse de source IP est l'adresse IP du nœud envoyeur. L'adresse IP de destination est l'adresse IP du nœud voisin.

Le mécanisme Hello est destiné à être utilisé entre des voisins immédiats. Lorsque les messages Hello sont échangés entre des voisins immédiats, le champ IP TTL pour tous les messages Hello sortants DEVRAIT être réglé à 1.

Le message Hello a un type de message de 20. Le format du message Hello est le suivant :

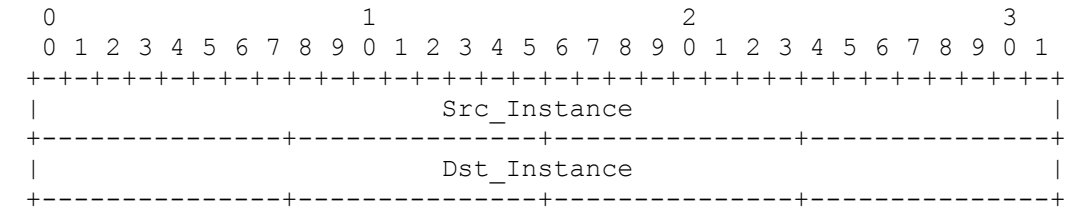
<message Hello> ::= <En-tête commun> [<INTÉGRITÉ>] <HELLO>

5.2 Formats d'objet HELLO

La classe de HELLO est 22. Deux C_Type sont définis.

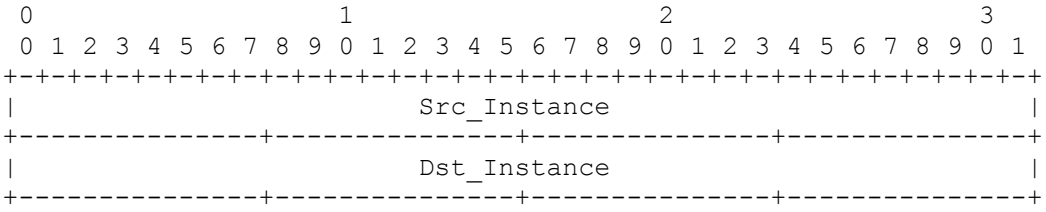
5.2.1 Objet HELLO_REQUEST

Classe = HELLO, C_Type = 1



5.2.2 Objet HELLO_ACK

Classe = HELLO, C_Type = 2



Src_Instance : 32 bits
Valeur de 32 bits qui représente l'instance de l'envoyeur. L'annonceur conserve une représentation/valeur par voisin. Cette valeur DOIT changer lorsque l'envoyeur fait un réamorçage, lorsque le nœud se réinitialise, ou quand la communication est perdue avec le nœud voisin, et autrement, elle reste la même. Ce champ NE DOIT PAS être réglé à zéro (0).

Dst_Instance : 32 bits
C'est la plus récente valeur de Src_Instance reçue du voisin. Ce champ DOIT être réglé à zéro (0) lorsque aucune valeur n'a jamais été vue de la part du voisin.

5.3 Usage du message Hello

Le message Hello est entièrement FACULTATIF. Tous les messages peuvent être ignorés par les nœuds qui ne souhaitent pas participer au traitement de message Hello. La rédaction de ce paragraphe suppose que le receveur tout comme l'envoyeur participent. En particulier, l'utilisation de DOIT et DEVRAIT par rapport au receveur ne s'applique qu'à un nœud qui prend en charge le traitement du message Hello.

Un nœud génère périodiquement un message Hello qui contient un objet HELLO_REQUEST pour chaque voisin dont l'état est suivi. La périodicité est réglée par le hello_interval. Cette valeur PEUT être configurée voisin par voisin. La valeur par défaut est 5 ms.

Lorsque on génère un message qui contient un objet HELLO_REQUEST, l'envoyeur remplit le champ Src_Instance avec une valeur qui représente son instance par voisin. Cette valeur NE DOIT PAS changer tandis que l'agent échange des Hello avec les voisins correspondants. L'envoyeur remplit aussi le champ Dst_Instance avec la valeur de Src_Instance la plus récemment reçue du voisin. Pour référence, on appellera cette variable Neighbor_Src_Instance. Si aucune valeur n'a jamais été reçue du voisin ou si ce nœud considère que la communication avec ce voisin a été perdue, le Neighbor_Src_Instance est réglé à zéro (0). La génération d'un message DEVRAIT être supprimée lorsque un objet HELLO_REQUEST a été reçu du nœud de destination dans l'intervalle hello_interval.

À réception d'un message contenant un objet HELLO_REQUEST, le receveur DOIT générer un message Hello contenant un objet HELLO_ACK. Le receveur DEVRAIT aussi vérifier que le voisin n'a pas réamorcé. Cela se fait en comparant la valeur du champ Src_Instance de l'envoyeur avec la valeur reçue précédemment. Si la valeur Neighbor_Src_Instance est zéro, et si le champ Src_Instance est différent de zéro, la Neighbor_Src_Instance est mise à jour avec la nouvelle valeur. Si les valeurs diffèrent ou si le champ Src_Instance est de zéro, alors le nœud DOIT traiter le voisin comme si la communication avait été perdue.

Le receveur d'un objet HELLO_REQUEST DEVRAIT aussi vérifier que le voisin reflète bien la valeur d'instance du receveur. Cela se fait en comparant le champ Dst_Instance reçu avec la valeur du champ Src_Instance le plus récemment transmis à ce voisin. Si le voisin continue à annoncer une mauvaise valeur différente de zéro après un nombre configuré d'intervalles, le nœud DOIT alors traiter le voisin comme si la communication avait été perdue.

À réception d'un message contenant un objet HELLO_ACK, le receveur DOIT vérifier que le voisin n'a pas réamorcé. Cela se fait en comparant la valeur du champ Src_Instance de l'envoyeur avec la valeur reçue précédemment. Si la valeur de Neighbor_Src_Instance est zéro, et si le champ Src_Instance est différent de zéro, le Neighbor_Src_Instance est mis à jour avec la nouvelle valeur. Si les valeurs diffèrent ou si le champ Src_Instance est de zéro, le nœud DOIT alors traiter le voisin comme si la communication avait été perdue.

Le receveur d'un objet HELLO_ACK DOIT aussi vérifier que le voisin reflète la valeur d'instance du receveur. Si le voisin annonce une mauvaise valeur dans le champ Dst_Instance, un nœud DOIT alors traiter le voisin comme si la communication avait été perdue.

Si aucune valeur d'instance n'est reçue, via des objets REQUEST ou ACK, d'un voisin pendant un nombre configuré de hello_interval, un nœud DOIT alors supposer qu'il ne peut pas communiquer avec le voisin. Par défaut, ce nombre est 3,5.

Lorsque la communication est perdue ou supposée perdue comme décrit ci-dessus, un nœud PEUT réinitialiser les HELLO. Si un nœud ne réinitialise pas, il DOIT utiliser une valeur de Src_Instance différente de celle annoncée dans les messages Hello précédents. Cette nouvelle valeur DOIT continuer d'être annoncée au voisin correspondant jusqu'à ce que survienne une réinitialisation ou un réamorçage, ou jusqu'à ce qu'une autre défaillance de communication soit détectée. Si une nouvelle valeur d'instance n'a pas été reçue du voisin, le nœud DOIT alors annoncer zéro dans le champ de valeur de Dst_instance.

5.4 Considérations de multi-liaisons

Comme on l'a noté précédemment, l'extension de Hello est ciblée sur la détection de la défaillance du nœud et non pas sur la défaillance de la liaison. Lorsque il y a seulement une liaison entre les nœuds voisins ou lorsque toutes les liaisons entre une paire de nœuds échouent, la distinction entre défaillance de nœud et défaillance de liaison n'a pas de réelle signification et le traitement de telles défaillances a déjà été couvert. Lorsque il y a plusieurs liaisons qui sont partagées entre les voisins, des aspects particuliers sont à considérer. Lorsque les liaisons entre les voisins sont numérotées, les Hello DOIVENT alors être gérés sur chaque liaison et le mécanisme décrit précédemment s'applique.

Lorsque les liaisons ne sont pas numérotées, la détection de la défaillance de la liaison DOIT être fournie par d'autres moyens que les Hello. Chaque nœud DEVRAIT utiliser un seul échange de Hello avec le voisin. Le cas où toutes les liaisons sont défaillantes est le même que celui de la valeur non reçue mentionné au paragraphe précédent.

5.5 Compatibilité

L'extension de Hello n'affecte pas le traitement des autres messages RSVP. Le seul effet est de permettre qu'un événement de mort d'une liaison (nœud) soit déclaré plus tôt. La réponse RSVP à cette condition est inchangée.

L'extension de Hello est pleinement rétro-compatible. Il est alloué à la classe Hello une valeur de classe de la forme 0bbbbbbb. Selon la mise en œuvre, celles qui ne prennent pas en charge l'extension vont éliminer en silence les messages Hello ou répondre par une erreur "Classe d'objet inconnue". Dans l'un et l'autre cas, l'envoyeur n'aura pas d'accusé de réception pour le Hello produit.

6. Considérations pour la sécurité

En principe, ces extensions à RSVP ne posent aucun problème de sécurité en plus de ceux de la [RFC2205]. Cependant, il y a une légère modification dans le modèle de confiance. Le trafic envoyé sur une session RSVP normale peut être filtré selon les adresses de source et de destination aussi bien que par numéro d'accès. Dans la présente spécification, le filtrage ne survient que sur la base de l'étiquette entrante. Pour cette raison, une administration peut souhaiter limiter le domaine sur lequel les LSP tunnels peuvent être établis. Cela peut être réalisé en réglant des filtres sur divers accès pour refuser une action sur un message Path RSVP avec un objet SESSION de type LSP_TUNNEL_IPv4 [RFC1349] ou LSP_TUNNEL_IPv6 [RFC2474].

7. Considérations relatives à l'IANA

L'IANA alloue des valeurs aux paramètres du protocole RSVP. Dans le présent document sont définis les objets EXPLICIT_ROUTE et ROUTE_RECORD. Chacun de ces objets contient des sous-objets. La présente section définit les règles pour l'allocation des numéros des sous-objets. Cette section utilise la terminologie du BCP 26 "Lignes directrices pour la rédaction d'une section Considérations relatives à l'IANA dans les RFC" [RFC2434].

Type de sous-objet EXPLICIT_ROUTE
Le type de sous-objet EXPLICIT_ROUTE est un nombre de 7 bits qui identifie la fonction du sous-objet. Il n'y a pas de restriction de gamme. Toutes les valeurs possibles sont disponibles pour l'allocation.

Suivant les politiques exposées dans la [RFC2434], les types de sous-objet dans la gamme de 0 à 63 (0x00 - 0x3F) sont alloués par action de consensus de l'IETF, les codes dans la gamme 64 à 95 (0x40 - 0x5F) sont alloués au premier qui les demande, et les codes dans la gamme 96 à 127 (0x60 - 0x7F) sont réservés pour utilisation privée.

Type de sous-objet ROUTE_RECORD
Le type de sous-objet ROUTE_RECORD est un nombre de 8 bits qui identifie la fonction du sous-objet. Il n'y a pas de restriction de gamme. Toutes les valeurs possibles sont disponibles pour l'allocation.

Suivant les politiques exposées dans la [RFC2434], les types de sous-objet dans la gamme de 0 - 127 (0x00 - 0x7F) sont alloués par action de consensus de l'IETF, les codes dans la gamme 128 - 191 (0x80 - 0xBF) sont alloués au premier qui les demande, et les codes dans la gamme 192 - 255 (0xC0 - 0xFF) sont réservés pour utilisation privée.
Les allocations suivantes sont faites dans le présent document.

7.1 Types de message

Numéro de message : 20
Nom du message : Hello

7.2 Numéros de classe et C-Types

Numéro de classe : 1
Nom de classe : SESSION

Types de classe ou C-Types :
7 LSP Tunnel IPv4
8 LSP Tunnel IPv6
10 FILTER_SPEC
Types de classe ou C-Types :
7 LSP Tunnel IPv4
8 LSP Tunnel IPv6

11 SENDER_TEMPLATE
Types de classe ou C-Types :
7 LSP Tunnel IPv4
8 LSP Tunnel IPv6

16 RSVP_LABEL
Types de classe ou C-Types :
1 Type 1 Label

19 LABEL_REQUEST
Types de classe ou C-Types :
1 Sans gamme d'étiquettes
2 Avec gamme d'étiquettes ATM
3 Avec gamme d'étiquettes de relais de trame

20 EXPLICIT_ROUTE
Types de classe ou C-Types :
1 Type 1 Chemin explicite

21 ROUTE_RECORD
Types de classe ou C-Types :
1 Type 1 Enregistrement de chemin

22 HELLO
Types de classe ou C-Types :
1 Demande
2 Accusé de réception

207 SESSION_ATTRIBUTE
Types de classe ou C-Types :
1 LSP_TUNNEL_RA
7 LSP Tunnel

7.3 Codes d'erreur et sous codes de valeur d'erreur à définition mondiale

La liste ci-après étend la liste de base des codes d'erreurs et des valeurs qui sont définis dans la [RFC2205].

Code d'erreur	Signification
24	Problème d'acheminement
	Ce code d'erreur a les sous codes de valeur d'erreur à définition mondiale :
	1 Mauvais objet EXPLICIT_ROUTE
	2 Mauvais nœud strict
	3 Mauvais nœud lâche
	4 Mauvais sous-objet initial
	5 Pas de chemin disponible pour cette destination
	6 Valeur d'étiquette inacceptable
	7 RRO indique une boucle d'acheminement
	8 MPLS en cours de négociation, mais un routeur sans capacité RSVP est sur le chemin
	9 Échec d'allocation d'étiquette MPLS
	10 L3PID non accepté
25	Erreur Notifier
	Ce code d'erreur a les sous codes de valeur d'erreur à définition mondiale :
	1 RRO trop grand pour la MTU
	2 Notification RRO
	3 Tunnel réparé localement

7.4 Définitions de sous-objet

Sous-objets de l'objet EXPLICIT_ROUTE avec le C-Type 1 :

1 préfixe IPv4
2 préfixe IPv6
32 numéro de système autonome

Sous-objets de l'objet RECORD_ROUTE avec le C-Type 1 :

1 adresse IPv4
2 adresse IPv6
3 étiquette

8. Considérations sur la propriété intellectuelle

Des droits de propriété intellectuelle ont été notifiés à l'IETF à l'égard de tout ou partie de la spécification contenue dans le présent document. Pour plus d'informations, consulter la liste en ligne des revendications de droits.

9. Remerciements

Le présent document contient des idées ainsi que du texte qui sont apparus dans de précédents projets Internet. Les auteurs du document actuel tiennent à remercier les auteurs de ces projets. Ce sont Steven Blake, Bruce Davie, Roch Guerin, Sanjay Kamat, Yakov Rekhter, Eric Rosen, et Arun Viswanathan. Nous souhaitons aussi adresser nos remerciements à Bora Akyol, Yoram Bernet et Alex Mondrus pour leurs commentaires sur le présent document.

10. Références

[RFC0792] J. Postel, "Protocole du message de contrôle Internet – Spécification du protocole du programme Internet DARPA", STD 5, septembre 1981.

[RFC1191] J. Mogul et S. Deering, "Découverte de la MTU de chemin", novembre 1990.

[RFC1349] P. Almquist, "Type de service dans la suite de protocole Internet", juillet 1992. *(Obsolète, voir RFC 2474)*

[RFC2119] S. Bradner, "Mots clés à utiliser dans les RFC pour indiquer les niveaux d'exigence", BCP 14, mars 1997.

[RFC2205] R. Braden, éd., L. Zhang, S. Berson, S. Herzog, S. Jamin, "Protocole de réservation de ressource (RSVP) -- version 1, spécification fonctionnelle", septembre 1997. (*MàJ par* RFC2750, RFC3936, RFC4495) *(P.S.)*

[RFC2210] J. Wroclawski, "Utilisation de RSVP avec les services intégrés de l'IETF", septembre 1997. *(P.S.)*

[RFC2211] J. Wroclawski, "Spécification du service d'élément de réseau à charge contrôlée", septembre 1997. *(P.S.)*

[RFC2434] T. Narten et H. Alvestrand, "Lignes directrices pour la rédaction d'une section Considérations relatives à l'IANA dans les RFC", BCP 26, octobre, 1998. *(Rendue obsolète par la RFC 5226)*

[RFC2463] A. Conta, S. Deering, "Protocole de message de contrôle Internet (ICMPv6) pour le protocole Internet version 6 (IPv6)", décembre 1998. (*Obsolète, voir* RFC4443) *(D.S.)*

[RFC2474] K. Nichols, S. Blake, F. Baker et D. Black, "Définition du champ Services différenciés (DS) dans les en-têtes IPv4 et IPv6", décembre 1998. (*MàJ par* RFC3168, RFC3260) *(P.S.)*

[RFC2702] D. Awduche et autres, "Exigences d'ingénierie du trafic sur MPLS", septembre 1999. *(Information)*

[RFC2751] S. Herzog, "Élément de politique de priorité par préemption signalée", janvier 2000. (*Obsolète, voir* RFC3181) *(P.S.)*

[RFC2997] Y. Bernet, A. Smith, B. Davie, "Spécification du type de service Null", novembre 2000. *(P.S.)*

[RFC3031] E. Rosen, A. Viswanathan, R. Callon, "Architecture de commutation d'étiquettes multi protocoles", janvier 2001. *(P.S.)*

[RFC3032] E. Rosen et autres, "Codage de pile d'étiquettes MPLS", janvier 2001.

[RFC3210] D. Awduche, A. Hannan, X. Xiao, "Déclaration d'applicabilité pour les extensions à RSVP pour LSP tunnels", décembre 2001. *(Information)*

11. Adresse des auteurs

Daniel O. Awduche
Movaz Networks, Inc.
7926 Jones Branch Drive, Suite 615
McLean, VA 22102
tél : +1 703-298-5291
mél ; awduche@movaz.com

Lou Berger
Movaz Networks, Inc.
7926 Jones Branch Drive, Suite 615
McLean, VA 22102
tél : +1 703 847 1801
mél : lberger@movaz.com

Der-Hwa Gan
Juniper Networks, Inc.
385 Ravendale Drive
Mountain View, CA 94043
mél : dhg@juniper.net

Tony Li
Procket Networks
3910 Freedom Circle, Ste. 102A
Santa Clara CA 95054
mél : tli@procket.com

Vijay Srinivasan
Cosine Communications, Inc.
1200 Bridge Parkway
Redwood City, CA 94065
tél : +1 650 628 4892
mél : vsriniva@cosinecom.com

George Swallow
Cisco Systems, Inc.
250 Apollo Drive
Chelmsford, MA 01824
tél : +1 978 244 8143
mél : swallow@cisco.com

12. Déclaration de droits de reproduction

Remerciement
Le financement de la fonction d'édition des RFC est actuellement fourni par la Internet Society.