Groupe de travail Réseau
Request for Comments : 4204
Catégorie : Sur la voie de la normalisation

J. Lang, éditeur, Sonos, Inc.
octobre 2005
Traduction Claude Brière de L'Isle

Protocole de gestion de liaison (LMP)

Statut de ce mémoire

Le présent document spécifie un protocole de l'Internet en cours de normalisation pour la communauté de l'Internet, et appelle à des discussions et suggestions pour son amélioration. Prière de se référer à l'édition en cours des "Normes officielles des protocoles de l'Internet" (STD 1) pour connaître l'état de la normalisation et le statut de ce protocole. La distribution du présent mémoire n'est soumise à aucune restriction.

Résumé

Pour une meilleure adaptabilité, plusieurs liaisons de données peuvent être combinées pour former une seule liaison d'ingénierie du trafic (TE, *traffic engineering*). De plus, la gestion des liaisons TE ne se restreint pas à la messagerie dans la bande, mais peut plutôt être faite en utilisant des techniques hors bande. Le présent document spécifie un protocole de gestion de liaisons (LMP, *Link Management Protocol*) qui fonctionne entre une paire de nœuds et est utilisé pour gérer les liaisons TE. Spécifiquement, LMP sera utilisé pour maintenir la connexité du canal de contrôle, vérifier la connexité physique des liaisons de données, corréler les informations de propriétés des liaisons, supprimer les alarmes vers l'aval, et localiser les défaillances des liaisons pour les besoins de protection/restauration dans diverses sortes de réseaux.

Table des Matières

1. Introduction

Les réseaux sont développés avec des routeurs, des commutateurs, des brasseurs, des systèmes à multiplexage par répartition en longueur d'onde à haute densité (DWDM, *dense wavelength division multiplexing*) et des multiplexeurs d'insertion/extraction (ADM, *add-drop multiplexor*) qui utilisent un plan de contrôle commun (par exemple, la commutation multi protocole avec étiquetage des flux généralisée (GMPLS, *Generalized MPLS*)) pour provisionner de façon dynamique des ressources et assurer la survie du réseau en utilisant des techniques de protection et de restauration. Une paire de nœuds peut avoir des milliers d'interconnexions, où chacune de ces interconnexions peut consister en plusieurs liaisons de données lorsque le multiplexage (par exemple, DLCI en relais de trame à la couche 2, des intervalles à multiplexage par répartition dans le temps (TDM, *time division multiplexed*) ou des longueurs d'ondes à multiplexage par répartition en longueur d'onde (WDM, *wavelength division multiplexed*) à la couche 1) est utilisé. Pour l'adaptabilité, plusieurs liaisons de données peuvent être combinées en une seule liaison d'ingénierie du trafic (TE, *traffic-engineering*).

Pour permettre la communication entre les nœuds pour l'acheminement, la signalisation, et la gestion de liaisons, il doit y avoir une paire d'interfaces IP qui soient mutuellement joignables. On appelle une telle paire d'interfaces un canal de contrôle. Noter que "mutuellement joignable" n'implique pas que ces deux interfaces soient (directement) connectées par une liaison IP ; il peut y avoir un réseau IP entre les deux. De plus, l'interface sur laquelle les messages de commandes sont envoyés/reçus peut ne pas être la même interface que celle sur laquelle s'écoulent les données. Le présent document spécifie un protocole de gestion de liaisons (LMP) qui fonctionne entre une paire de nœuds et est utilisé pour gérer les liaisons TE et vérifier l'accessibilité du canal de contrôle. Pour les besoins du présent document, de tels nœuds sont considérés comme des "voisins LMP" ou simplement des "nœuds du voisinage".

Dans GMPLS, les canaux de contrôle entre deux nœuds adjacents ne sont plus obligés d'utiliser le même support physique comme des liaisons de données entre ces nœuds. Par exemple, un canal de contrôle pourrait utiliser un circuit virtuel séparé, une longueur d'onde, une fibre, une liaison Ethernet, un tunnel IP acheminé sur un réseau de gestion séparé, ou un réseau IP à plusieurs bonds. Une conséquence de permettre que les canaux de contrôle entre deux nœuds soient logiquement ou physiquement différents des liaisons de données associées est que la santé d'un canal de contrôle n'est pas nécessairement corrélée à la santé des liaisons de données, et vice-versa. Donc, une nette séparation entre le sort du canal de contrôle et celui des liaisons de données doit être faite. De nouveaux mécanismes doivent être développés pour gérer les liaisons de données, à la fois en termes de provisionnement de liaisons et de gestion des fautes.

Parmi les tâches qu'accomplit LMP est la vérification que le groupage de liaisons en liaisons TE, ainsi que les propriétés de ces liaisons, sont les mêmes aux deux points d'extrémité des liaisons – c'est ce qu'on appelle la "corrélation de propriétés de liaisons". Aussi, LMP peut communiquer ces propriétés de liaison au module IGP, qui peut alors les annoncer aux autres

nœuds dans le réseau. LMP peut aussi dire au module de signalisation la transposition entre liaisons TE et canaux de contrôle. Donc, LMP effectue une précieuse fonction de "glu" dans le plan de contrôle.

Noter qu'alors que l'existence du réseau de contrôle (à un seul ou à plusieurs bonds) est nécessaire pour permettre la communication, elle n'est en aucune façon suffisante. Par exemple, si les deux interfaces sont séparées par un réseau IP, les fautes dans le réseau IP peuvent résulter en l'absence d'un chemin IP d'une interface à l'autre, et donc, une interruption de communication entre les deux interfaces. D'un autre côté, toutes les défaillances du réseau de contrôle n'affectent pas un certain canal de contrôle, et donc le besoin d'établir et gérer des canaux de contrôle.

Pour les besoins du présent document, une liaison de données peut être considérée par chaque nœud sur lequel elle se termine soit comme un "accès", soit comme une "liaison composante", selon la capacité de multiplexage du point d'extrémité sur cette liaison ; les liaisons composantes sont capables de multiplexage, tandis que les accès ne le sont pas. Cette distinction est importante car la gestion de telles liaisons (incluant, par exemple, l'allocation de ressource, l'allocation d'étiquettes, et leur vérification physique) est différente selon leur capacité de multiplexage. Par exemple, un commutateur de relais de trame est capable de démultiplexer une interface en circuits virtuels sur la base des DLCI ; de même, un brasseur SONET avec des interfaces OC-192 peut être capable de démultiplexer le flux OC-192 en quatre flux OC-48. Si plusieurs interfaces sont groupées en une seule liaison TE en utilisant la mise en faisceaux de liaisons [RFC4201], les ressources en liaisons doivent être identifiées en utilisant trois niveaux : identifiant de liaison, identifiant d'interface de composant, et étiquette identifiant le circuit virtuel, l'intervalle de temps, etc. L'allocation de ressource se produit au plus bas niveau (celui des étiquettes) mais la connexité physique se fait au niveau de la liaison composante. Pour un autre exemple, on considèrera le cas d'un commutateur optique (par exemple, PXC) qui commute de façon transparente des faisceaux lumineux OC-192. Si plusieurs interfaces sont à nouveau groupées dans une seule liaison TE, la mise en faisceau de liaison de la [RFC4201] n'est pas requise et seulement deux niveaux d'identification sont nécessaires : l'identifiant de liaison et l'identifiant d'accès. Dans ce cas, l'allocation de ressource et la connexité physique se passent toutes deux au niveau inférieur (c'est-à-dire au niveau accès).

Pour assurer l'inter fonctionnement entre des liaisons de données avec des capacités de multiplexage différentes, les appareils à capacité LMP DEVRAIENT permettre aux sous canaux d'une liaison composante d'être configurés en local comme des liaisons de données (logiques). Par exemple, si un routeur avec quatre interfaces OC-48 est connecté à travers un MUX 4:1 à un brasseur avec des interfaces OC-192, le brasseur devrait être capable de configurer chaque sous canal (par exemple, SPE STS-48c si le MUX 4:1 est un MUX SONET) comme une liaison de données.

LMP est conçu pour prendre en charge l'agrégation d'une ou plusieurs liaisons de données dans une liaison TE (soit des accès dans des liaisons TE, soit des liaisons composantes dans des liaisons TE). L'objet de la formation d'une liaison TE est de grouper/transposer les informations sur certaines ressources physiques (et leurs propriétés) dans les informations qui sont utilisées par la fonction de traitement de la signalisation (SPF, *Signalling Processing Function*) avec contrainte pour les besoins du calcul du chemin, et par la signalisation GMPLS.

1.1 Terminologie

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

Le lecteur est supposé être familiarisé avec la terminologie des [RFC3471], [RFC4201], et [RFC4202].

Liaison en faisceau *(Bundled Link)* : comme défini dans la [RFC4201], une liaison en faisceau est une liaison TE telle que, pour les besoins de la signalisation GMPLS, une combinaison de <identifiant de liaison, étiquette> n'est pas suffisante pour identifier sans ambiguïté les ressources appropriées utilisées par un LSP. Une liaison en faisceau se compose de deux liaisons composantes ou plus.

Canal de contrôle : un canal de contrôle est une paire d'interfaces mutuellement accessibles qui sont utilisées pour permettre la communication entre des nœuds pour l'acheminement, la signalisation, et la gestion des liaisons.

Liaison composante : comme défini dans la [RFC4201], une liaison composante est un sous ensemble de ressources d'une liaison TE telles que (a) la partition soit minimale, et (b) au sein de chaque sous ensemble une étiquette soit suffisante pour identifier sans ambiguïté les ressources appropriées utilisées par un LSP.

Liaison de données : une liaison de données est une paire d'interfaces qui sont utilisées pour transférer les données d'utilisateur. Noter que dans GMPLS, le ou les canaux de contrôle entre deux nœuds adjacents ne sont plus obligés d'utiliser le même support physique comme liaisons de données entre ces nœuds.

Corrélation de propriétés de liaison : c'est une procédure pour corréler les propriétés locales et distantes d'une liaison TE.

Capacité de multiplexage : capacité de multiplexer/démultiplexer un flux de données en flux de sous débit pour les besoins de la commutation.

Identifiant de nœud *(Node_Id)* : pour un nœud qui fonctionne avec OSPF, le Node_Id LMP est le même que l'adresse contenue dans la TLV Adresse de routeur OSPF. Pour un nœud qui fonctionne avec IS-IS et annonce la TLV Identifiant de routeur TE, le Node_Id est le même que l'identifiant de routeur annoncé.

Accès *(Port)* : interface qui termine une liaison de données.

Liaison TE : comme défini dans la [RFC4202], une liaison TE est une construction logique qui représente un moyen de grouper/transposer les informations sur certaines ressources physiques (et leurs propriétés) qui interconnecte des LSR pour les informations utilisées par la SPF contrainte pour les besoins du calcul du chemin, et par la signalisation GMPLS.

Transparent : un appareil est dit X-transparent si il transmet des signaux entrants d'entrée à sortie sans examiner ou modifier l'aspect X du signal. Par exemple, un commutateur de relais de trame est transparent à la couche réseau ; un commutateur tout optique est électriquement transparent.

2. Vue d'ensemble de LMP

Les deux procédures centrales de LMP sont la gestion du canal de contrôle et la corrélation des propriétés de liaison. La gestion du canal de contrôle est utilisée pour établir et maintenir les canaux de contrôle entre nœuds adjacents. Ceci est fait en utilisant un échange de messages Config et un mécanisme de garde en vie rapide entre les nœuds. Ce dernier est requis si des mécanismes de niveau inférieur ne sont pas disponibles pour détecter les défaillances du canal de contrôle. La corrélation des propriétés de liaison est utilisée pour synchroniser les propriétés de liaison TE et vérifier la configuration de la liaison TE.

LMP exige qu'une paire de nœuds ait au moins un canal de contrôle bidirectionnel actif entre eux. Chaque direction du canal de contrôle est identifiée par un identifiant de canal de contrôle (CC_Id), et les deux directions sont couplées en utilisant l'échange LMP de messages Config. Sauf pour les messages Test, qui peuvent être limités par le mécanisme de transport pour la messagerie dans la bande, tous les paquets LMP passent sur UDP avec un numéro d'accès LMP. Le codage de niveau liaison du canal de contrôle sort du domaine d'application de ce document.

Une "adjacence LMP" est formée entre deux nœuds lorsque au moins un canal de contrôle bidirectionnel est établi entre eux. Plusieurs canaux de contrôle peuvent être actifs simultanément pour chaque adjacence ; cependant, les paramètres du canal de contrôle DOIVENT être négociés individuellement pour chaque canal de contrôle. Si le mécanisme de garde en vie rapide de LMP est utilisé sur un canal de contrôle, les messages Hello LMP DOIVENT être échangés sur le canal de contrôle. Les autres messages LMP PEUVENT être transmis sur tous les canaux de contrôle actifs entre une paire de nœuds adjacents. Un ou plusieurs canaux de contrôle actifs peuvent être groupés en un canal de contrôle logique pour la signalisation, l'acheminement, et les besoins de corrélation des propriétés de liaison.

La fonction de corrélation de propriété de liaison de LMP est conçue pour agréger plusieurs liaisons de données (accès ou liaisons composantes) en une liaison TE et pour synchroniser les propriétés de la liaison TE. Au titre de la fonction de corrélation de propriétés de liaison, un échange de messages LinkSummary est défini. Le message LinkSummary comporte les identifiants de liaison locale et distante, une liste de toutes les liaisons de données qui composent la liaison TE, et diverses propriétés de liaison. Un message LinkSummaryAck ou LinkSummaryNack DOIT être envoyé en réponse à la réception d'un message LinkSummary indiquant l'accord ou le désaccord sur les propriétés de la liaison.

Les messages LMP sont transmis fiablement en utilisant les identifiants de message et les retransmissions. Les identifiants de message sont portés dans les objets MESSAGE_ID. Pas plus d'un objet MESSAGE_ID ne peut être inclus dans un message LMP. Pour les messages spécifiques du canal de contrôle, l'identifiant de message est dans la portée du canal de contrôle sur lequel le message est envoyé. Pour les messages spécifiques de liaison TE, l'identifiant de message est dans la portée de l'adjacence LMP. La valeur de l'identifiant de message est à croissance monotone et revient à zéro lorsque la valeur maximum est atteinte.

Dans ce document, deux procédures LMP supplémentaires sont définies : la vérification de connexité de liaison et la gestion des fautes. Ces procédures sont particulièrement utiles lorsque les canaux de contrôle sont physiquement différents des liaisons de données. La vérification de connexité de liaison est utilisée pour la découverte du plan des données, l'échange des identifiants d'interface (les identifiants d'interface sont utilisés dans la signalisation GMPLS, soit comme

étiquette d'un accès, soit comme identifiants de liaison composante, selon la configuration) et la vérification de la connexité physique. Ceci est fait par l'envoi de messages Test sur les liaisons de données et le retour de messages TestStatus sur le canal de contrôle. Noter que le message Test est le seul message LMP qui doive être transmis sur la liaison de données. L'échange de message ChannelStatus est utilisé entre les nœuds adjacents pour la suppression d'alarmes vers l'aval et pour la localisation des fautes aux fins de protection et restauration.

Pour la vérification de la connexité de liaison LMP, le message Test est transmis sur les liaisons de données. Pour les appareils X-transparents, cela exige d'examiner et modifier l'aspect X du signal. La procédure de vérification de la connexité de liaison LMP est coordonnée en utilisant un échange de messages BeginVerify sur un canal de contrôle. Pour prendre en charge les divers aspects de transparence, un mécanisme VerifyTransport est inclus dans les messages BeginVerify et BeginVerifyAck. Noter qu'il n'est pas exigé que toutes les liaisons de données doivent perdre leur transparence simultanément ; mais, au minimum, il doit être possible de les terminer l'une après l'autre. Il n'est pas non plus exigé que le canal de contrôle et la liaison TE utilisent le même support physique ; cependant, le canal de contrôle DOIT être terminé par les deux mêmes éléments de contrôle qui contrôlent la liaison TE. Comme l'échange de messages BeginVerify coordonne la procédure Test, elle coordonne naturellement aussi la transition des liaisons de données de et vers le mode transparent.

La procédure de gestion de faute de LMP se fonde sur un échange de messages ChannelStatus qui utilise les messages suivants : ChannelStatus, ChannelStatusAck, ChannelStatusRequest, et ChannelStatusResponse. Le message ChannelStatus est envoyé non sollicité et est utilisé pour notifier un LMP voisin de l'état d'un ou plusieurs canaux de données d'une liaison TE. Le message ChannelStatusAck est utilisé pour accuser réception du message ChannelStatus. Le message ChannelStatusRequest est utilisé pour interroger un voisin LMP sur l'état d'un ou plusieurs canaux de données d'une liaison TE. Le message ChannelStatusResponse est utilisé pour accuser réception du message ChannelStatusRequest et indique l'état des liaisons de données qui sont l'objet de l'interrogation.

3. Gestion du canal de contrôle

Pour initier une adjacence LMP entre deux nœuds, un ou plusieurs canaux de contrôle bidirectionnels DOIVENT être activés. Les canaux de contrôle peuvent être utilisés pour échanger des informations de plan de contrôle comme le provisionnement des liaisons et des informations de gestion de fautes (mis en œuvre en utilisant un protocole de messages comme LMP, proposé dans le présent document) des informations de gestion de chemin et de distribution d'étiquettes (mis en œuvre en utilisant un protocole de signalisation comme RSVP-TE [RFC3209]) et des informations de topologie et de distribution d'état (mis en œuvre en utilisant des extensions de protocoles d'ingénierie du trafic comme OSPF [RFC3630] et IS-IS [RFC3784]).

Pour les besoins de LMP, la mise en œuvre exacte du canal de contrôle n'est pas spécifiée ; ce pourrait être, par exemple, une longueur d'onde ou une fibre séparée, une liaison Ethernet, un tunnel IP à travers un réseau de gestion séparé, ou les octets de sur débit d'une liaison de données. Chaque nœud alloue un identifiant de canal de contrôle (CC_Id) qui est un entier de 32 bits non zéro, unique pour le nœud. Cet identifiant vient du même espace que l'identifiant d'interface non numérotée. De plus, les paquets LMP sont envoyés sur UDP avec un numéro d'accès LMP. Donc, le codage de niveau liaison du canal de contrôle ne fait pas partie de la spécification de LMP.

Pour établir un canal de contrôle, la destination à l'extrémité distante du canal de contrôle doit cependant être connue. Cette connaissance peut être configurée manuellement ou découverte automatiquement. Noter que pour la signalisation dans la bande, un canal de contrôle pourrait être configuré explicitement sur une certaine liaison de données. Dans ce cas, l'échange de messages Config peut être utilisé pour apprendre de façon dynamique l'adresse IP sur l'extrémité distante du canal de contrôle. Ceci se fait en envoyant le message Config avec l'adresse IP de source en envoi individuel et l'adresse IP de destination de diffusion groupée (224.0.0.1 ou ff02::1). Les messages ConfigAck et ConfigNack DOIVENT être envoyés à l'adresse IP de source trouvée dans l'en-tête IP du message Config reçu.

Les canaux de contrôle existent indépendamment des liaisons TE et plusieurs canaux de contrôle peuvent être actifs simultanément entre une paire de nœuds. Des canaux de contrôle individuels peuvent être réalisés de différentes façons ; on peut les mettre en œuvre dans la fibre ou hors de la fibre. C'est pour cela que les paramètres de canal de contrôle DOIVENT être négociés sur chaque canal de contrôle individuel, et les paquets Hello LMP DOIVENT être échangés sur chaque canal de contrôle pour conserver la connexité LMP si d'autres mécanismes ne sont pas disponibles. Comme les canaux de contrôle sont terminés électriquement à chaque nœud, il est possible de détecter les défaillances de canal de contrôle en utilisant les couches inférieures (par exemple, SONET/SDH).

Il y a quatre messages LMP qui sont utilisés pour gérer les canaux de contrôle individuels .Ce sont les messages Config, ConfigAck, ConfigNack, et Hello. Ces messages DOIVENT être transmis sur le canal auquel ils se réfèrent. Tous les autres messages LMP peuvent être transmis sur un des canaux de contrôle actifs entre une paire de nœuds LMP adjacents.

Afin de maintenir une adjacence LMP, il est nécessaire d'avoir au moins un canal de contrôle actif entre une paire de nœuds adjacents (on rappelle que plusieurs canaux de contrôle peuvent être actifs simultanément entre une paire de nœuds). En cas de défaillance d'un canal de contrôle, des canaux de contrôle actifs de remplacement peuvent être utilisés et il est possible d'activer des canaux de contrôle supplémentaires comme décrit ci-dessous.

3.1 Négociation des paramètres

L'activation de canal de contrôle commence par un échange de négociation de paramètres utilisant les messages Config, ConfigAck, et ConfigNack. Le contenu de ces messages est construit en utilisant des objets LMP, qui peuvent être négociables ou non négociables (identifiés par le bit N dans l'en-tête d'objet). Les objets négociables peuvent être utilisés pour permettre à l'homologue LMP d'accepter certaines valeurs. Les objets non négociables sont utilisés pour l'annonce de valeurs spécifiques qui n'ont pas besoin de, ou ne permettent pas, la négociation.

Pour activer un canal de contrôle, un message Config DOIT être transmis au nœud distant, et en réponse, un message ConfigAck DOIT être reçu au nœud local. Le message Config contient l'identifiant de contrôle local (CC_Id, *Control Channel Id*) l'identifiant de nœud de l'envoyeur, un identifiant de message pour la messagerie fiable, et un objet CONFIG. Il est possible que les deux nœuds, local et distant, initient la procédure de configuration en même temps. Pour éviter les ambiguïtés, le nœud qui a le plus fort identifiant de nœud gagne la compétition ; le nœud qui a le plus faible identifiant de nœud DOIT arrêter de transmettre le message Config et répondre au message Config qu'il reçoit. Si les identifiants de nœud sont égaux, l'un d'eux (ou les deux) a été mal configuré. Les nœuds PEUVENT continuer à retransmettre les messages Config dans l'espoir que la mauvaise configuration sera corrigée. Noter que le problème peut être résolu par le changement par un opérateur des identifiants de nœud sur un des nœuds, ou sur les deux.

Le message ConfigAck est utilisé pour accuser réception du message Config et exprimer l'accord sur TOUS les paramètres configurés (négociables et non négociables).

Le message ConfigNack est utilisé pour accuser réception du message Config, indiquer quels objets CONFIG non négociables (s'il en est) sont inacceptables, et pour proposer des valeurs de remplacement pour les paramètres négociables.

Si un nœud reçoit un message ConfigNack avec des valeurs de remplacement acceptables pour les paramètres négociables, le nœud DEVRAIT transmettre un message Config utilisant ces valeurs pour ces paramètres.

Si un nœud reçoit un message ConfigNack avec des valeurs de remplacement inacceptables, le nœud PEUT continuer à retransmettre les messages Config dans l'espoir que la mauvaise configuration sera corrigée. Noter que le problème peut être résolu si un opérateur change les paramètres sur un des nœuds ou les deux.

Dans le cas où plusieurs canaux de contrôle utilisent la même interface physique, l'échange de négociation de paramètres est effectué pour chaque canal de contrôle. Les divers messages de négociation de paramètres LMP sont associés à leur canal de contrôle correspondant par leur identifiant unique à l'échelle du nœud (CC_Id).

3.2 Protocole Hello

Une fois un canal de contrôle activé entre deux nœuds adjacents, le protocole LMP Hello peut être utilisé pour maintenir la connexité du canal de contrôle entre les nœuds et détecter les défaillances du canal de contrôle. Le protocole LMP Hello est destiné à être un mécanisme léger de garde en vie qui va réagir rapidement aux défaillances du canal de contrôle afin que les Hello IGP ne soient pas perdus et que les adjacences d'état de liaison associées ne soient pas supprimées sans nécessité.

3.2.1 Négociation des paramètres de Hello

Avant d'envoyer des messages Hello, les paramètres HelloInterval et HelloDeadInterval DOIVENT être acceptés par les nœuds, local et distant. Ces paramètres sont échangés dans le message Config. Le HelloInterval indique la fréquence à laquelle les messages Hello LMP seront envoyés, et est mesuré en millisecondes (ms). Par exemple, si la valeur est 150, le nœud transmetteur va envoyer le message Hello au moins toutes les 150 ms. Le HelloDeadInterval indique combien de temps un appareil devrait attendre de recevoir un message Hello avant de déclarer qu'un canal de contrôle est mort, et est mesuré en millisecondes (ms).

Le HelloDeadInterval DOIT être supérieur au HelloInterval, et DEVRAIT être d'au moins 3 fois la valeur de HelloInterval. Si le mécanisme de garde en vie rapide de LMP n'est pas utilisé, les paramètres HelloInterval et HelloDeadInterval DOIVENT être réglés à zéro.

Les valeurs de HelloInterval et HelloDeadInterval devraient être choisies avec soin pour répondre à temps aux défaillances du canal de contrôle sans causer d'encombrement. À ce titre, des valeurs différentes seront probablement configurées pour les différentes mises en œuvre de canal de contrôle. Lorsque le canal de contrôle est mis en œuvre sur une liaison directement connectée, les valeurs par défaut suggérées sont 150 ms pour le HelloInterval et 500 ms pour le HelloDeadInterval.

Lorsque un nœud a envoyé ou reçu un message ConfigAck, il peut commencer à envoyer des messages Hello. Une fois qu'il a envoyé un message Hello et reçu un message Hello valide (c'est-à-dire, avec le numéro de séquence attendu ; voir au paragraphe 3.2.2) le canal de contrôle passe à l'état actif *(up)*. (Il est aussi possible de passer à l'état actif sans envoyer de Hello si d'autres méthodes sont utilisées pour indiquer la connexité bidirectionnelle du canal de contrôle. Par exemple, l'indication de la connexité bidirectionnelle peut être apprise de la couche transport.) Si, cependant, un nœud reçoit un message ConfigNack au lieu d'un message ConfigAck, le nœud NE DOIT PAS envoyer de messages Hello et le canal de contrôle NE DEVRAIT PAS passer à l'état actif. Voir au paragraphe 11.1 l'automate à états finis (FSM, *Finite State Machine*) complet de canal de contrôle.

3.2.2 Garder en vie rapide

Chaque message Hello contient deux numéros de séquence : le premier numéro de séquence (TxSeqNum) est le numéro de séquence pour le message Hello en cours d'envoi et le second numéro de séquence (RcvSeqNum) est le numéro de séquence du dernier message Hello reçu du nœud adjacent sur ce canal de contrôle.

Il y a deux numéros de séquence spéciaux. TxSeqNum NE DOIT PAS être 0. TxSeqNum = 1 est utilisé pour indiquer que l'envoyeur vient juste de commencer ou a recommencé et n'a pas de souvenir du dernier TxSeqNum envoyé. Donc, le premier Hello envoyé a un TxSeqNum de 1 et un RxSeqNum de 0. Lorsque TxSeqNum atteint (2^32)-1, le numéro de séquence suivant utilisé est 2, ni 0 ni 1, car ils ont une signification spéciale.

En fonctionnement normal, la différence entre le RcvSeqNum dans un message Hello qui est reçu et le TxSeqNum local qui est généré sera au plus de 1. Cette différence ne peut être de plus de un que lorsque un canal de contrôle redémarre ou lorsque les valeurs reviennent au début.

Comme les numéros de séquence de 32 bits peuvent revenir au début, l'expression suivante peut être utilisée pour vérifier si une valeur de TxSeqNum qui vient d'être reçue est inférieure à une valeur reçue antérieurement :

Si ((int) old_id - (int) new_id > 0) { la nouvelle valeur est inférieure à l'ancienne valeur ; }

Avoir les numéros de séquence dans les messages Hello permet à chaque nœud de vérifier que son homologue reçoit ses messages Hello. En incluant le RcvSeqNum dans les paquets Hello, le nœud local va savoir quels paquets Hello le nœud distant a reçu.

L'exemple suivant illustre comment opère le numéro de séquence. Noter qu'on ne montre le fonctionnement qu'à un seul nœud, et d'autres scénarios sont possibles :

1) Après achèvement de l'étape de configuration, Nœud A envoie des messages Hello à Nœud B avec {TxSeqNum=1;RcvSeqNum=0}.

2) Nœud A reçoit un Hello de Nœud B avec {TxSeqNum=1;RcvSeqNum=1}. Quand le HelloInterval expire à Nœud A, il envoie des Hello à Nœud B avec {TxSeqNum=2;RcvSeqNum=1}.

3) Nœud A reçoit un Hello de Nœud B avec {TxSeqNum=2;RcvSeqNum=2}. Quand le HelloInterval expire à Nœud A, il envoie des Hello à Nœud B avec {TxSeqNum=3;RcvSeqNum=2}.

3.2.3 Canal de contrôle défaillant

Pour permettre d'arrêter en douceur un canal de contrôle pour des besoins d'administration, un fanion ControlChannelDown est disponible dans l'en-tête commun des paquets LMP. Lorsque des liaisons de données sont encore en service entre une paire de nœuds, un canal de contrôle DEVRAIT n'être arrêté administrativement que lorsque il y a d'autres canaux de contrôle actifs qui peuvent être utilisés pour gérer les liaisons de données.

Lors de l'arrêt administratif d'un canal de contrôle, un nœud DOIT établir le fanion ControlChannelDown dans tous les messages LMP envoyés sur le canal de contrôle. Le nœud qui initie la procédure d'arrêt du canal de contrôle peut cesser d'envoyer des messages Hello après HelloDeadInterval secondes, ou si il reçoit un message LMP sur le même canal de contrôle avec le fanion ControlChannelDown établi.

Lorsque un nœud reçoit un paquet LMP avec le fanion ControlChannelDown établi, il DEVRAIT envoyer un message Hello avec le fanion ControlChannelDown établi et passer le canal de contrôle à l'état arrêt *(down)*.

3.2.4 État dégradé

Permettre aux canaux de contrôle d'être physiquement différents des liaisons de données associées a pour conséquence qu'il peut n'y avoir aucun canal de contrôle actif disponible alors que des liaisons de données sont encore en service. Pour de nombreuses applications, il est inacceptable de supprimer une liaison qui porte du trafic d'utilisateur simplement parce que le canal de contrôle n'est plus disponible ; cependant, il n'est plus possible de garantir le même niveau de service au trafic qui utilise les liaisons de données. Donc, la liaison TE est dans un état dégradé *(Degraded)*.

Lorsque une liaison TE est dans l'état dégradé, l'acheminement et la signalisation DEVRAIENT en être notifiées afin que de nouvelles connexions ne soient pas acceptées et que la liaison TE soit annoncée avec pas de ressources non réservées.

4. Corrélation de propriétés de liaison

Au titre de LMP, un échange de corrélation de propriétés de liaison est défini pour les liaisons TE en utilisant les messages LinkSummary, LinkSummaryAck, et LinkSummaryNack. Le contenu de ces messages est construit en utilisant des objets LMP, qui peuvent être négociables ou non négociables (identifiés par le bit fanion N dans l'en-tête d'objet). Les objets négociables peuvent être utilisés pour permettre aux deux côtés de se mettre d'accord sur certains paramètres de liaison. Les objets non négociables sont utilisés pour l'annonce de valeurs spécifiques qui n'ont pas besoin de, ou ne permettent pas, la négociation.

Chaque liaison TE a un identifiant (Link_Id) qui est alloué à chaque extrémité de la liaison. Ces identifiants DOIVENT être du même type (c'est-à-dire, IPv4, IPv6, non numéroté) au deux extrémités. Si un message LinkSummary est reçu avec des types de liaison TE locale et distante différents, un message LinkSummaryNack DOIT être envoyé avec le code d'erreur "Mauvais objet de liaison TE". De même, chaque liaison de données reçoit un identifiant (Interface_Id) à chaque extrémité. Ces identifiants DOIVENT aussi être du même type aux deux extrémités. Si un message LinkSummary est reçu avec des types différents d'Interface_Id local et distant, un message LinkSummaryNack DOIT être envoyé avec le code d'erreur "Mauvais objet de liaison de données".

La corrélation de propriétés de liaison DEVRAIT être faite avant l'activation de la liaison et PEUT être faite à tout moment pendant que la liaison est active et n'est pas dans le processus de vérification.

Le message LinkSummary est utilisé pour vérifier la cohérence des informations de TE et de la liaison de données entre les deux côtés. Les messages LinkSummary sont aussi utilisés (1) pour agréger plusieurs liaisons de données (soit accès, soit liaisons composantes) en une liaison TE ; (2) pour échanger, corréler (pour déterminer les incohérences) ou changer les paramètres de liaison TE ; et (3) pour échanger, corréler (pour déterminer les incohérences) ou changer les identifiants d'interface (identifiant d'accès ou de liaison composante).

Le message LinkSummary inclut un objet TE_LINK suivi par un ou plusieurs objets DATA_LINK. L'objet TE_LINK identifie les identifiants de liaison local et distant de la liaison TE et indique la prise en charge des procédures de gestion de faute et de vérification de liaison pour cette liaison TE. Les objets DATA_LINK sont utilisés pour caractériser les liaisons de données qui composent la liaison TE. Ces objets incluent les identifiants d'interface local et distant, et peuvent inclure un ou plusieurs sous objets décrivant plus en détails les propriétés des liaisons de données.

Si le message LinkSummary est reçu d'un nœud distant, et si les transpositions d'identifiant d'interface correspondent à celles mémorisées en local, les deux nœuds sont en accord sur la procédure de vérification (voir à la Section 5) et la configuration d'identification de liaison de données. Si la procédure de vérification n'est pas utilisée, le message LinkSummary peut être utilisé pour vérifier l'accord sur une configuration manuelle.

Le message LinkSummaryAck est utilisé pour signaler l'accord sur les transpositions d'identifiant d'interface et les définitions des propriétés de liaison. Autrement, un message LinkSummaryNack DOIT être transmis, indiquant quelles transpositions d'interface ne sont pas correctes et/ou quelles propriétés de liaison ne sont pas acceptées. Si un message LinkSummaryNack indique que les transpositions d'identifiant d'interface ne sont pas correctes et si la procédure de vérification de liaison est activée, le processus de vérification de liaison DEVRAIT être répété pour toutes les liaisons de données libres qui ne correspondent pas ; si une liaison de données allouée a une discordance de transposition, elle DEVRAIT être marquée d'un fanion et vérifiée lorsque elle devient libre. Si un message LinkSummaryNack inclut des paramètres négociables, les valeurs acceptables pour ces paramètres DOIVENT alors être incluses. Si un message LinkSummaryNack est reçu et comporte des paramètres négociables, l'initiateur du message LinkSummary DEVRAIT

alors envoyer un nouveau message LinkSummary. Le nouveau message LinkSummary DEVRAIT inclure les nouvelles valeurs pour les paramètres négociables. Ces valeurs DEVRAIENT tenir compte des valeurs acceptables reçues dans le message LinkSummaryNack.

Il est possible que le message LinkSummary devienne assez gros à cause du nombre d'objets DATA LINK. Une mise en œuvre de LMP DEVRAIT être capable de fragmenter lors de la transmission de messages LMP, et DOIT être capable de réassembler les fragments IP lors de la réception de messages LMP.

5. Vérification de la connexité de liaison

Cette Section décrit une procédure facultative qui peut être utilisée pour vérifier la connexité physique des liaisons de données et apprendre de façon dynamique (c'est-à-dire, découvrir) les associations de liaison TE et d'identifiant d'interface. La procédure DEVRAIT être effectuée lors de l'établissement d'une liaison TE, et ensuite, de façon périodique pour toutes les liaisons de données non allouées (libres) de la liaison TE.

La prise en charge de cette procédure est indiquée par l'établissement du fanion "Prise en charge de la vérification de liaison" dans l'objet TE_LINK du message LinkSummary.

Si un message BeginVerify est reçu et si la vérification de liaison n'est pas prise en charge pour la liaison TE, un message BeginVerifyNack DOIT alors être transmis avec un code d'erreur indiquant "Procédure de vérification de liaison non prise en charge pour cette liaison TE".

Une caractéristique unique des appareils transparents est que les données ne sont pas modifiées ni examinée durant le fonctionnement normal. Cette caractéristique lance un défi à la validation de la connexité des liaisons de données et à l'établissement des transpositions d'étiquettes. Donc, pour assurer une vérification appropriée de la connexité de la liaison de données, il est exigé que, jusqu'à ce que les liaisons de données soient allouées au trafic d'utilisateur, elles soient opaques (c'est-à-dire, perdent leur transparence). Pour prendre en charge les divers degrés d'opacité (par exemple, examiner les octets de sur débit, terminer la charge utile IP, etc.) et donc, les différents mécanismes pour transporter les messages Test, un champ "Vérifier le mécanisme de transport" est inclus dans les messages BeginVerify et BeginVerifyAck.

Il n'est pas exigé que toutes les liaisons de données soient terminées simultanément; mais au minimum, les liaisons de données DOIVENT être capables de se terminer l'une après l'autre. De plus, on suppose pour la procédure de vérification de liaison que l'architecture des nœuds est conçue de telle sorte que les messages puissent être envoyés et reçus sur toute liaison de données. Noter que cette exigence est triviale pour les appareils opaques car chaque liaison de données est terminée électriquement et traitée avant d'être transmise au prochain appareil opaque, mais que dans les appareils transparents c'est une exigence supplémentaire.

Pour interconnecter deux nœuds, une liaison TE est définie entre eux, et au minimum, il DOIT y avoir au moins un canal de contrôle actif entre les nœuds. Pour la vérification de liaison, une liaison TE DOIT inclure au moins une liaison de données.

Une fois qu'un canal de contrôle a été établi entre les deux nœuds, la connexité des liaisons de données peut être vérifiée par l'échange de messages Test sur chacune des liaisons de données spécifiées dans la liaison TE. Il devrait être noté que tous les messages LMP sauf le message Test sont échangés sur les canaux de contrôle et que les messages Hello continuent d'être échangés sur chaque canal de contrôle durant le procès de vérification de la liaison de données. Le message Test est envoyé sur la liaison de données qui est vérifiée. Les liaisons de données sont vérifiées dans la direction émission parce que elles sont unidirectionnelles ; donc, il est possible pour les deux nœuds d'échanger simultanément (indépendamment) des messages Test.

Pour initier la procédure de vérification de liaison, le nœud local DOIT envoyer un message BeginVerify sur un canal de contrôle. Pour limiter la portée de la vérification de liaison à une liaison TE particulière, l'identifiant de liaison locale DOIT être différent de zéro. Si ce champ est à zéro, les liaisons de données peuvent s'étendre sur plusieurs liaisons TE et/ou elles peuvent comprendre une liaison TE qui reste à configurer. Dans le cas où le champ Link_Id local est zéro, le fanion "Vérifier toutes les liaisons" de l'objet BEGIN_VERIFY est utilisé pour distinguer entre liaisons de données qui s'étendent sur plusieurs liaisons TE et celles qui n'ont pas encore été allouées à une TE. Précisément, la vérification de liaisons de données qui s'étendent sur plusieurs liaisons TE est indiquée par le réglage à zéro du champ Link_Id local et par l'établissement du fanion "Vérifier toutes les liaisons". La vérification des liaisons de données qui n'ont pas encore été allouées à une liaison TE est indiquée par le réglage à zéro du champ Link_Id local et du fanion "Vérifier toutes les liaisons".

Le message BeginVerify contient aussi le nombre de liaisons de données à vérifier, l'intervalle (appelé VerifyInterval) auquel les messages Test seront envoyés, le schéma de codage et les mécanismes de transport pris en charge, le débit de

données des messages Test, et, quand les liaisons de données correspondent à des fibres, l'identifiant de longueur d'onde sur laquelle les messages Test seront transmis.

Si le nœud distant reçoit un message BeginVerify et si il est prêt à traiter les messages Test, il DOIT renvoyer un message BeginVerifyAck au nœud local spécifiant le mécanisme de transport désiré pour les messages Test. Le nœud distant inclut un Verify_Id de 32 bits, unique pour le nœud dans le message BeginVerifyAck. Le Verify_Id PEUT être choisi au hasard ; cependant, il NE DOIT PAS se chevaucher avec un autre Verify_Id actuellement utilisé par le nœud qui le choisit. Le Verify_Id est alors utilisé dans tous les messages de vérification correspondants pour les différencier des différents homologues LMP et/ou des différentes procédures Test parallèles. Lorsque le nœud local reçoit un message BeginVerifyAck du nœud distant, il peut commencer à vérifier les liaisons de données en émettant des messages Test périodiques sur chaque liaison de données. Le message Test inclut le Verify_Id et le Interface_Id local pour la liaison de données associée. Le nœud distant DOIT envoyer soit un message TestStatusSuccess, soit un message TestStatusFailure en réponse pour chaque liaison de données. Un message TestStatusAck DOIT être envoyé pour confirmer la réception des messages TestStatusSuccess et TestStatusFailure. Les messages TestStatusSuccess et TestStatusFailure non acquittés DEVRAIENT être retransmis jusqu'à ce que le message reçoive un accusé de réception ou jusqu'à ce que la limite d'essais soit atteinte (voir à la Section 10).

Il est aussi permis à l'envoyeur de terminer la procédure Test à tout moment après l'envoi du message BeginVerify. Un message EndVerify DEVRAIT être envoyé à cette fin.

La corrélation de message est faite en utilisant les identifiants de message et le Verify_Id ; cela permet la vérification des liaisons de données, appartenant aux différents faisceaux de liaisons ou sessions LMP, en parallèle.

Lorsque le message Test est reçu, l'identifiant d'interface reçu (utilisé dans GMPLS soit comme étiquette d'accès, soit comme identifiant de liaison composante, selon la configuration) est enregistré et transposé en identifiant d'interface local pour cette liaison de données, et un message TestStatusSuccess DOIT être envoyé. Le message TestStatusSuccess comporte l'identifiant d'interface local ainsi que le Interface_Id et le Verify_Id reçus dans le message Test. La réception d'un message TestStatusSuccess indique que le message Test a été détecté au nœud distant et que la connexité physique de la liaison de données a été vérifiée. Lorsque le message TestStatusSuccess est reçu, le nœud local DEVRAIT marquer la liaison de données comme active et envoyer un message TestStatusAck au nœud distant. Si, cependant, le message Test n'est pas détecté au nœud distant dans une période d'observation (spécifiée par le VerifyDeadInterval) le nœud distant DOIT envoyer un message TestStatusFailure sur le canal de contrôle, qui indique que la vérification de la connexité physique de la liaison de données a échoué. Lorsque le nœud local reçoit un message TestStatusFailure, il DEVRAIT marquer la liaison de données comme ÉCHEC et envoyer un message TestStatusAck au nœud distant. Lorsque toutes les liaisons de données de la liste ont été vérifiées, le nœud local DEVRAIT envoyer un message EndVerify pour indiquer que la vérification est achevée sur cette liaison.

Si les transpositions de liaisons de données local/distante sont connues, la procédure de vérification de liaison peut être optimisée en vérifiant les liaisons de données dans un ordre défini connu des deux nœuds. Le critère suggéré pour cet ordre est la valeur croissante de l'identifiant d'interface distante.

Les deux nœuds, local et distant, DEVRAIENT conserver la liste complète des transpositions d'identifiant d'interface pour les besoins de corrélation.

5.1 Exemple de vérification de la connexité de la liaison

La Figure 1 montre un exemple de scénario de vérification de liaison qui est exécuté lorsque la liaison entre Nœud A et Nœud B est ajoutée. Dans cet exemple, la liaison TE consiste en trois accès libres (chacun transmis sur une fibre séparée) et est associée à un canal de contrôle bidirectionnel (indiqué par un "c"). Le processus de vérification est le suivant :

- o A envoie un message BeginVerify sur le canal de contrôle à B, indiquant qu'il va commencer à vérifier les accès qui forment la liaison TE. L'objet LOCAL_LINK_ID porté dans le message BeginVerify porte l'identifiant (adresse IP ou indice d'interface) que A alloue à la liaison.
- o À réception du message BeginVerify, B crée un Verify_Id et le lie à la liaison TE à partir de A. Ce lien est utilisé ultérieurement lorsque B reçoit les messages Test de A, et ces messages portent le Verify_Id. B découvre l'identifiant (adresse IP ou indice d'interface) que A alloue à la liaison TE en examinant l'objet LOCAL_LINK_ID porté dans le message BeginVerify reçu. (Si les accès de données ne sont pas encore alloués à la liaison TE, le lien se limite à l'identifiant de nœud de A.) En réponse au message BeginVerify, B envoie le message BeginVerifyAck à A. L'objet LOCAL_LINK_ID porté dans le message BeginVerifyAck est utilisé pour porter l'identifiant (adresse IP ou indice d'interface) que B alloue à la liaison TE. L'objet REMOTE_LINK_ID porté dans le message BeginVerifyAck est utilisé pour lier les identifiants de liaison alloués par A et B. Le Verify_Id est retourné à A dans le message BeginVerifyAck sur le canal de contrôle.
- o Lorsque A reçoit le message BeginVerifyAck, il commence à transmettre des messages Test périodiques sur le premier

accès (Interface_Id=1). Le message Test inclut l'Interface_Id pour l'accès et le Verify_Id qui a été alloué par B.

- Lorsque B reçoit les messages Test, il transpose l'Interface_Id reçu en son propre Interface_Id = 10 local et retransmet un message TestStatusSuccess sur le canal de contrôle au nœud A. Le message TestStatusSuccess inclut les deux Interface_Id local et distant reçus pour l'accès ainsi que le Verify_Id. Le Verify_Id est utilisé pour déterminer les identifiants de liaison TE local/distant (adresses IP ou indices d'interface) auxquels appartiennent les liaisons de données.
- A va renvoyer un message TestStatusAck sur le canal de contrôle à B, indiquant qu'il a reçu le message TestStatusSuccess.
- Le processus est répété jusqu'à ce que tous les accès soient vérifiés.
- À ce point, A va envoyer un message EndVerify sur le canal de contrôle à B, indiquant que la vérification est achevée.
- B va répondre en renvoyant à A un message EndVerifyAck sur le canal de contrôle.

Noter que cette procédure peut être utilisée pour "découvrir" la connexité des accès de données.

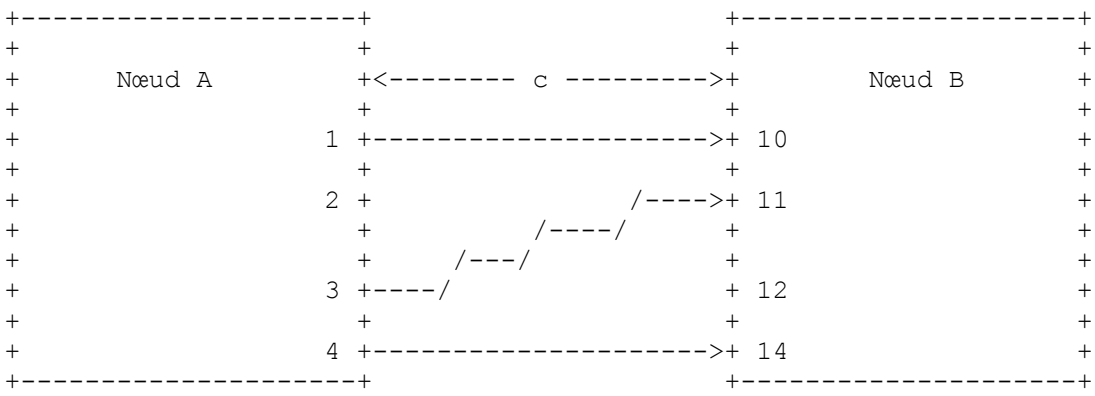


Figure 1 : Exemple de connexité de liaison entre Nœud A et Nœud B.

6. Gestion des fautes

Dans cette section, une procédure LMP facultative est décrite qui est utilisée pour gérer les défaillances par une notification rapide de l'état d'un ou plusieurs canaux de données d'une liaison TE. La portée de cette procédure est une liaison TE, et à ce titre, l'utilisation de cette procédure est négociée dans l'échange LinkSummary. La procédure peut être utilisée pour isoler rapidement les défaillances de liaison de données et de liaison TE, et elle est conçue pour fonctionner sur les LSP unidirectionnels et bidirectionnels.

Une implication importante de l'utilisation d'appareils transparents est que les méthodes traditionnelles qui sont utilisées pour surveiller la santé des liaisons de données allouées peuvent n'être plus appropriées. Au lieu que la détection de fautes soit dans la couche 2 ou la couche 3, elle est déléguée à la couche physique (c'est-à-dire, perte de lumière ou surveillance optique des données).

On se rappelle qu'une liaison TE qui connecte deux nœuds peut consister en un certain nombre de liaisons de données. Si une ou plusieurs liaisons de données ont une défaillance entre deux nœuds, un mécanisme doit être utilisé pour une notification rapide de la défaillance afin que les mécanismes appropriés de protection/restauration puissent être initiés. Si la défaillance est ensuite réparée, un mécanisme doit être utilisé pour notifier que la défaillance est réparée et que l'état du canal est OK.

6.1 Détection de faute

La détection de faute devrait être traitée à la couche la plus proche de la défaillance ; pour les réseaux optiques, c'est la couche physique (optique). Une mesure de détection de faute à la couche physique est de détecter la perte de lumière (LOL, *Loss Of Light*). D'autres techniques pour surveiller les signaux optiques restent à développer et ne seront pas considérées dans le présent document. Cependant, il devrait être clair que le mécanisme utilisé pour la notification des fautes dans LMP est indépendant du mécanisme utilisé pour détecter la défaillance, et s'appuie simplement sur le fait qu'une défaillance est détectée.

6.2 Procédure de localisation de faute

Dans certaines situations, une défaillance de liaison de données entre deux nœuds est propagée en aval de telle sorte que tous les nœuds vers l'aval détectent la défaillance sans la localiser. Pour éviter que plusieurs alarmes découlent de la même défaillance, LMP fournit une notification de défaillance grâce au message ChannelStatus. Ce message peut être utilisé pour indiquer qu'un seul canal de données est défaillant, que plusieurs canaux de données sont défaillants, ou qu'une liaison TE entière est défaillante. La corrélation de défaillance est faite localement à chaque nœud à réception de la notification de défaillance.

Pour localiser une faute sur une certaine liaison entre des nœuds adjacents, un nœud aval (en termes de flux de données) qui détecte des défaillances de liaison de données va envoyer un message ChannelStatus à son voisin amont indiquant qu'une défaillance a été détectée (faisant un bouquet des notifications de toutes les liaisons de données défaillantes). Un nœud amont qui reçoit le message ChannelStatus DOIT envoyer un message ChannelStatusAck au nœud aval indiquant qu'il a reçu le message ChannelStatus. Le nœud amont devrait corréler les défaillances pour voir si elles sont aussi détectées en local pour le ou les LSP correspondants. Si, par exemple, la défaillance n'existe pas sur l'entrée du nœud amont ou en interne, le nœud amont aura alors localisé la défaillance. Une fois la défaillance corrélée, le nœud amont DEVRAIT envoyer un message ChannelStatus au nœud aval indiquant que le canal est défaillant ou est OK. Si un message ChannelStatus n'est pas reçu par le nœud aval, il DEVRAIT envoyer un message ChannelStatusRequest pour le canal en question. Une fois la défaillance localisée, le protocole de signalisation peut être utilisé pour initier les procédures de protection et restauration de portée ou de chemin.

Si toutes les liaisons de données d'une liaison TE sont défaillantes, le nœud amont peut être notifié de la défaillance de la liaison TE sans spécifier chaque liaison de données de la liaison TE défaillante. On fait cela en envoyant une notification de défaillance dans un message ChannelStatus qui identifie la liaison TE sans inclure les Interface_Id dans l'objet CHANNEL_STATUS.

6.3 Exemples de localisation de faute

La Figure 2 montre un exemple de réseau où quatre nœuds sont connectés dans une configuration linéaire. Les canaux de contrôle sont bidirectionnels et sont marqués d'un "c". Tous les LSP sont aussi bidirectionnels.

Dans le premier exemple [Figure 2(a)], il y a une défaillance dans une direction du LSP bidirectionnel. Le Nœud 4 va détecter la défaillance et va envoyer un message ChannelStatus au Nœud 3 indiquant la défaillance (par exemple, LOL) au nœud amont correspondant. Lorsque Nœud 3 reçoit le message ChannelStatus de Nœud 4, il retourne un message ChannelStatusAck au Nœud 4 et corrèle la défaillance en local. Lorsque Nœud 3 corrèle la défaillance et vérifie que la défaillance n'existe pas, il a localisé la défaillance à la liaison de données entre Nœud 3 et Nœud 4. À ce moment, Nœud 3 devrait envoyer un message ChannelStatus au Nœud 4 indiquant que la défaillance a été localisée.

Dans le second exemple [Figure 2(b)], une seule défaillance (par exemple, coupure de fibre) affecte les deux directions du LSP bidirectionnel. Nœud 2 (Nœud 3) va détecter la défaillance de la direction amont (aval) et envoyer un message ChannelStatus au nœud amont (en termes de flux de données) indiquant la défaillance (par exemple, LOL). Simultanément (ignorant les délais de propagation) Nœud 1 (Nœud 4) va détecter la défaillance dans la direction amont (aval) et va envoyer un message ChannelStatus au nœud amont correspondant (en termes de flux de données) indiquant la défaillance. Nœud 2 et Nœud 3 vont avoir localisé les deux directions de la défaillance.

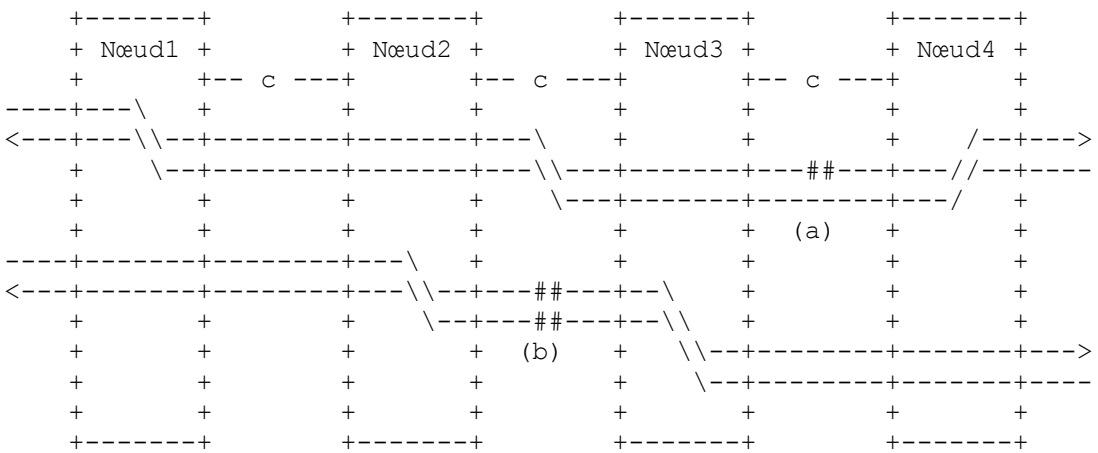


Figure 2 : Défaillance de liaisons de données

Figure 2 : Deux types de défaillance de liaison de données sont montrées (indiquées par ## sur la figure) :
(a) une liaison de données correspondant à la direction vers l'aval d'un LSP bidirectionnel est défaillante.
(b) deux liaisons de données correspondant aux deux directions d'un LSP bidirectionnel sont défaillantes. Le canal de contrôle connectant deux nœuds est indiqué par un "c".

6.4 Indication d'activation de canal

Le message ChannelStatus peut aussi être utilisé pour notifier à un voisin LMP que la liaison de données devrait être activement surveillée. C'est ce qu'on appelle Indication d'activation de canal. C'est particulièrement utile dans des réseaux avec des nœuds transparents où l'état des liaisons de données peut devoir être déclenché en utilisant un message de canal de contrôle. Par exemple, si une liaison de données est pré provisionnée et si la liaison physique est défaillante après vérification et avant d'insérer du trafic d'utilisateur, un mécanisme est nécessaire pour indiquer que la liaison de données devrait être activée, autrement, la défaillance peut n'être pas détectable.

Le message ChannelStatus est utilisé pour indiquer qu'un canal, ou groupe de canaux, n'est actuellement pas actif. Le message ChannelStatusAck DOIT être transmis dès réception d'un message ChannelStatus. Lorsque un message ChannelStatus est reçu, la ou les liaisons de données correspondantes DOIVENT être mises dans l'état Actif. Si en les mettant dans l'état Actif, une défaillance est détectée, le message ChannelStatus DEVRAIT être transmis comme décrit au paragraphe 6.2.

6.5 Indication de désactivation de canal

Le message ChannelStatus peut aussi être utilisé pour notifier à un voisin LMP qu'une liaison de données n'a plus besoin d'être surveillée activement. C'est la contrepartie de l'indication Canal actif

Lorsque un message ChannelStatus est reçu avec l'indication de désactivation de canal, la ou les liaisons de données correspondantes DOIVENT être retirées de l'état Actif.

7. Usage de Message_Id

Les objets MESSAGE_ID et MESSAGE_ID_ACK sont inclus dans les messages LMP pour prendre en charge la livraison fiable des messages. Cette Section décrit l'usage de ces objets. Les objets MESSAGE_ID et MESSAGE_ID_ACK contiennent un champ Message_Id.

Un seul objet MESSAGE_ID/MESSAGE_ID_ACK peut être inclus dans un message LMP.

Pour les messages spécifiques du canal de contrôle, le champ Message_Id est dans la portée du CC_Id. Pour les messages spécifiques de liaisons TE, le champ Message_Id est dans la portée de l'adjacence LMP.

Le champ Message_Id de l'objet MESSAGE_ID contient une valeur choisie par le générateur. Cette valeur DOIT être à croissance monotone. Une valeur est considérée être utilisée précédemment lorsque elle a été envoyée dans un message LMP avec le même CC_Id (pour les messages spécifiques du canal de contrôle) ou adjacence LMP (pour les messages spécifiques de liaison TE). Le champ Message_Id de l'objet MESSAGE_ID_ACK contient le champ Message_Id du message dont on accuse réception.

Les messages non acquittés envoyés avec l'objet MESSAGE_ID DEVRAIENT être retransmis jusqu'à ce que le message soit acquitté ou jusqu'à ce qu'une limite de répétition des essais soit atteinte (voir aussi la Section 10).

Noter que la valeur de 32 bits du Message_Id peut revenir à l'origine. L'expression suivante peut être utilisée pour vérifier si une valeur nouvellement reçue de Message_Id est inférieure à une valeur reçue précédemment :

Si ((int) old_id - (int) new_id > 0) { La nouvelle valeur est inférieure à l'ancienne valeur; }

Les nœuds qui traitent les messages entrants DEVRAIENT vérifier si un message qui vient d'être reçu est déclassé et peut être ignoré. Les messages déclassés peuvent être identifiés en examinant la valeur dans le champ Message_Id. Si il est déterminé qu'un message est déclassé, il devrait être éliminé en silence.

Si le message est un message Config, et si la valeur du Message_Id est inférieure à la plus grande valeur de Message_Id reçue antérieurement de l'envoyeur pour ce CC_Id, le message DEVRAIT alors être traité comme déclassé.

Si le message est un message LinkSummary et si la valeur du Message_Id est inférieure à la plus grande valeur de

Message_Id reçue précédemment de l'envoyeur pour la liaison TE, le message DEVRAIT alors être traité comme déclassé.

Si le message est un message ChannelStatus et si la valeur du Message_Id est inférieure à la plus grande valeur de Message_Id reçue précédemment de l'envoyeur pour la liaison TE spécifiée , le receveur DEVRAIT alors vérifier la valeur du Message_Id reçue précédemment avec l'état de chaque canal de données inclus dans le message ChannelStatus. Si la valeur du Message_Id est supérieure à celle du plus récent Message_Id reçu associé à au moins un des canaux de données inclus dans le message, le message NE DOIT PAS être traité comme déclassé ; autrement, le message DEVRAIT être traité comme déclassé. Cependant, l'état de tout canal de données NE DOIT PAS être mis à jour si la valeur de Message_Id est inférieure à celle du plus récent Message_Id reçu associé à ce canal de données.

Aucun autre message NE DOIT PAS être traité comme déclassé.

8. Redémarrage en douceur

Cette section décrit le mécanisme pour resynchroniser l'état LMP après un redémarrage du plan de contrôle. Un redémarrage du plan de contrôle peut survenir lors de l'activation du premier canal de contrôle après une défaillance des communications de contrôle. Une défaillance des communications de contrôle peut être le résultat d'une défaillance d'adjacence LMP ou d'une défaillance d'un nœud suivant laquelle l'état de contrôle LMP est perdu, mais le plan des données n'est pas affecté. Cette dernière est détectée en réglant le bit "Redémarrage LMP" dans l'en-tête commun du message LMP. Lorsque le plan de contrôle est défaillant du fait de la perte du canal de contrôle, les informations de la liaison LMP devraient être conservées. Il est possible qu'un nœud soit capable de conserver les informations de la liaison LMP à travers la défaillance d'un nœud. Cependant, dans les deux cas, l'état des canaux de données DOIT être synchronisé.

On suppose que les identifiants de nœud et d'interface locale restent stables à travers un redémarrage de plan de contrôle.

Après le redémarrage du plan de contrôle d'un nœud, le ou les canaux de contrôle doivent être rétablis en utilisant les procédures du paragraphe 3.1. Lors du rétablissement des canaux de contrôle, le message Config DEVRAIT être envoyé en utilisant les adresses de source et de destination d'envoi individuel IP.

Si la défaillance du plan de contrôle était le résultat de la défaillance d'un nœud et que l'état de contrôle LMP est perdu, le fanion "Redémarrage LMP" DOIT être établi dans les messages LMP jusqu'à ce qu'un message Hello soit reçu avec le RcvSeqNum *(numéro de séquence de réception)* égal au TxSeqNum *(numéro de séquence d'émission)* local. Cela indique que le canal de contrôle est actif et que le voisin LMP a détecté le redémarrage.

Dans ce qui suit, on suppose que le redémarrage du composant LMP ne se produit que sur une extrémité de la liaison TE. Si le redémarrage du composant LMP se produit sur les deux extrémités de la liaison TE, les procédures normales de LinkSummary devraient être utilisées, comme décrit à la Section 4.

Une fois qu'un canal de contrôle est activé, le voisin LMP DOIT envoyer un message LinkSummary pour chaque liaison TE à travers l'adjacence. Tous les objets du message LinkSummary DOIVENT avoir le bit N à zéro, ce qui indique que les paramètres sont non négociables. Cela fournit les transpositions d'identifiant d'interface et d'identifiant de liaison locale/distante, les paramètres associés de liaison de données, et l'indication de quelles liaisons de données sont actuellement allouées au trafic d'utilisateur. Lorsque un nœud reçoit le message LinkSummary, il vérifie sa configuration locale. Si le nœud est capable de conserver les informations de liaison LMP à travers un redémarrage, il doit traiter le message LinkSummary comme décrit à la Section 4 avec l'exception que le fanion alloué/désalloué de l'objet DATA_LINK reçu dans le message LinkSummary DOIT prendre le pas sur toute valeur locale. Si, cependant, le nœud n'a pas été capable de conserver les informations de liaison LMP à travers le redémarrage, il DOIT accepter les paramètres de liaison de données du message LinkSummary reçu et répondre par un message LinkSummaryAck.

À l'achèvement de l'échange LinkSummary, le nœud qui a redémarré le plan de contrôle DEVRAIT envoyer un message ChannelStatusRequest pour cette liaison TE. Le nœud DEVRAIT aussi vérifier la connexité de tous les canaux de données non alloués.

9. Adressage

Tous les messages LMP sont envoyés sur UDP avec un numéro d'accès LMP (sauf, dans certains cas, les messages Test, qui peuvent être limités par le mécanisme de transport pour la messagerie dans la bande). L'adresse de destination du paquet IP PEUT être l'adresse apprise dans la procédure de configuration (c'est-à-dire, l'adresse IP de source trouvée dans l'en-tête IP du message Config reçu) une adresse IP configurée sur le nœud distant, ou l'identifiant de nœud. Le message Config est une

exception comme décrit ci-dessous.

La manière dont un message Config est adressé peut dépendre du mécanisme de transport de la signalisation. Lorsque le mécanisme de transport est une liaison point à point, les messages Config DEVRAIENT être envoyés à l'adresse de diffusion groupée (224.0.0.1 ou ff02::1). Autrement, les messages Config DOIVENT être envoyés à une adresse IP sur le nœud voisin. Cela peut être configuré aux deux extrémités du canal de contrôle ou peut être découvert automatiquement.

10. Procédures de retard exponentiel

Cette Section se fonde sur la [RFC2961] et décrit les procédures de retard exponentiel pour la retransmission de message. Les mises en œuvre DOIVENT utiliser les procédures décrites ou leur équivalent.

10.1 Fonctionnement

Le fonctionnement suivant est un des mécanismes possibles pour le retard exponentiel de retransmission des messages LMP non acquittés. Le nœud envoyeur retransmet le message jusqu'à ce qu'un message d'accusé de réception soit reçu ou qu'une limite de répétition des essais soit atteinte. Lorsque le nœud envoyeur reçoit l'accusé de réception, la retransmission de message est arrêtée. L'intervalle entre les retransmissions de message est gouvernée par un temporisateur de retransmission rapide. Le temporisateur de retransmission rapide commence à un faible intervalle et augmente de façon exponentielle jusqu'à atteindre un seuil.

Les paramètres temporels suivants sont utiles pour caractériser les procédures :

Intervalle de retransmission rapide Ri : Ri est l'intervalle de retransmission initial pour les messages qui n'ont pas reçu d'accusé de réception. Après l'envoi du message pour la première fois, le nœud envoyeur va programmer une retransmission après Ri millisecondes.

Limite de répétition des essais rapides Rl : Rl est le nombre maximum de fois qu'un message sera transmis sans recevoir d'accusé de réception.

Valeur d'incrément Delta : Delta gouverne la vitesse à laquelle l'envoyeur augmente l'intervalle de retransmission. Le ratio de deux intervalles de retransmission successifs est (1 + Delta).

Les valeurs par défaut suggérées pour un intervalle initial de retransmission (Ri) de 500 ms sont une puissance de 2 de retard exponentiel (Delta = 1) et une limite de répétition d'essais de 3.

10.2 Algorithme de retransmission

Après qu'un nœud a transmis un message exigeant un accusé de réception, il devrait immédiatement programmer une retransmission après Ri secondes. Si un message d'accusé de réception correspondant est reçu avant Ri secondes, la retransmission de message DEVRAIT alors être annulée. Autrement, il va retransmettre le message après (1+Delta)*Ri secondes. La retransmission va continuer jusqu'à ce que, soit un message d'accusé de réception approprié soit reçu, soit que la limite de répétition d'essai rapide, Rl, soit atteinte.

Un nœud envoyeur peut utiliser l'algorithme suivant lors de la transmission d'un message qui exige un accusé de réception :

```
Avant la transmission initiale, initialiser Rk = Ri et Rn = 0.
si (Rn++ < Rl) {transmettre le message;
     réveiller après Rk millisecondes;
     Rk = Rk * (1 + Delta);
    }                          /* accusé de réception du message ou pas de réponse du receveur et Rl atteint*/
    faire tout nettoyage nécessaire;
    sortie;
```

En asynchrone, lorsque un nœud envoyeur reçoit un message d'accusé de réception correspondant, il va changer le compte de répétition d'essais, Rn, en Rl.

Noter que le nœud émetteur n'annonce ni ne négocie l'utilisation des procédures décrites de retard exponentiel dans les messages Config ou LinkSummary.

11. Automates à états finis LMP

11.1 Automate à états finis de canal de contrôle

L'automate à états finis (FSM, *Finite State Machine*) du canal de contrôle définit les états et la logique du fonctionnement d'un canal de contrôle LMP.

11.1.1 États du canal de contrôle

Un canal de contrôle (CC) peut être dans un des états décrits ci-dessous. Chaque état correspond à une certaine condition du canal de contrôle et est généralement associé à un type spécifique de message LMP qui est transmis périodiquement à l'extrémité distante.

Down *(arrêt)* : c'est l'état initial du canal de contrôle. Dans cet état, aucune tentative n'est faite pour activer le canal de contrôle et aucun message LMP n'est envoyé. Les paramètres du canal de contrôle devraient être réglés à leur valeur initiale.

ConfSnd *(configuration envoyée)* : le canal de contrôle est dans l'état de négociation de paramètres. Dans cet état, le nœud envoie périodiquement un message Config, et s'attend à ce que l'autre côté réponde avec un message ConfigAck ou ConfigNack. Le FSM ne passe pas à l'état Actif tant que le côté distant n'a pas accusé positivement réception des paramètres.

ConfRcv *(configuration reçue)* : le canal de contrôle est dans l'état de négociation des paramètres. Dans cet état, le nœud attend des paramètres de configuration acceptables de la part du côté distant. Une fois que de tels paramètres sont reçus et acquittés, le FSM peut passer à l'état Actif.

Actif : Dans cet état le nœud envoie périodiquement un message Hello et attend de recevoir un message Hello valide. Une fois qu'un message Hello valide a été reçu, il peut passer à l'état opérationnel *(Up)*.

Up : le canal de contrôle est dans l'état opérationnel. Le nœud reçoit des messages Hello valides et envoie des messages Hello.

GoingDown *(en train de fermer)* : un CC peut passer à cet état par suite d'une action administrative. Quand un CC est dans cet état, le nœud établit le bit ControlChannelDown dans tous les messages qu'il envoie.

11.1.2 Événements du canal de contrôle

Le fonctionnement du canal de contrôle LMP est décrit en termes d'états et événements de FSM. Les événements de canal de contrôle sont générés par les modules sous-jacents de protocoles et logiciels, ainsi que par les sous programmes de traitement de paquet et les automates à états des liaisons TE associées. Chaque événement a son numéro et un nom symbolique. La description des événements de canal de contrôle possibles est donnée ci-dessous.

1. evBringUp *(activation)* : c'est un événement déclenché en externe qui indique que la négociation du canal de contrôle devrait commencer. Cet événement, par exemple, peut être déclenché par une commande de l'opérateur, par l'achèvement réussi d'une procédure d'amorçage d'un canal de contrôle, ou par configuration. Selon la configuration, cela va déclencher soit
   1a) l'envoi d'un message Config,
   1b) soit une période d'attente de la réception d'un message Config du nœud distant.

2. evCCDn *(CC en arrêt)* : cet événement est généré lorsque il est indiqué que le canal de contrôle n'est plus disponible.

3. evConfDone : cet événement indique qu'un message ConfigAck a été reçu, accusant réception des paramètres de configuration.

4. evConfErr : cet événement indique qu'un message ConfigNack a été reçu, rejetant les paramètres de configuration.

5. evNewConfOK : un nouveau message Config a été reçu du voisin et a été acquitté positivement.

6. evNewConfErr : un nouveau message Config a été reçu du voisin et rejeté avec un message ConfigNack.

7. evContenWin : un nouveau message Config a été reçu du voisin au même moment qu'un message Config était envoyé au voisin. Le nœud local gagne le concours. Par suite, le message Config reçu est ignoré.

8. evContenLost : un nouveau message Config a été reçu du voisin au même moment qu'un message Config était envoyé au voisin. Le nœud local perd le concours .
   8a) le message Config est acquitté positivement.
   8b) le message Config est acquitté négativement.

9. evAdminDown : l'administrateur a demandé que le canal de contrôle soit arrêté pour des raisons administratives.

10. evNbrGoesDn : un paquet avec le fanion ControlChannelDown est reçu du voisin.

11. evHelloRcvd : un paquet Hello avec le numéro de séquence attendu a été reçu.

12. evHoldTimer : le temporisateur HelloDeadInterval est arrivé à expiration ce qui indique qu'aucun paquet Hello n'a été reçu. Cela fait repasser le canal de contrôle à l'état Négociation, et selon la configuration locale, cela va déclencher soit
   12a) l'envoi de messages Config périodiques,
   12b) soit une période d'attente de recevoir des messages Config du nœud distant.

13. evSeqNumErr : un Hello avec un numéro de séquence inattendu est reçu et éliminé.

14. evReconfig : les paramètres du canal de contrôle ont été reconfigurés et exigent une renégociation.

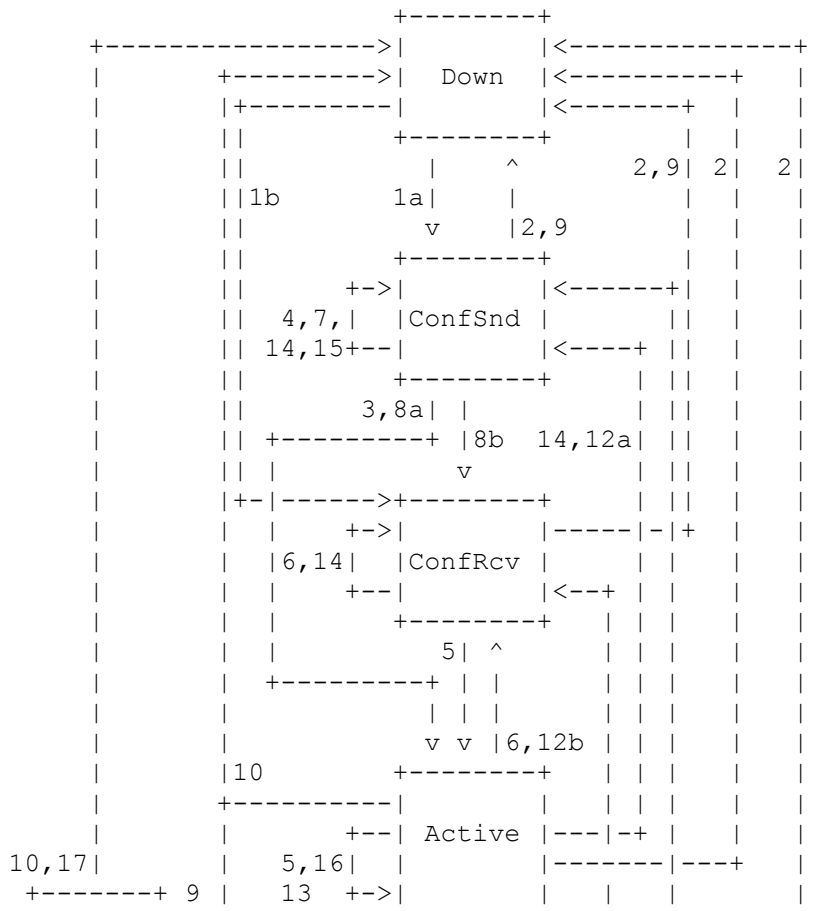
15. evConfRet : un temporisateur de retransmission est arrivé à expiration et un message Config est renvoyé.

16. evHelloRet : le temporisateur HelloInterval est arrivé à expiration et un paquet Hello est envoyé.

17. evDownTimer : un temporisateur est arrivé à expiration et aucun message n'a été reçu avec le fanion ControlChannelDown établi.

11.1.3 Description du FSM de canal de contrôle

La Figure 3 illustre le fonctionnement du FSM de canal de contrôle sous forme d'un diagramme de transition d'états de FSM.



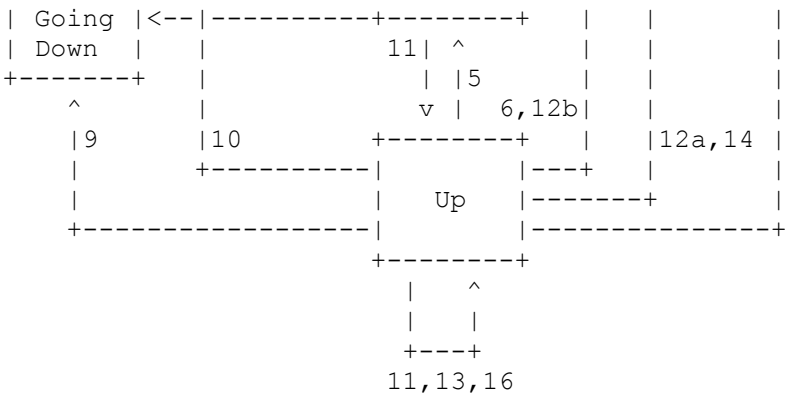


Figure 3 : FSM de canal de contrôle

L'événement evCCDn force toujours le FSM à l'état Down. Les événements evHoldTimer et evReconfig forcent toujours le FSM à l'état Négociation (ConfSnd ou ConfRcv).

11.2 FSM de liaison TE

Le FSM de liaison TE définit les états et la logique du fonctionnement de la liaison TE LMP.

11.2.1 États de liaison TE

Une liaison TE LMP peut être dans un des états décrits ci-dessous. Chaque état correspond à une certaine condition de la liaison TE et est généralement associé à un type spécifique de message LMP qui est périodiquement transmis à l'extrémité distante via le canal de contrôle associé ou dans la bande via les liaisons de données.

Down *(arrêté)* : Il n'y a pas de liaison de données allouée à la liaison TE.

Init : Des liaisons de données ont été allouées à la liaison TE, mais la configuration n'a pas encore été synchronisée avec le voisin LMP. Le message LinkSummary est périodiquement transmis au voisin LMP.

Up *(actif)* : C'est l'état normal de fonctionnement de la liaison TE. Au moins un canal de contrôle LMP doit être opérationnel entre les nœuds qui partagent la liaison TE. Au titre du fonctionnement normal, le message LinkSummary peut être périodiquement transmis au voisin LMP ou généré par une demande externe.

Dégradé : Dans cet état, tous les canaux de contrôle LMP sont arrêtés, mais la liaison TE comporte toujours des liaisons de données qui sont allouées au trafic d'utilisateur.

11.2.2 Événements de liaison TE

Le fonctionnement de la liaison TE LMP est décrit en termes d'états et d'événements de FSM. Les événements de liaison TE sont générés par les sous programmes de traitement de paquet et par les FSM du ou des canaux de contrôle associés et des liaisons de données. Chaque événement a son numéro et un nom symbolique. Les descriptions des événements possibles sont données ci-dessous.

1. evDCUp : un ou plusieurs canaux de données sont activés et alloués à la liaison TE.
2. evSumAck : un message LinkSummary est reçu et a un accusé de réception positif.
3. evSumNack : un message LinkSummary est reçu et a un accusé de réception négatif.
4. evRcvAck : un message LinkSummaryAck est reçu qui accuse réception de la configuration de la liaison TE.
5. evRcvNack : un message LinkSummaryNack est reçu.
6. evSumRet : le temporisateur de retransmission est arrivé à expiration et le message LinkSummary est renvoyé.
7. evCCUp : le premier canal de contrôle s'active.
8. evCCDown : le dernier canal de contrôle s'arrête.
9. evDCDown : le dernier canal de données de la liaison TE a été retiré.

11.2.3 Description du FSM de liaison TE

La Figure 4 illustre le fonctionnement du FSM de liaison TE LMP sous la forme d'un diagramme de transition d'états de FSM.

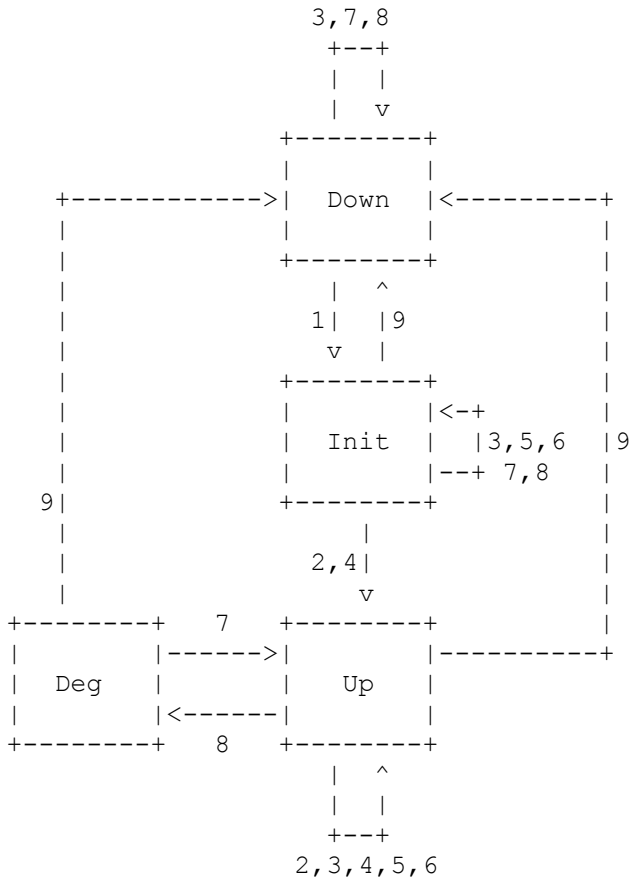


Figure 4 : FSM LMP de liaison TE

Dans le FSM ci-dessus, les sous états qui peuvent être mis en œuvre quand la procédure de vérification de liaison est utilisée ont été omis.

11.3. FSM de liaison de données

Le FSM de liaison de données définit les états et la logique du fonctionnement d'une liaison de données au sein d'une liaison TE LMP. Le fonctionnement d'une liaison de données est décrit en termes d'états et événements de FSM. Les liaisons de données peuvent être en mode actif (émission) où les messages Test sont transmis à partir d'elles, ou en mode passif (réception) où les messages Test sont reçus à travers elles. Pour la clarté de l'exposé, des FSM séparés sont définis pour les liaisons de données actives et passives ; cependant, un seul ensemble d'état et événements de liaison de données est défini.

11.3.1 États de liaison de données

Toute liaison de données peut être dans un des états décrits ci-dessous. Chaque état correspond à une certaine condition de la liaison de données.

Down : la liaison de données n'a pas été mise dans le réservoir de ressources (c'est-à-dire, la liaison n'est pas "en service")

Test : la liaison de données est en cours de vérification. Un message Test LMP est envoyé périodiquement sur la liaison.

PasvTest : la liaison de données est vérifiée pour les messages d'essai entrants.

Up/Free : la liaison a été vérifiée avec succès et est maintenant mise dans le réservoir de ressources (en service). La liaison n'a pas encore été allouée au trafic de données.

Up/Alloc : la liaison est active et a été allouée au trafic de données.

11.3.2 Événements de liaison de données

Les événements de liaisons sont générés par les sous programmes de traitement des paquets et par les FSM du canal de contrôle associé et de la liaison TE.

Chaque événement a son numéro et un nom symbolique. La description des événements possibles de liaison de données figure ci-dessous :

1. evCCUp: le premier canal de contrôle actif passe à l'état Up.

2. evCCDown : la connexité du voisin LMP est perdue. Cela indique que le dernier canal de contrôle LMP a une défaillance entre des nœuds voisins.

3. evStartTst : c'est un événement externe qui déclenche l'envoi de messages Test sur la liaison de données.

4. evStartPsv : c'est un événement externe qui déclenche l'écoute des messages Test sur la liaison de données.

5. evTestOK : La vérification de la liaison a réussi et la liaison peut être utilisée pour l'établissement du chemin.
   (a) Cet événement indique que la procédure de vérification de liaison (voir à la Section 5) a réussi pour cette liaison de données et un message TestStatusSuccess a été reçu sur le canal de contrôle.
   (b) Cet événement indique que la liaison est prête pour l'établissement de chemin, mais la procédure de vérification de liaison n'a pas été utilisée. Pour la signalisation dans la bande du canal de contrôle, l'établissement du canal de contrôle peut être suffisant pour vérifier la liaison.

6. evTestRcv : un message Test a été reçu sur l'accès de données et un message TestStatusSuccess est transmis sur le canal de contrôle.

7. evTestFail : la vérification de liaison a retourné un résultat négatif. Ce peut être parce que (a) un message TestStatusFailure a été reçu, ou (b) la procédure de vérification s'est terminée sans recevoir de message TestStatusSuccess ou TestStatusFailure pour la liaison de données.

8. evPsvTestFail : la vérification de liaison a retourné un résultat négatif. Cela indique qu'un message Test n'a pas été détecté et soit (a) le VerifyDeadInterval est arrivé à expiration, soit (b) la procédure de vérification s'est terminée et le VerifyDeadInterval n'est pas encore arrivé à expiration.

9. evLnkAlloc : la liaison de données a été allouée.

10. evLnkDealloc : la liaison de données a été désallouée.

11. evTestRet : un temporisateur de retransmission est arrivé à expiration et le message Test est renvoyé.

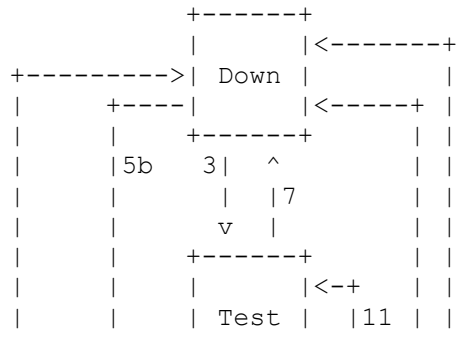
12. evSummaryFail : le LinkSummary ne correspond pas pour cet accès de données.

13. evLocalizeFail : une défaillance a été localisée pour cette liaison de données.

14. evdcDown : le canal de données n'est plus disponible.

11.3.3. Description du FSM de liaison de données active

La Figure 5 illustre le fonctionnement du FSM de liaison de données LMP active sous forme d'un diagramme de transition d'états de FSM.



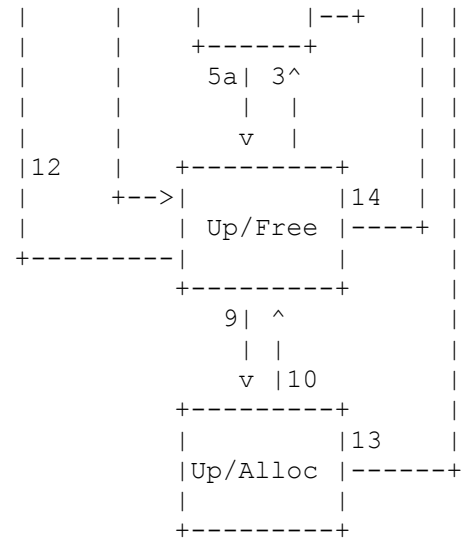


Figure 5 : FSM LMP de liaison de données active

11.3.4 Description du FSM de liaison de données passive

La Figure 6 illustre le fonctionnement du FSM de liaison de données LMP passive sous la forme d'un diagramme de transition d'états de FSM.

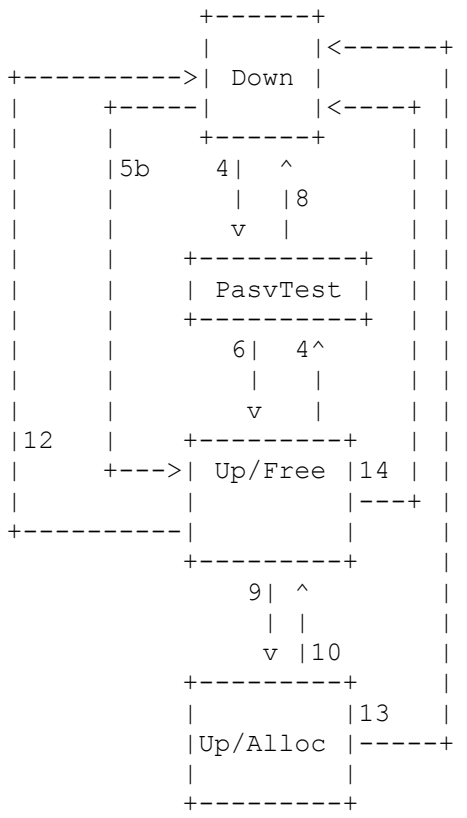


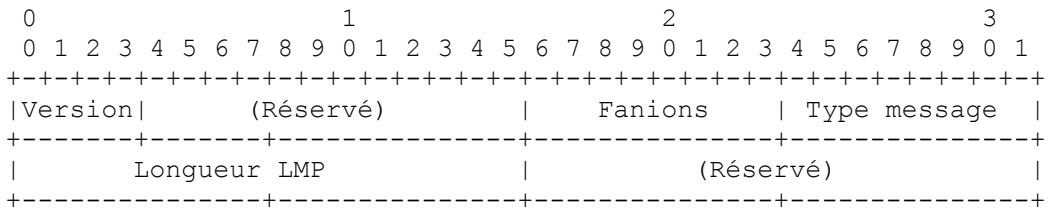
Figure 6 : FSM LMP de liaison de données passive

12. Formats de message LMP

Tous les messages LMP (sauf, dans certains cas, les messages Test, qui sont limités par le mécanisme de transport pour la messagerie dans la bande) fonctionnent sur UDP avec un numéro d'accès LMP de 701.

12.1 En-tête commun

En plus de l'en-tête UDP et de l'en-tête IP standard, tous les messages LMP (sauf, dans certains cas, les messages Test qui peuvent être limités par le mécanisme de transport pour la messagerie dans la bande) ont l'en-tête commun suivant :



Le champ Réservé devrait être envoyé à zéro et ignoré à réception.

Toutes les valeurs sont définies dans l'ordre des octets du réseau (c'est-à-dire l'ordre gros boutien des octets).

Version : 4 bits. Numéro de version du protocole. C'est la version 1.

Fanions : 8 bits. Les valeurs de bit suivantes sont définies. Tous les autres bits sont réservés et devraient être envoyés à zéro et ignorés à réception.

0x01 : ControlChannelDown *(canal de contrôle à l'arrêt)*
0x02 : LMP Restart *(redémarrage de LMP)*
Ce bit est établi pour indiquer qu'une défaillance de nœud s'est produite et que l'état de contrôle LMP a été perdu. Ce fanion peut être remis à 0 quand un message Hello est reçu avec RcvSeqNum égal au TxSeqNum local.

Type de message : 8 bits. Les valeurs suivantes sont définies. Toutes les autres valeurs sont réservées

1 = Config
2 = ConfigAck
3 = ConfigNack
4 = Hello
5 = BeginVerify
6 = BeginVerifyAck
7 = BeginVerifyNack
8 = EndVerify
9 = EndVerifyAck
10 = Test
11 = TestStatusSuccess
12 = TestStatusFailure
13 = TestStatusAck
14 = LinkSummary
15 = LinkSummaryAck
16 = LinkSummaryNack
17 = ChannelStatus
18 = ChannelStatusAck
19 = ChannelStatusRequest
20 = ChannelStatusResponse

Tous les messages sont envoyés sur le canal de contrôle SAUF le message Test, qui est envoyé sur la liaison de données objet des essais.

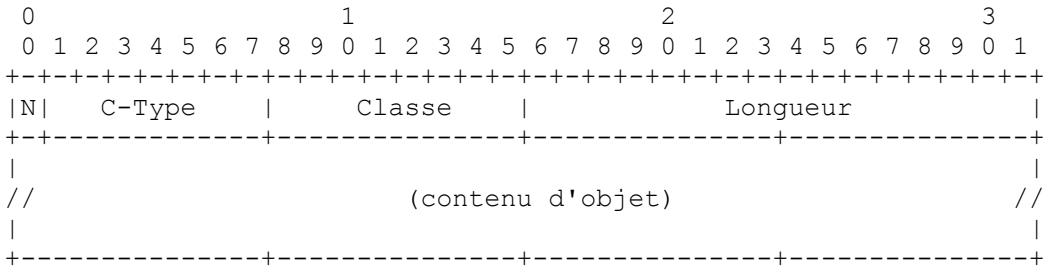
Longueur LMP : 16 bits. Longueur totale de ce message LMP en octets, incluant l'en-tête commun et tous les objets de longueur variable qui suivent.

12.2 Format d'objet LMP

Les messages LMP sont construits avec des objets. Chaque objet est identifié par sa classe d'objet et son type de classe. Chaque objet a un nom, qui est toujours en majuscules dans ce document. Les objets LMP peuvent être négociables ou non négociables (identifié par le bit N dans l'en-tête d'objet). Les objets négociables peuvent être utilisés pour permettre aux appareils de se mettre d'accord sur certaines valeurs. Les objets non négociables sont utilisés pour l'annonce de valeurs spécifiques qui n'ont pas besoin de, ou ne permettent pas, la négociation.

Toutes les valeurs sont définies dans l'ordre des octets du réseau (c'est-à-dire l'ordre gros boutien des octets).

Le format de l'objet LMP est le suivant :



N : 1 bit. Le fanion N indique si l'objet est négociable (N=1) ou non négociable (N=0).

C-Type : 7 bits. Type de classe, unique au sein d'une classe d'objet. Les valeurs sont définies à la Section 13.

Classe : 8 bits. La classe indique le type d'objet. Chaque objet a un nom, qui est toujours en majuscules dans ce document.

Longueur : 16 bits. Le champ Longueur indique la longueur de l'objet en octets, incluant les champs N, C-Type, Classe, et Longueur.

12.3 Messages de négociation de paramètres

12.3.1 Message Config (Type de message = 1)

Le message Config est utilisé dans la phase de négociation de canal de contrôle de LMP. Le contenu du message Config se construit avec des objets LMP. Le format du message Config est le suivant :

<Message Config> ::= <En-tête commun> <LOCAL_CCID> <MESSAGE_ID> <LOCAL_NODE_ID> <CONFIG>

L'ordre de transmission ci-dessus DEVRAIT être respecté.

L'objet MESSAGE_ID est dans la portée de l'objet LOCAL_CCID.

Le message Config DOIT être transmis périodiquement jusqu'à ce que (1) il reçoive un message ConfigAck ou ConfigNack, (2) une limite de répétition d'essais ait été atteinte sans qu'aucun message ConfigAck ou ConfigNack ait été reçu, ou (3) qu'il reçoive un message Config du nœud distant et ait perdu le concours (par exemple, le Nœud_Id du nœud distant est plus élevé que le Nœud_Id du nœud local). L'intervalle de retransmission et la limite de répétition d'essais sont des paramètres de configuration locaux.

12.3.2 Message ConfigAck (Type de message = 2)

Le message ConfigAck est utilisé pour accuser réception du message Config et indique l'accord sur tous les paramètres.

<Message ConfigAck> ::= <En-tête commun> <LOCAL_CCID> <LOCAL_NODE_ID> <REMOTE_CCID>
<MESSAGE_ID_ACK> <REMOTE_NODE_ID>

L'ordre de transmission ci-dessus DEVRAIT être respecté.

Le contenu des objets REMOTE_CCID, MESSAGE_ID_ACK, et REMOTE_NODE_ID DOIT être obtenu du message Config dont on accuse réception.

12.3.3 Message ConfigNack (Type de message = 3)

Le message ConfigNack est utilisé pour accuser réception du message Config et indiquer le désaccord sur les paramètres non négociables ou proposer d'autres valeurs pour les paramètres négociables. Les paramètres pour lesquels un accord a été conclu NE DOIVENT PAS être inclus dans le message ConfigNack. Le format du message ConfigNack est comme suit :

<Message ConfigNack> ::= <En-tête commun> <LOCAL_CCID> <LOCAL_NODE_ID> <REMOTE_CCID>
<MESSAGE_ID_ACK> <REMOTE_NODE_ID> <CONFIG>

L'ordre de transmission ci-dessus DEVRAIT être respecté.

Le contenu des objets REMOTE_CCID, MESSAGE_ID_ACK, et REMOTE_NODE_ID DOIT être obtenu du message Config qui reçoit un accusé de réception négatif

Il est possible que plusieurs paramètres soient invalides dans le message Config.

Si un objet CONFIG négociable est inclus dans le message ConfigNack, il DOIT inclure des valeurs acceptables pour les paramètres.

Si le message ConfigNack inclut des objets CONFIG pour des paramètres non négociables, ils DOIVENT être copiés des objets CONFIG reçus dans le message Config.

Si le message ConfigNack est reçu et n'inclut que des objets CONFIG qui sont négociables, un nouveau message Config DEVRAIT alors être envoyé. Les valeurs dans l'objet CONFIG du nouveau message Config DEVRAIENT tenir compte des valeurs acceptables incluses dans le message ConfigNack.

Si un nœud reçoit un message Config et reconnaît l'objet CONFIG, mais ne reconnaît pas le C-Type, un message ConfigNack incluant l'objet CONFIG inconnu DOIT être envoyé.

12.4 Message Hello (Type de message = 4)

Le format du message Hello est comme suit :

<Message Hello> ::= <En-tête commun> <LOCAL_CCID> <HELLO>

L'ordre de transmission ci-dessus DEVRAIT être respecté.

Le message Hello DOIT être émis périodiquement au moins une fois toutes les HelloInterval ms. Si aucun message Hello n'est reçu dans le HelloDeadInterval, le canal de contrôle est supposé défaillant.

12.5 Messages de vérification de liaison

12.5.1 Message BeginVerify (Type de message = 5)

Le message BeginVerify est envoyé sur le canal de contrôle et est utilisé pour initier le processus de vérification de liaison. Le format est le suivant :

<Message BeginVerify> ::= <En-tête commun> <LOCAL_LINK_ID> <MESSAGE_ID> [<REMOTE_LINK_ID>]
<BEGIN_VERIFY>

L'ordre de transmission ci-dessus DEVRAIT être respecté.

Pour limiter la portée d'une vérification de liaison à une liaison TE particulière, le champ Link_Id de l'objet LOCAL_LINK_ID DOIT être non zéro. Si ce champ est à zéro, les liaisons de données peuvent s'étendre sur plusieurs liaisons TE et/ou elles peuvent comprendre une liaison TE qui est déjà en cours de configuration. Dans le cas particulier où le champ local Link_Id est zéro, le fanion "Vérifier toutes les liaisons" de l'objet BEGIN_VERIFY est utilisé pour distinguer entre les liaisons de données qui s'étendent sur plusieurs liaisons TE et celles qui n'ont pas encore été allouées à une liaison TE (voir à la Section 5).

L'objet REMOTE_LINK_ID peut être inclus si la transposition de Link_Id local/distant est connue.

Le champ Link_Id de l'objet REMOTE_LINK_ID DOIT être non zéro si il est inclus.

Le message BeginVerify DOIT être émis périodiquement jusqu'à ce que (1) le nœud reçoive un message soit BeginVerifyAck, soit BeginVerifyNack, pour accepter ou rejeter le processus de vérification, ou (2) qu'une limite de répétition des essais ait été atteinte et qu'aucun message BeginVerifyAck ou BeginVerifyNack n'ait été reçu. L'intervalle de retransmission et la limite de répétition des essais sont des paramètres de configuration locaux.

12.5.2 Message BeginVerifyAck (Type de message = 6)

Lorsque un message BeginVerify est reçu et que des messages Test sont prêts à être traités, un message BeginVerifyAck DOIT être émis.

<Message BeginVerifyAck> ::= <En-tête commun> [<LOCAL_LINK_ID>] <MESSAGE_ID_ACK>
<BEGIN_VERIFY_ACK> <VERIFY_ID>

L'ordre de transmission ci-dessus DEVRAIT être respecté.

L'objet LOCAL_LINK_ID peut être inclus si la transposition de Link_Id local/distant est connue ou apprise par le message BeginVerify.

Le champ Link_Id du LOCAL_LINK_ID DOIT être non zéro si il est inclus.

Le contenu de l'objet MESSAGE_ID_ACK DOIT être obtenu du message BeginVerify dont on accuse réception.

L'objet VERIFY_ID contient une valeur unique pour le nœud qui est allouée par le générateur du message BeginVerifyAck. Cette valeur est utilisée pour identifier de façon univoque le processus de vérification parmi les voisins LMP et/ou les procédures parallèles d'essai entre les mêmes voisins LMP.

12.5.3 Message BeginVerifyNack (Type de message = 7)

Si un message BeginVerify est reçu et si un nœud ne veut pas ou est incapable de commencer la procédure de vérification, un message BeginVerifyNack DOIT être transmis.

<Message BeginVerifyNack> ::= <En-tête commun> [<LOCAL_LINK_ID>] <MESSAGE_ID_ACK>
<ERROR_CODE>

L'ordre de transmission ci-dessus DEVRAIT être respecté.

Le contenu de l'objet MESSAGE_ID_ACK DOIT être obtenu du message BeginVerify qui fait l'objet d'un accusé de réception négatif.

Si le processus de vérification n'est pas pris en charge, le ERROR_CODE DOIT indiquer "Procédure de vérification de liaison non acceptée".

Si la vérification est acceptée, mais si le nœud est incapable de commencer la procédure, l'objet ERROR_CODE DOIT indiquer "Ne veut pas vérifier". Si un message BeginVerifyNack est reçu avec un tel code d'erreur, le nœud qui a généré le BeginVerify DEVRAIT programmer une retransmission du BeginVerify après Rf secondes, où Rf est un paramètre défini en local.

Si mécanisme de transport de vérification n'est pas pris en charge, le code d'erreur DOIT indiquer "Mécanisme de transport de vérification non pris en charge".
Si la configuration distante de l'identifiant de liaison n'est pas accepté et si le contenu de l'objet REMOTE_LINK_ID (inclus dans le message BeginVerify) ne correspond à aucune valeur configurée , le code d'erreur DOIT indiquer "Erreur de configuration de l'identifiant de liaison".

Si un nœud reçoit un message BeginVerify et si il reconnaît l'objet BEGIN_VERIFY mais ne reconnaît pas le C-Type, le code d'erreur DOIT indiquer "Objet C-Type inconnu".

12.5.4 Message EndVerify (Type de message = 8)

Le message EndVerify est envoyé sur le canal de contrôle et est utilisé pour terminer le processus de vérification de liaison. Le message EndVerify peut être envoyé à tout moment lorsque le nœud initiateur désire mettre fin à la procédure de vérification. Le format est le suivant :

<Message EndVerify> ::=<En-tête commun> <MESSAGE_ID> <VERIFY_ID>

L'ordre de transmission ci-dessus DEVRAIT être respecté.
Le message EndVerify va être émis périodiquement jusqu'à ce que (1) un message EndVerifyAck ait été reçu, ou (2) une limite de répétition des essais ait été atteinte et qu'aucun message EndVerifyAck n'ait été reçu. L'intervalle de retransmission et la limite de répétition des essais sont des paramètres de configuration locaux.

12.5.5 Message EndVerifyAck (Type de message =9)

Le message EndVerifyAck est envoyé sur le canal de contrôle et est utilisé pour accuser réception de la terminaison du processus de vérification de liaison. Le format est le suivant :

<Message EndVerifyAck> ::= <En-tête commun> <MESSAGE_ID_ACK> <VERIFY_ID>

L'ordre de transmission ci-dessus DEVRAIT être respecté.

Le contenu de l'objet MESSAGE_ID_ACK DOIT être obtenu du message EndVerify dont on accuse réception.

12.5.6 Message Test (Type de message = 10)

Le message Test est transmis sur la liaison de données et est utilisé pour vérifier sa connexité physique. Sauf mention contraire explicite, ces messages DOIVENT être transmis sur UDP comme tous les autres messages LMP. Le format des messages Test est comme suit :

<Message Test> ::= <En-tête commun> <LOCAL_INTERFACE_ID> <VERIFY_ID>

L'ordre de transmission ci-dessus DEVRAIT être respecté.

Noter que ce message est envoyé sur une liaison de données et NON sur le canal de contrôle. Le mécanisme de transport pour le message Test est négocié en utilisant le champ Mécanisme de transport de Verify de l'objet BEGIN_VERIFY et le champ Réponse de transport de Verify de l'objet BEGIN_VERIFY_ACK (voir aux paragraphes 13.8 et 13.9).

Le nœud local (émetteur) envoie périodiquement un message Test (au moins une fois toutes les VerifyInterval ms) sur la liaison de données correspondante jusqu'à ce que (1) il reçoive un message TestStatusSuccess ou TestStatusFailure corrélatif sur le canal de contrôle de la part du nœud distant (receveur) ou (2) tous les canaux de contrôle actifs entre les deux nœuds soient défaillants. Le nœud distant va envoyer périodiquement un message TestStatus sur le canal de contrôle jusqu'à ce qu'il reçoive, soit un message TestStatusAck corrélatif, soit un message EndVerify.

12.5.7 Message TestStatusSuccess (Type de message = 11)

Le message TestStatusSuccess est transmis sur le canal de contrôle et est utilisé pour transmettre la transposition entre l'Interface_Id local et l'Interface_Id qui a été reçu dans le message Test.

<Message TestStatusSuccess> ::= <En-tête commun> <LOCAL_LINK_ID> <MESSAGE_ID>
<LOCAL_INTERFACE_ID> <REMOTE_INTERFACE_ID> <VERIFY_ID>

L'ordre de transmission ci-dessus DEVRAIT être respecté.

Le contenu de l'objet REMOTE_INTERFACE_ID DOIT être obtenu du message Test correspondant dont il est positivement accusé réception.

12.5.8 Message TestStatusFailure (Type de message = 12)

Le message TestStatusFailure est transmis sur le canal de contrôle et est utilisé pour indiquer que le message Test n'a pas été reçu.

<Message TestStatusFailure> ::= <En-tête commun> <MESSAGE_ID> <VERIFY_ID>

L'ordre de transmission ci-dessus DEVRAIT être respecté.

12.5.9 Message TestStatusAck (Type de message = 13)

Le message TestStatusAck est utilisé pour accuser réception du message TestStatusSuccess ou TestStatusFailure.

<Message TestStatusAck> ::= <En-tête commun> <MESSAGE_ID_ACK> <VERIFY_ID>

L'ordre de transmission ci-dessus DEVRAIT être respecté.
Le contenu de l'objet MESSAGE_ID_ACK DOIT être obtenu du message TestStatusSuccess ou TestStatusFailure dont on accuse réception.

12.6 Messages de résumé de liaison

12.6.1 Message LinkSummary (Type de message = 14)

Le message LinkSummary est utilisé pour synchroniser les identifiants d'interface et corréler les propriétés de la liaison TE. Le format du message LinkSummary est comme suit :

```
<Message LinkSummary> ::= <En-tête commun> <MESSAGE_ID> <TE_LINK> <DATA_LINK>
                          [<DATA_LINK>...]
```

L'ordre de transmission ci-dessus DEVRAIT être respecté.

Le message LinkSummary peut être échangé à tout moment sauf si une liaison est engagée dans le processus de vérification. Le message LinkSummary DOIT être émis périodiquement jusqu'à ce que (1) le nœud reçoive un message LinkSummaryAck ou LinkSummaryNack, ou (2) une limite de répétition des essais ait été atteinte sans qu'aucun message LinkSummaryAck ou LinkSummaryNack ait été reçu. L'intervalle de retransmission et la limite de répétition des essais sont tous deux des paramètres de configuration locaux.

12.6.2 Message LinkSummaryAck (Type de message = 15)

Le message LinkSummaryAck est utilisé pour indiquer l'accord sur la synchronisation d'identifiant d'interface et sur l'acceptation/accord sur tous les paramètres de la liaison. C'est à la réception de ce message que le nœud local constitue les associations d'identifiant de liaison.

```
<Message LinkSummaryAck> ::=  <En-tête commun> <MESSAGE_ID_ACK>
```

L'ordre de transmission ci-dessus DEVRAIT être respecté.

12.6.3 Message LinkSummaryNack (Type de message = 16)

Le message LinkSummaryNack est utilisé pour indiquer le désaccord sur les paramètres non négociés ou proposer d'autres valeurs pour les paramètres négociables. Les paramètres sur lesquels l'accord s'est fait NE DOIVENT PAS être inclus dans le message LinkSummaryNack.

```
<Message LinkSummaryNack> ::= <En-tête commun> <MESSAGE_ID_ACK> <ERROR_CODE>
                              [<DATA_LINK>...]
```

L'ordre de transmission ci-dessus DEVRAIT être respecté.

Les objets DATA_LINK DOIVENT inclure des valeurs acceptables pour tous les paramètres négociables. Si le LinkSummaryNack inclut des objets DATA_LINK pour des paramètres non négociables, ils DOIVENT être copiés des objets DATA_LINK reçus dans le message LinkSummary.

Si le message LinkSummaryNack est reçu et inclut seulement des paramètres négociables, un nouveau message LinkSummary DEVRAIT être envoyé. Les valeurs reçues dans le nouveau message LinkSummary DEVRAIENT tenir compte des paramètres acceptables inclus dans le message LinkSummaryNack.

Si le message LinkSummary est reçu avec des paramètres non négociables non acceptables, le code d'erreur DOIT indiquer "Paramètres LINK_SUMMARY non négociables inacceptables".

Si le message LinkSummary est reçu avec des paramètres négociables inacceptables, le code d'erreur DOIT indiquer "Renégocier les paramètres de LINK_SUMMARY".

Si le message LinkSummary est reçu avec un objet TE_LINK invalide, le code d'erreur DOIT indiquer "Objet TE_LINK invalide".

Si le message LinkSummary est reçu avec un objet DATA_LINK invalide, le code d'erreur DOIT indiquer "Objet DATA_LINK invalide".

Si le message LinkSummary est reçu avec un objet TE_LINK mais si le C-Type est inconnu, le code d'erreur DOIT

indiquer, "C-Type d'objet TE_LINK inconnu".

Si le message LinkSummary est reçu avec un objet DATA_LINK mais si le C-Type est inconnu, le code d'erreur DOIT indiquer, "C-Type d'objet DATA_LINK inconnu".

12.7 Messages de gestion des fautes

12.7.1 Message ChannelStatus (Type de message = 17)

Le message ChannelStatus est envoyé sur le canal de contrôle et est utilisé pour notifier à un voisin LMP l'état d'une liaison de données. Un nœud qui reçoit un message ChannelStatus DOIT répondre par un message ChannelStatusAck. Le format est le suivant :

<Message ChannelStatus> ::= <En-tête commun> <LOCAL_LINK_ID> <MESSAGE_ID> <CHANNEL_STATUS>

L'ordre de transmission ci-dessus DEVRAIT être respecté.

Si l'objet CHANNEL_STATUS ne comporte aucun identifiant d'interface, cela indique la défaillance de la liaison TE entière.

12.7.2 Message ChannelStatusAck (Type de message = 18)

Le message ChannelStatusAck est utilisé pour accuser réception du message ChannelStatus. Le format est le suivant :

<Message ChannelStatusAck> ::= <En-tête commun> <MESSAGE_ID_ACK>

L'ordre de transmission ci-dessus DEVRAIT être respecté.

Le contenu de l'objet MESSAGE_ID_ACK DOIT être obtenu du message ChannelStatus dont on accuse réception.

12.7.3 Message ChannelStatusRequest (Type de message = 19)

Le message ChannelStatusRequest est envoyé sur le canal de contrôle et est utilisé pour demander l'état d'une ou plusieurs liaisons de données. Un nœud qui reçoit un message ChannelStatusRequest DOIT répondre par un message ChannelStatusResponse. Le format est le suivant :

<Message ChannelStatusRequest> ::= <En-tête commun> <LOCAL_LINK_ID> <MESSAGE_ID>
[<CHANNEL_STATUS_REQUEST>]

L'ordre de transmission ci-dessus DEVRAIT être respecté.

Si l'objet CHANNEL_STATUS_REQUEST n'est pas inclus, la ChannelStatusRequest est alors utilisée pour demander l'état de TOUTES les liaisons de données de la liaison TE.

12.7.4 Message ChannelStatusResponse (Type de message = 20)

Le message ChannelStatusResponse est utilisé pour accuser réception du message ChannelStatusRequest et notifier au voisin LMP l'état de la ou des liaisons de données. Le format est le suivant :

<Message ChannelStatusResponse> ::= <En-tête commun> <MESSAGE_ID_ACK> <CHANNEL_STATUS>

L'ordre de transmission ci-dessus DEVRAIT être respecté.

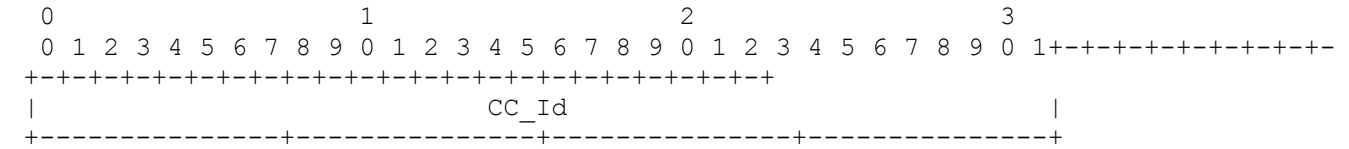
Le contenu des objets MESSAGE_ID_ACK DOIT être obtenu du message ChannelStatusRequest dont on accuse réception.

13. Définitions d'objet LMP

13.1 Classe d'identifiant de canal de contrôle (CCID, *Control Channel ID*)

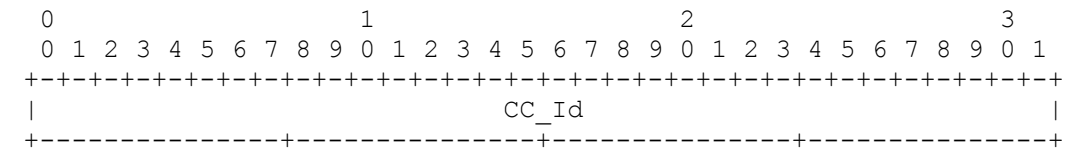
Classe = 1

C-Type = 1, LOCAL_CCID



CC_Id : 32 bits. Ceci DOIT être unique à l'échelle du nœud et non zéro. Le CC_Id identifie le canal de contrôle de l'envoyeur associé au message. Cet objet n'est pas négociable.

C-Type = 2, REMOTE_CCID *(CCID distant)*

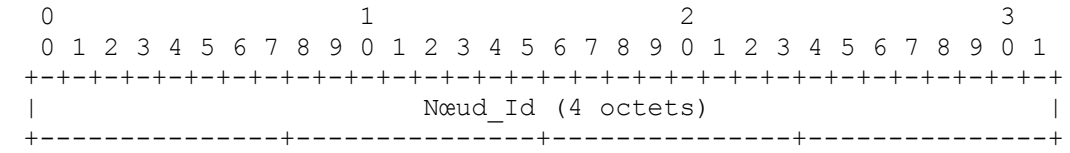


CC_Id : 32 bits. Ceci identifie le l'identifiant de canal de contrôle (CC_Id) du nœud distant et DOIT être non zéro. Cet objet n'est pas négociable.

13.2 Classe NODE_ID

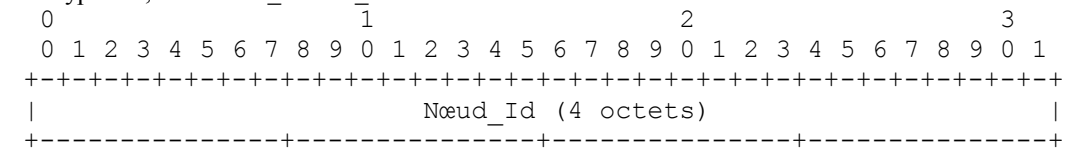
Classe= 2

C-Type = 1, LOCAL_NODE_ID



Nœud_Id : Ceci identifie le nœud qui a généré le paquet LMP. Cet objet n'est pas négociable.

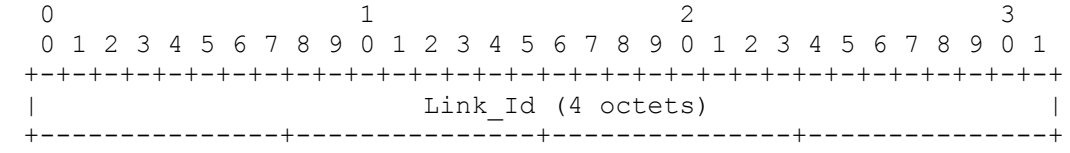
C-Type = 2, REMOTE_NODE_ID



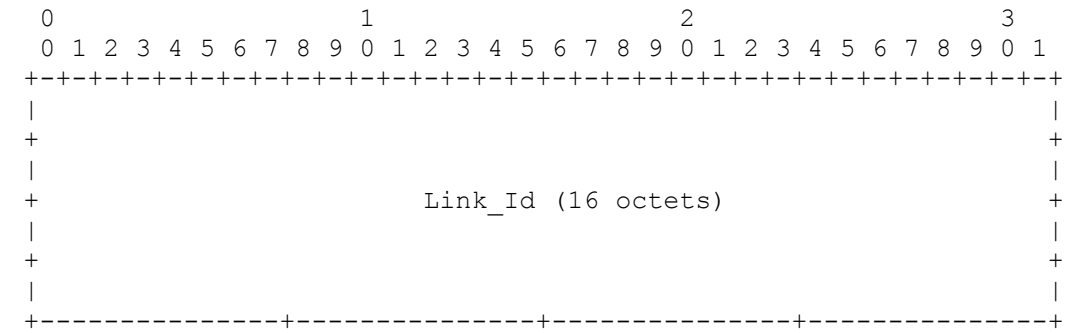
Nœud_Id : Ceci identifie le nœud distant. Cet objet n'est pas négociable.

13.3 Classe LINK_ID

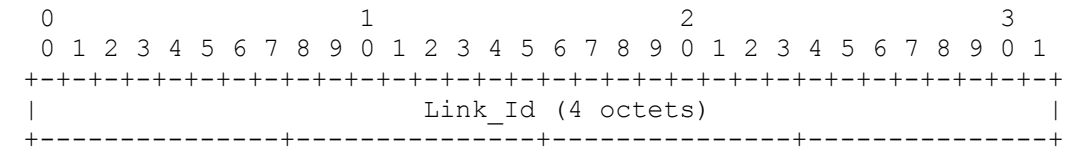
Classe = 3
C-Type = 1, LOCAL_LINK_ID IPv4
C-Type = 2, REMOTE_LINK_ID IPv4



o C-Type = 3, LOCAL_LINK_ID IPv6
o C-Type = 4, REMOTE_LINK_ID IPv6



o C-Type = 5, LOCAL_LINK_ID non numéroté
o C-Type = 6, REMOTE_LINK_ID non numéroté

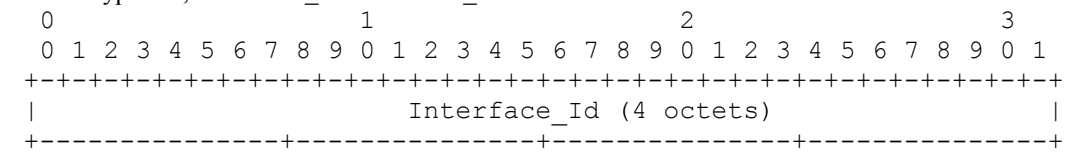


Link_Id : pour LOCAL_LINK_ID, cela identifie la liaison de l'envoyeur associée au message. Cette valeur DOIT être non zéro. Pour REMOTE_LINK_ID, cela identifie le Link_Id du nœud distant et DOIT être non zéro. Cet objet n'est pas négociable.

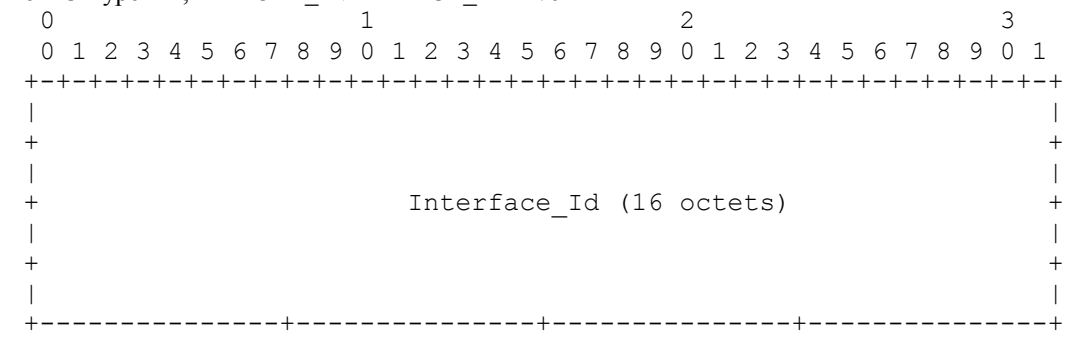
13.4 Classe INTERFACE_ID

Classe = 4

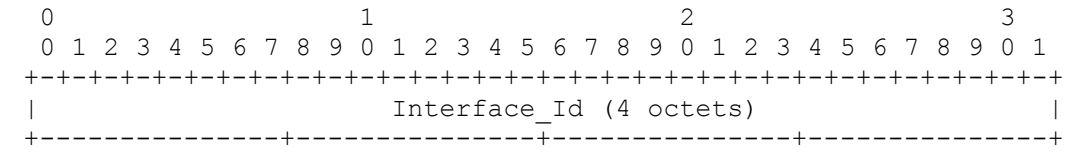
o C-Type = 1, LOCAL_INTERFACE_ID IPv4
o C-Type = 2, REMOTE_INTERFACE_ID IPv4



o C-Type = 3, LOCAL_INTERFACE_ID IPv6
o C-Type = 4, REMOTE_INTERFACE_ID IPv6



o C-Type = 5, LOCAL_INTERFACE_ID non numéroté
o C-Type = 6, REMOTE_INTERFACE_ID non numéroté

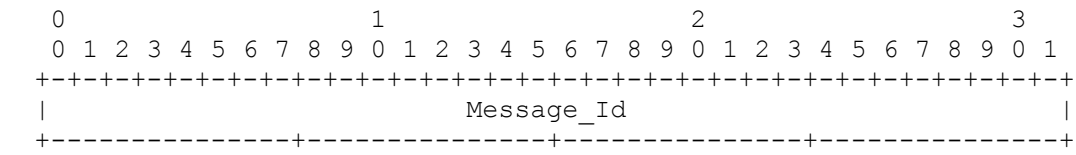


Interface_Id : pour le LOCAL_INTERFACE_ID, cela identifie la liaison de données. Cette valeur DOIT être unique pour le nœud et non zéro. Pour le REMOTE_INTERFACE_ID, cela identifie la liaison de données du nœud distant. L'Interface_Id DOIT être non zéro. Cet objet n'est pas négociable.

13.5 Classe MESSAGE_ID

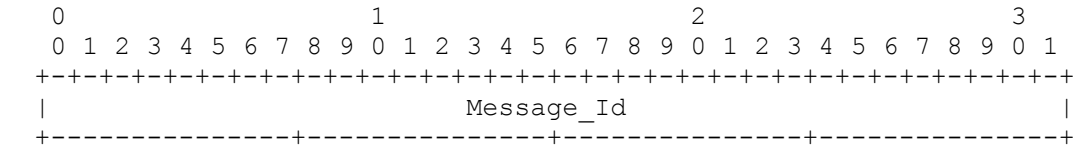
Classe = 5

o C-Type=1, Message_Id



Message_Id : le champ Message_Id est utilisé pour identifier un message. Cette valeur est incrémentée et ne diminue que quand la valeur revient à zéro. Cela est utilisé pour les accusés de réception de message. Cet objet n'est pas négociable.

o C-Type = 2, MessageIdAck

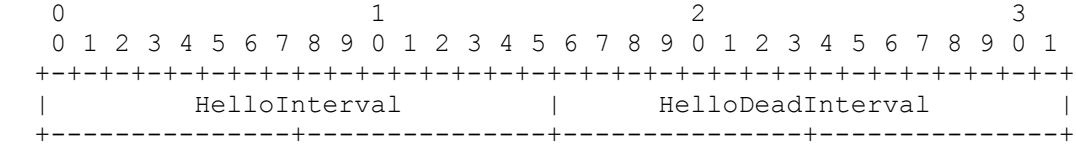


Message_Id : le champ Message_Id est utilisé pour identifier le message dont on accuse réception. Cette valeur est copiée de l'objet MESSAGE_ID du message dont on accuse réception. Cet objet n'est pas négociable.

13.6 Classe CONFIG

Classe = 6.

o C-Type = 1, HelloConfig



HelloInterval : 16 bits. Indique la fréquence d'envoi des paquets Hello et est mesuré en millisecondes (ms).

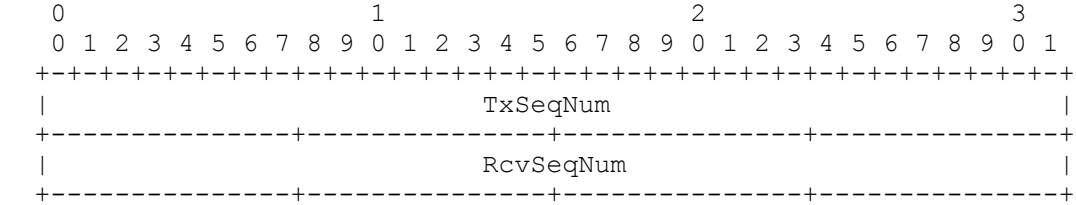
HelloDeadInterval : 16 bits. Si aucun paquet Hello n'est reçu dans le HelloDeadInterval, le canal de contrôle est supposé être défaillant. Le HelloDeadInterval est mesuré en millisecondes (ms). Le HelloDeadInterval DOIT être supérieur au HelloInterval, et DEVRAIT être d'au moins 3 fois la valeur de HelloInterval.

Si le mécanisme de garde en vie rapide de LMP n'est pas utilisé, le HelloInterval et le HelloDeadInterval DOIVENT être réglés à zéro.

13.7 Classe HELLO

Classe = 7

o C-Type = 1, Hello



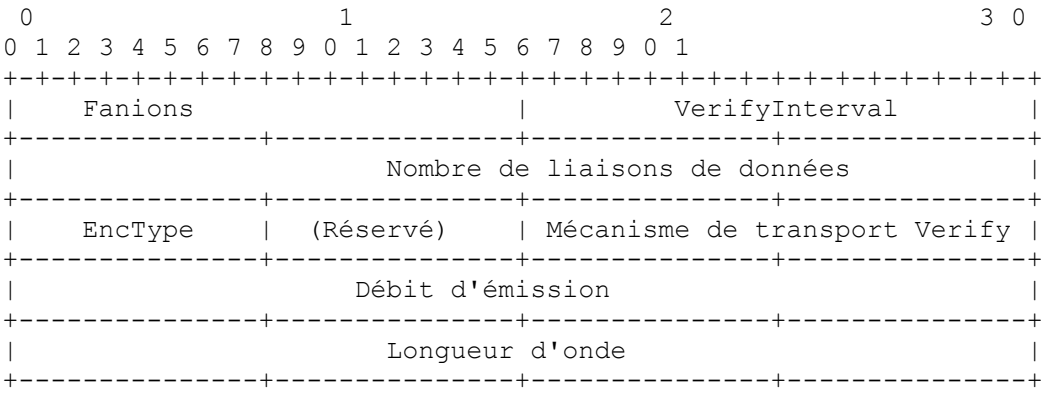
TxSeqNum : 32 bits. c'est le numéro de séquence actuel pour ce message Hello. Ce numéro de séquence sera incrémenté lorsque le numéro de séquence est reflété dans le RcvSeqNum d'un paquet Hello qui est reçu sur le canal de contrôle. TxSeqNum=0 n'est pas permis. TxSeqNum=1 est utilisé pour indiquer que c'est le premier message Hello envoyé sur le canal de contrôle.

RcvSeqNum : 32 bits. C'est le numéro de séquence du dernier message Hello reçu sur le canal de contrôle. RcvSeqNum=0 est utilisé pour indiquer qu'un message Hello n'a pas encore été reçu. Cet objet n'est pas négociable.

13.8 Classe BEGIN_VERIFY

Classe = 8

o C-Type = 1



Le champ Réservé devrait être envoyé à zéro et ignoré à réception.

Fanions : 16 bits. Les fanions suivants sont définis :
0x0001 : Vérifier toutes les liaisons. Si ce bit est établi, le processus de vérification vérifie toutes les liaisons non allouées ; autrement il vérifie seulement les nouveaux accès ou les liaisons composantes qui sont à ajouter à cette liaison TE.
0x0002 : Type de liaison de données. Si il est établi, les liaisons de données à vérifier sont des accès, autrement ce sont des liaisons composantes

VerifyInterval : 16 bits. C'est l'intervalle entre les messages Test successifs et il est mesuré en millisecondes (ms).

Nombre de liaisons de données : 32 bits. C'est le nombre de liaisons de données qui seront vérifiées.

EncType : 8 bits. C'est le type de codage de la liaison de données. Les valeurs de EncType définies sont cohérentes avec les valeurs de type de codage de LSP de la [RFC3471].

Mécanisme de transport de Verify : 16 bits. Cela définit le mécanisme de transport pour les messages Test. La portée de ce gabarit binaire se restreint à chaque type de codage. Le nœud local va régler les bits correspondants aux divers mécanismes qu'il peut accepter pour transmettre les messages d'essai LMP. Le receveur choisit le mécanisme approprié dans le message BeginVerifyAck. Le fanion qui suit est défini pour tous les types de codage. Tous les autres fanions dépendent du type de codage.
0x8000 : Charge utile, le message Test transmis dans la charge utile. Il est capable de transmettre des messages Test dans la charge utile. Le message Test est envoyé comme paquet IP comme défini plus haut.

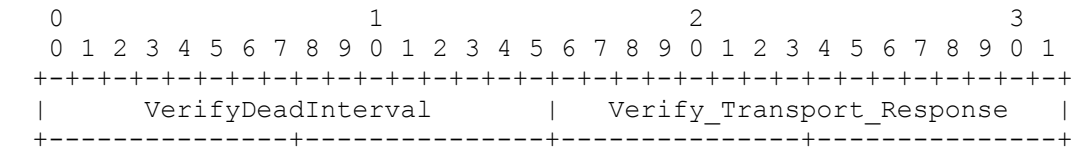
Débit d'émission : 32 bits. C'est le débit d'émission de la liaison de données sur laquelle les messages Test vont être transmis. Ceci est exprimé en octets par seconde et représenté en format IEEE à virgule flottante.

Longueur d'onde : 32 bits. Lorsque une liaison de données est allouée à un accès ou à une liaison composante qui est capable de transmettre de multiples longueurs d'onde (par exemple, une fibre ou un accès à capacité de bande d'ondes) il est essentiel de savoir sur quelle longueur d'onde les messages d'essais seront transmis. Cette valeur correspond à la longueur d'onde sur laquelle les messages Test seront transmis et a une signification locale. Si il n'y a pas d'ambiguïté quant à la longueur d'onde sur laquelle le message sera envoyé, cette valeur DEVRAIT être réglée à 0.

13.9 Classe BEGIN_VERIFY_ACK

Classe = 9

o C-Type = 1

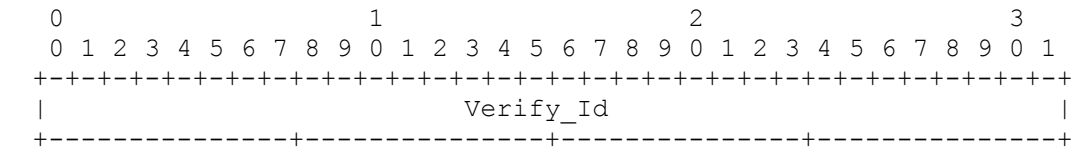


VerifyDeadInterval : 16 bits. Si un message Test n'est pas détecté dans le VerifyDeadInterval, un nœud enverra le message TestStatusFailure pour cette liaison de données.

Verify_Transport_Response : 16 bits. Le receveur du message BeginVerify (et le futur receveur des messages Test) choisit le mécanisme de transport parmi les divers types qui sont offert par l'émetteur des messages Test. Un bit, et un seul, DOIT être établi dans la réponse de transport de vérification. Cet objet n'est pas négociable.

13.10. Classe VERIFY_ID

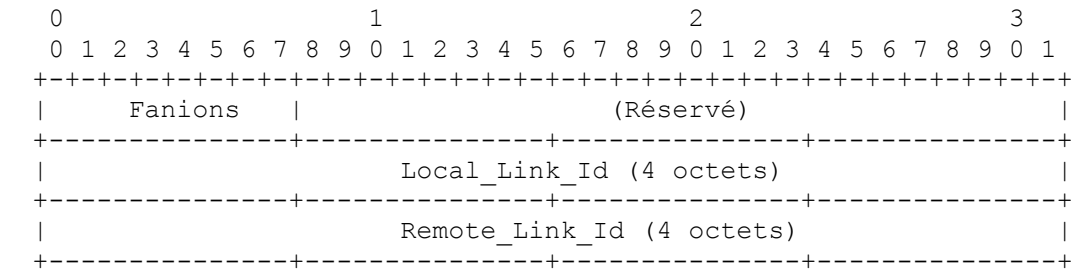
Classe = 10
o C-Type = 1



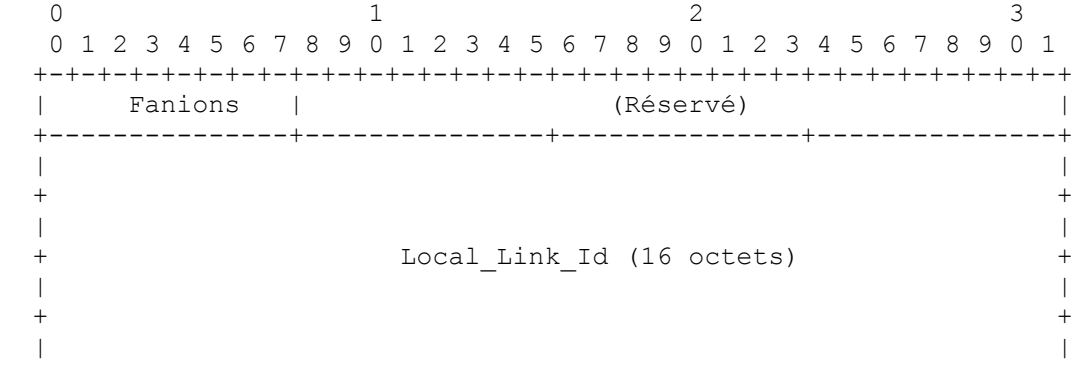
Verify_Id : 32 bits. Ceci est utilisé pour différencier les messages Test provenant des différentes liaisons TE et/ou homologues LMP. C'est une valeur unique pour le nœud qui est allouée par le receveur du message BeginVerify. Cet objet n'est pas négociable.

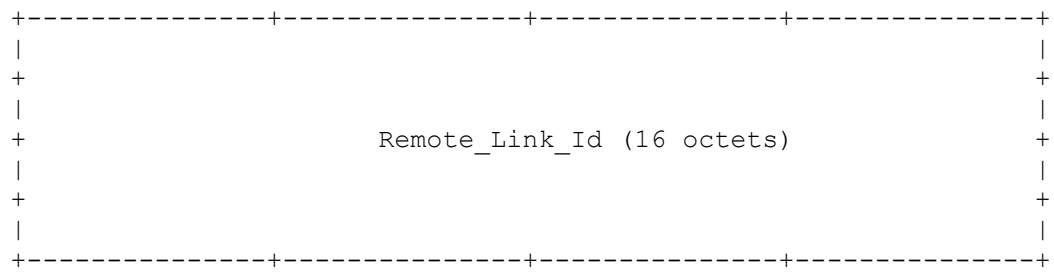
13.11 Classe TE_LINK

Classe = 11
o C-Type = 1, TE_LINK IPv4

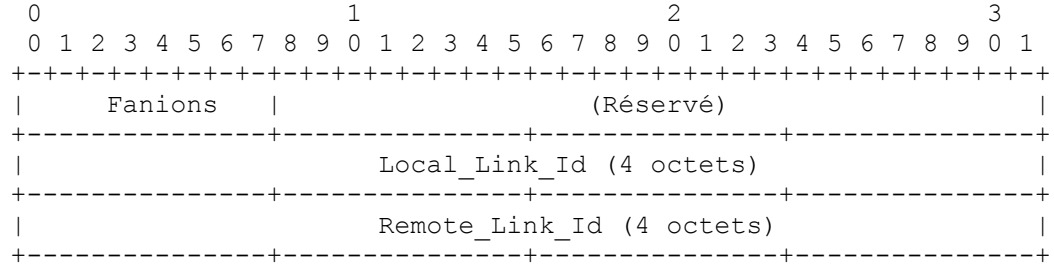


o C-Type = 2, TE_LINK IPv6





o C-Type = 3, TE_LINK non numérotée



Le champ Réservé devrait être envoyé à zéro et ignoré à réception.

Fanions : 8 bits. Les fanions suivants sont définis. Toutes les autres valeurs binaires sont réservées et devraient être envoyées à zéro et ignorées à réception.

0x01 : Gestion de faute prise en charge.

0x02 : Vérification de liaison prise en charge.

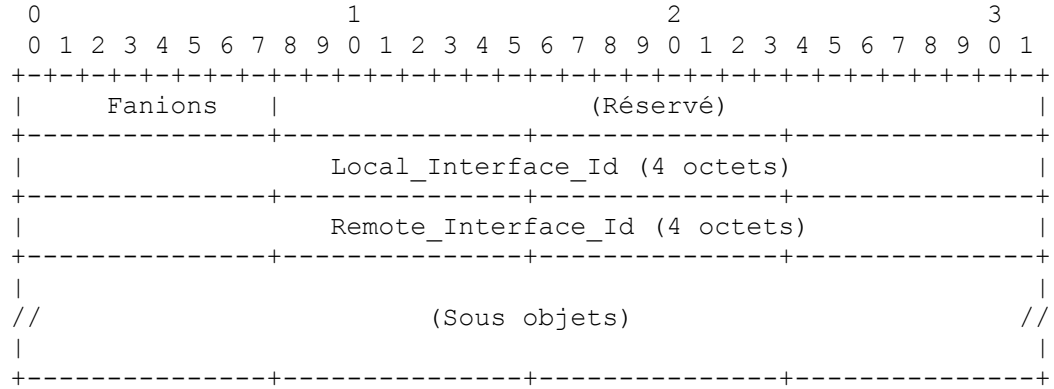
Local_Link_Id : Cela identifie l'identifiant de liaison local du nœud et DOIT être non zéro.

Remote_Link_Id : Cela identifie l'identifiant de liaison distant du nœud et DOIT être non zéro.

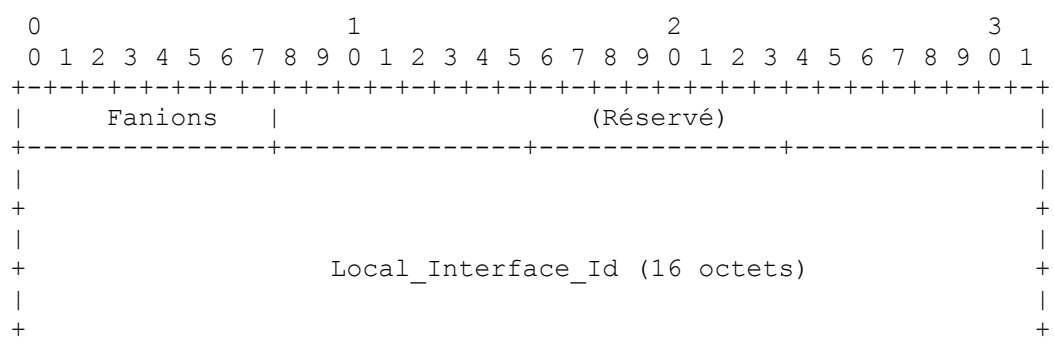
13.12 Classe DATA_LINK

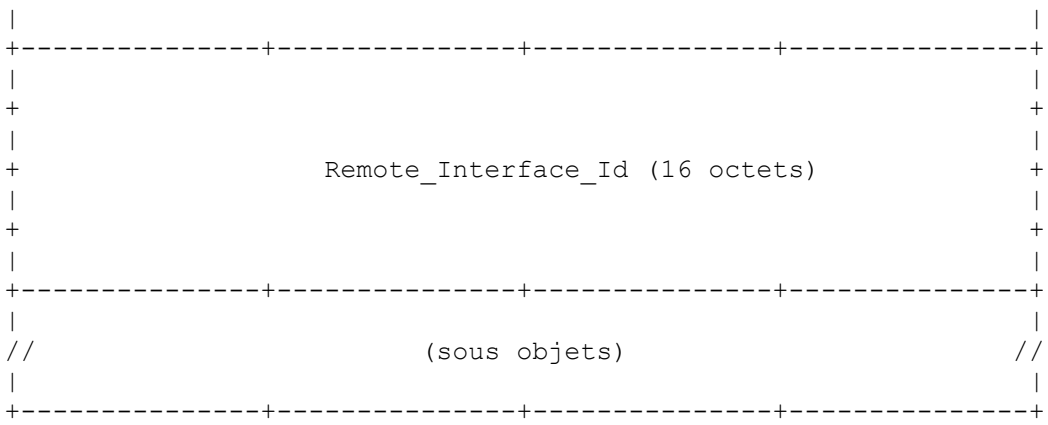
Classe = 12

o C-Type = 1, DATA_LINK IPv4

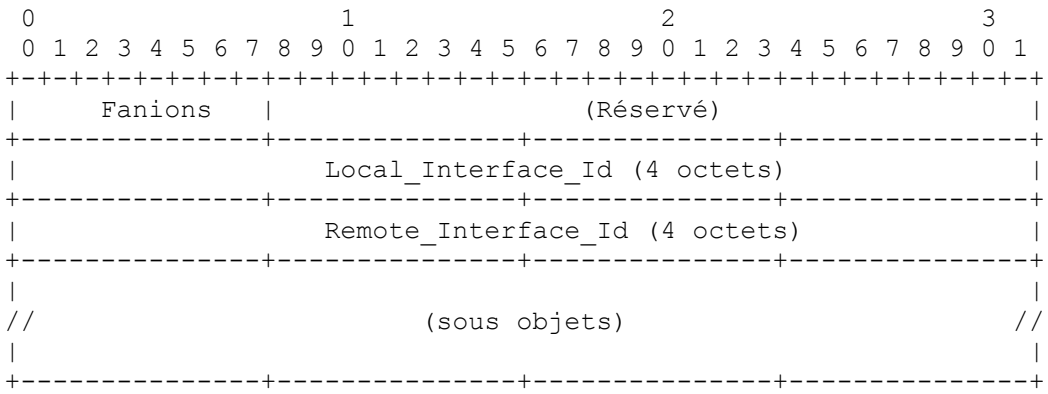


o C-Type = 2, DATA_LINK IPv6





o C-Type = 3, DATA_LINK non numérotée



Le champ Réservé devrait être envoyé à zéro et ignoré à réception.

Fanions : 8 bits. Les fanions suivants sont définis. Toutes les autres valeurs de bits sont réservées et devraient être envoyées à zéro et ignorées à réception.

0x01 : Type d'interface. Si il est établi, la liaison de données est un accès, autrement, c'est une liaison composante.

0x02 : Liaison allouée. S'il est établi, la liaison de données est actuellement allouée pour du trafic d'utilisateur. Si un seul identifiant d'interface est utilisé pour les liaisons de données d'émission et de réception, ce bit ne s'applique alors qu'à l'interface d'émission.

0x04 : Liaison défaillante. Si il est établi, la liaison de données est défaillante et ne convient pas pour du trafic d'utilisateur.

Local_Interface_Id : C'est l'identifiant local de la liaison de données. Il DOIT être unique pour le nœud et non zéro.

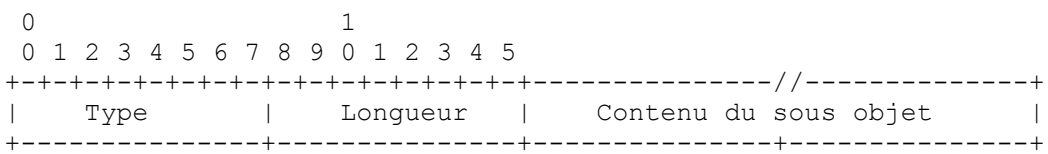
Remote_Interface_Id : C'est l'identifiant distant de la liaison de données. Il DOIT être non zéro.

Sous objets : Le contenu de l'objet DATA_LINK consiste en une série d'éléments de données de longueur variable appelés des sous objets. Les sous objets sont définis au paragraphe13.12.1.

Un objet DATA_LINK peut contenir plus d'un sous objet. Plus d'un sous objet du même type peut apparaître si plusieurs capacités sont prises en charge sur la liaison de données.

13.12.1 Sous objets de liaison de données

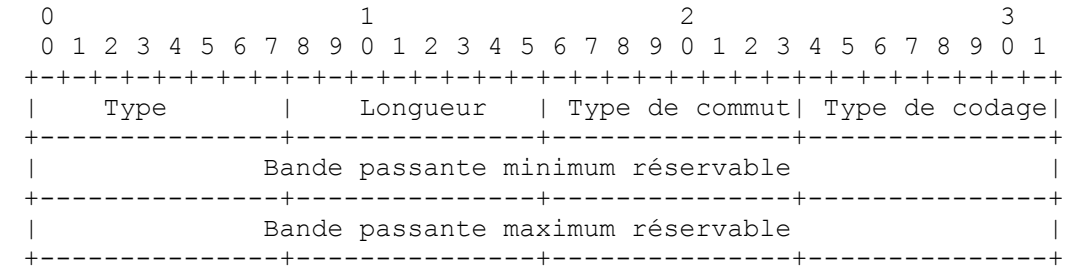
Le contenu de l'objet DATA_LINK inclut une série d'éléments de données de longueur variable appelés sous objets. Chaque sous objet est de la forme :



Type : 8 bits. Le type indique le type de contenu du sous objet. Les valeurs actuellement définies sont :
Type = 1, type d'interface de commutation
Type = 2, longueur d'onde

Longueur : 8 bits. Le champ Longueur contient la longueur totale du sous objet en octets, incluant les champs Type et Longueur. La longueur DOIT être au moins 4, et DOIT être un multiple de 4.

13.12.1.1 Sous objet de type 1 : type d'interface de commutation



Type de commut : 8 bits. C'est utilisé pour identifier le type de commutation de l'interface locale de la liaison TE comme défini dans la [RFC3471].

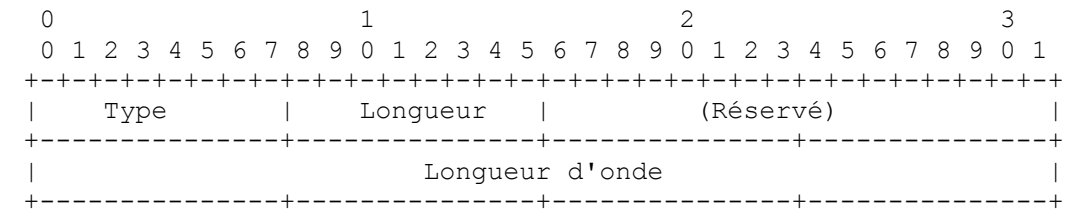
Type de codage : 8 bits. C'est le type de codage de la liaison de données. Les valeurs définies de type de codage sont cohérentes avec les valeurs de type de codage de LSP de la [RFC3471].

Bande passante minimum réservable : 32 bits. Elle est mesurée en octets par seconde et représentée dans le format IEEE à virgule flottante.

Bande passante maximum réservable : 32 bits. Elle est mesurée en octets par seconde et représentée dans le format IEEE à virgule flottante.

Si l'interface ne prend en charge qu'un débit fixe, les champs Bande passante minium et maximum sont réglés à la même valeur.

13.12.1.2 Sous objet de type 2 : longueur d'onde



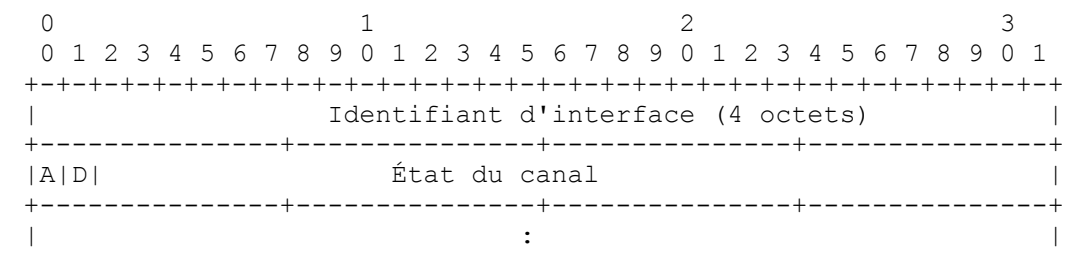
Le champ Réservé devrait être envoyé à zéro et ignoré à réception.

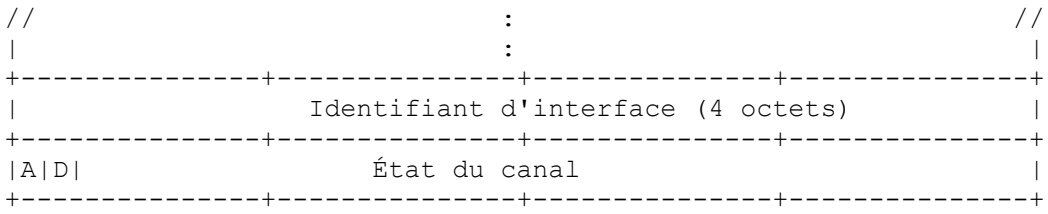
Longueur d'onde : 32 bits. Cette valeur indique la longueur d'onde portée sur l'accès. Les valeurs utilisées dans ce champ n'ont de signification qu'entre deux voisins.

13.13 Classe CHANNEL_STATUS

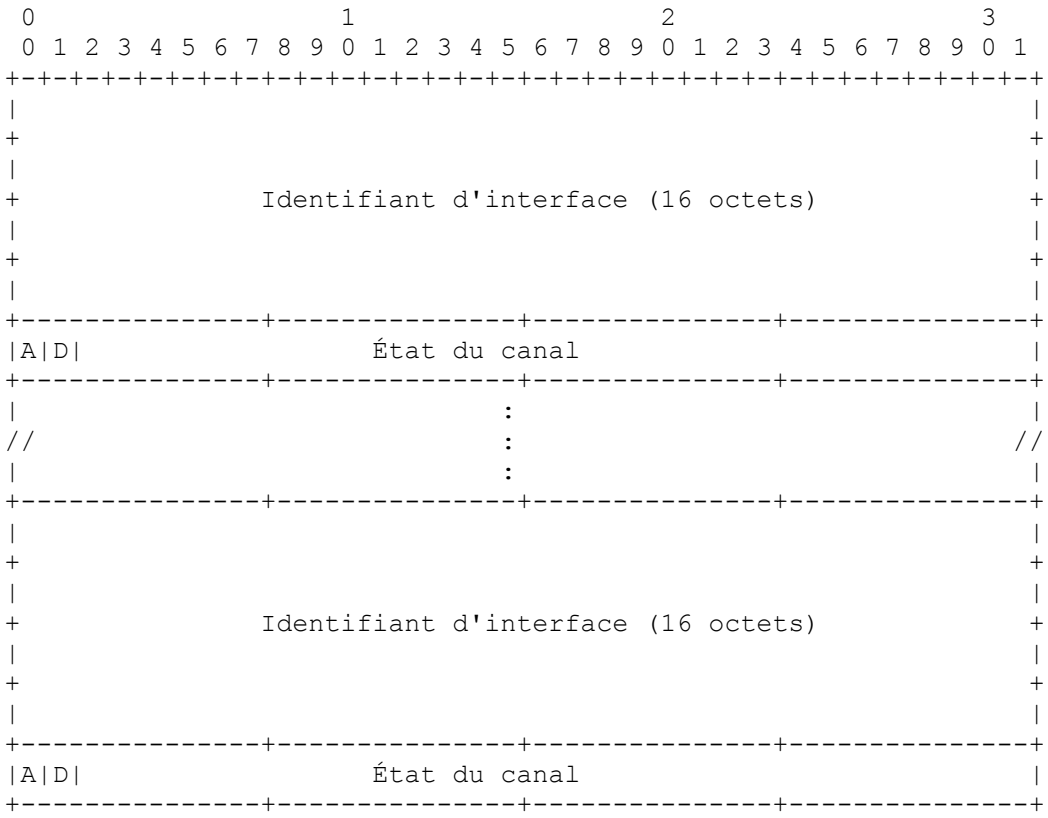
Classe = 13

o C-Type = 1, INTERFACE_ID IPv4

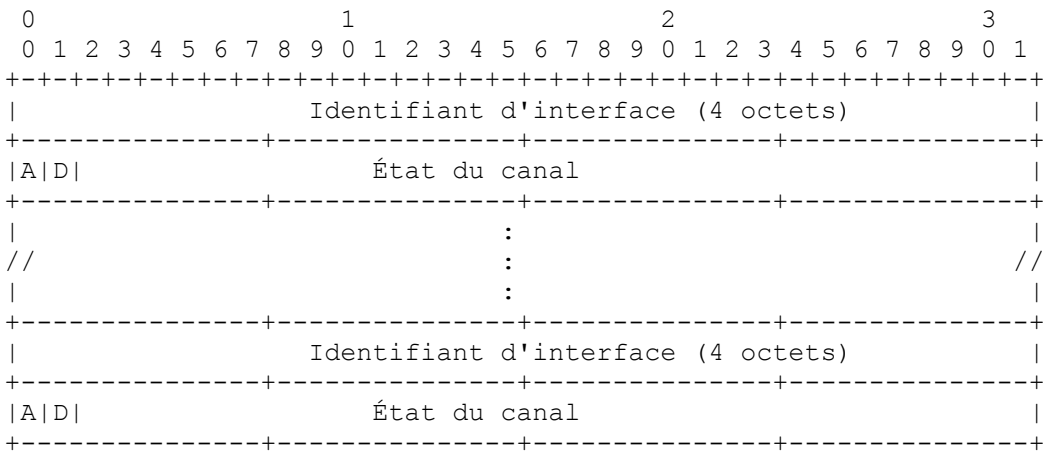




o C-Type = 2, INTERFACE_ID IPv6



o C-Type = 3, INTERFACE_ID non numéroté



Bit A(ctif) : 1 bit. Il indique que le canal est alloué au trafic d'utilisateur et que la liaison de données devrait être activement surveillée.

Bit D(irection) : 1 bit. Il indique la direction (émission/réception) du canal de données référencé dans l'objet CHANNEL_STATUS. Réglé à 1, il indique que le canal de données est dans la direction d'émission.

État du canal : 30 bits. Il indique la condition d'état du canal de données. Les valeurs suivantes sont définies. Toutes les

autres valeurs sont réservées.
1 : Signal OK : le canal est en fonctionnement
2 : Signal dégradé (SD) : une défaillance mineure causée par un BER excédant un seuil pré sélectionné. Le BER spécifique utilisé pour définir le seuil est configuré.
3 : Défaillance du signal (SF) : un signal de défaillance majeure incluant (sans s'y limiter) la perte du signal (LOS), la perte de trame (LOF), ou une ligne de signal d'indication d'alarme (AIS, *Alarm Indication Signal)*.

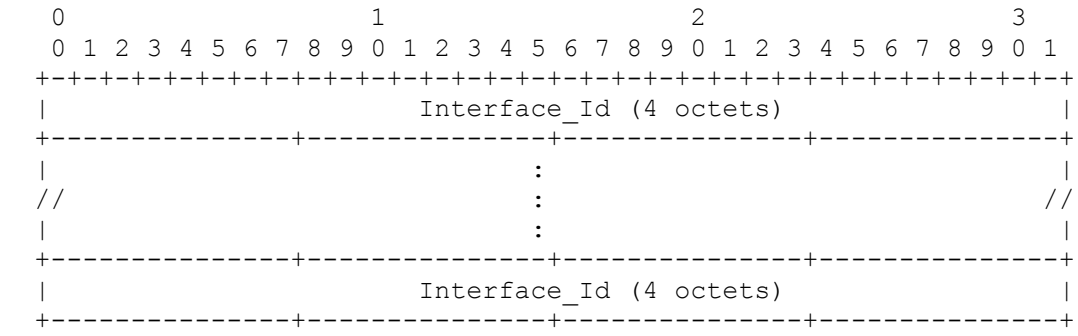
Cet objet contient un ou plusieurs identifiants d'interface suivis par un champ Channel_Status.

Pour indiquer l'état de la liaison TE entière, il DOIT y avoir un seul Interface_Id, et il DOIT être à zéro. Cet objet n'est pas négociable.

13.14 Classe CHANNEL_STATUS_REQUEST

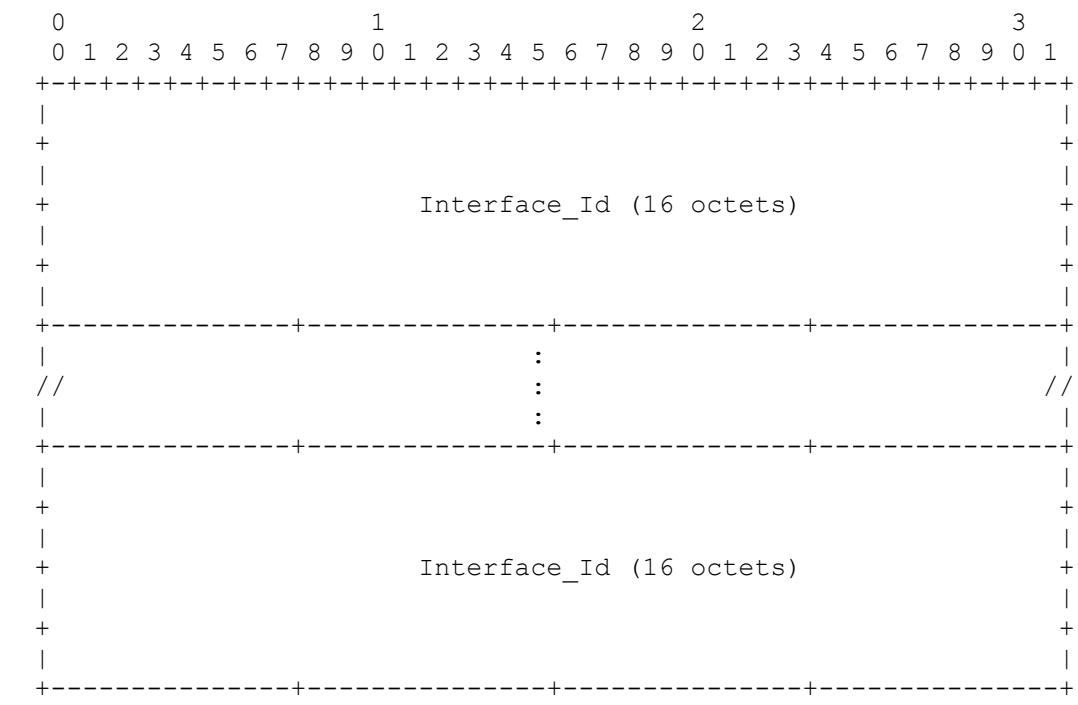
Classe = 14

o C-Type = 1, INTERFACE_ID IPv4



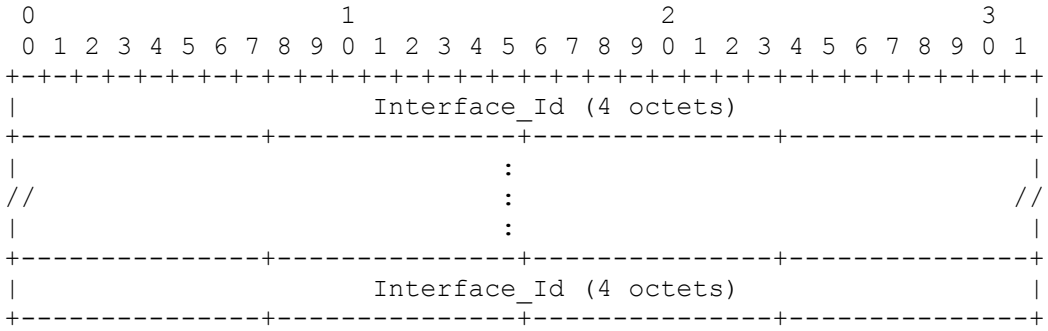
Cet objet contient un ou plusieurs Interface_Id.
La longueur de cet objet est 4 + 4N en octets, où N est le nombre des Interface_Id.

o C-Type = 2, INTERFACE_ID IPv6



Cet objet contient un ou plusieurs Interface_Id.
La longueur de cet objet est 4 + 16N en octets, où N est le nombre des Interface_Id.

o C-Type = 3, INTERFACE_ID non numéroté

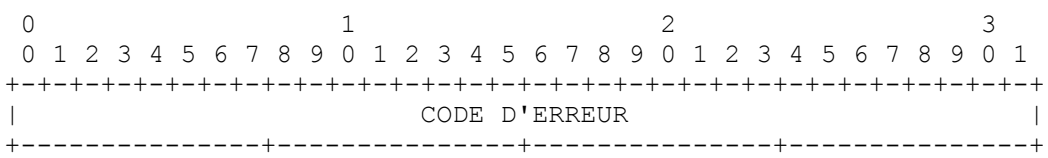


Cet objet contient un ou plusieurs Interface_Id.
La longueur de cet objet est 4 + 4N en octets, où N est le nombre des Interface_Id.
Cet objet n'est pas négociable.

13.15 Classe ERROR_CODE

Classe = 20

o C-Type = 1, BEGIN_VERIFY_ERROR

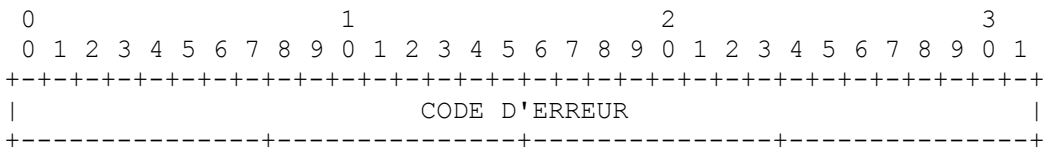


Les valeurs binaires suivantes sont définies dans l'ordre des octets du réseau (c'est-à-dire l'ordre gros boutien des octets):
0x01 = Procédure de vérification de liaison non acceptée.
0x02 = Ne veut pas vérifier.
0x04 = Mécanisme de transport de vérification non accepté.
0x08 = erreur de configuration de l'identifiant de liaison.
0x10 = C-Type d'objet inconnu.

Toutes les autres valeurs de bits sont réservées et devraient être envoyées à zéro et ignorées à réception. Plusieurs bits peuvent être établis pour indiquer plusieurs erreurs. Cet objet n'est pas négociable.

Si un message BeginVerifyNack est reçu avec le code d'erreur 2, le nœud qui a généré le BeginVerify DEVRAIT programmer une retransmission de BeginVerify après Rf secondes, où Rf est un paramètre défini en local.

o C-Type = 2, LINK_SUMMARY_ERROR



Les valeurs binaires suivantes sont définies dans l'ordre des octets du réseau (c'est-à-dire l'ordre gros boutien des octets):
0x01 = Paramètres LINK_SUMMARY non négociables inacceptables.
0x02 = Renégocier les paramètres LINK_SUMMARY.
0x04 = Objet TE_LINK invalide.
0x08 = Objet DATA_LINK invalide.
0x10 = C-Type d'objet TE_LINK inconnu.
0x20 = C-Type d'objet DATA_LINK inconnu.

Toutes les autres valeurs de bits sont réservées et devraient être envoyées à zéro et ignorées à réception. Plusieurs bits peuvent être établis pour indiquer plusieurs erreurs. Cet objet n'est pas négociable.

8. Références

14.1 Références normatives

[RFC2119] S. Bradner, "Mots clés à utiliser dans les RFC pour indiquer les niveaux d'exigence", BCP 14, mars 1997. *(MàJ par* RFC8174)

[RFC2402] S. Kent et R. Atkinson, "En-tête d'authentification IP", novembre 1998. (*Obsolète, voir* RFC4302, 4305)

[RFC2406] S. Kent et R. Atkinson, "Encapsulation de charge utile de sécurité IP (ESP)", novembre 1998. *(Obsolète, voir* RFC4303)

[RFC2407] D. Piper, "Le domaine d'interprétation de sécurité IP de l'Internet pour ISAKMP", novembre 1998. *(Obsolète, voir* 4306)

[RFC2409] D. Harkins et D. Carrel, "Échange de clés Internet (IKE)", novembre 1998. *(Obsolète, voir la* RFC4306)

[RFC2961] L. Berger et autres, "Extensions de réduction de redondance de rafraîchissement pour RSVP", avril 2001. (*MàJ par* RFC5063) *(P.S.)*

[RFC3471] L. Berger, éd., "Commutation d'étiquettes multi-protocoles généralisée (GMPLS) : description fonctionnelle de la signalisation", janvier 2003. (*MàJ par* RFC4201, RFC4328, RFC4872, RFC8359) *(P.S.)*

[RFC4201] K. Kompella et autres, "Faisceaux de liaisons dans l'ingénierie du trafic MPLS", octobre 2005. *(P.S.)*

[RFC4202] K. Kompella et autres, "Extensions d'acheminement pour la prise en charge de la commutation généralisée d'étiquettes multi-protocoles (GMPLS)", octobre 2005. *(P.S.)*

14.2 Références pour information

[RFC2401] S. Kent et R. Atkinson, "Architecture de sécurité pour le protocole Internet", novembre 1998. (*Obsolète, voir* RFC4301)

[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* RFC5226)

[RFC3209] D. Awduche, et autres, "RSVP-TE : Extensions à RSVP pour les tunnels LSP", décembre 2001. (*Mise à jour par* RFC3936, RFC4420, RFC4874, RFC5151, RFC5420, RFC6790)

[RFC3630] D. Katz, K. Kompella et D. Yeung, "Extensions d'ingénierie de trafic à OSPF version 2", septembre 2003.

[RFC3784] H. Smit, T. Li, "Extensions de système intermédiaire à système intermédiaire (IS-IS) pour l'ingénierie du trafic (TE)", juin 2004. (*Obsolète, voir* RFC5305) (*MàJ par* RFC4205) *(Information)*

15. Considérations sur la sécurité

Une session de protocole LMP peut subir un certain nombre d'attaques. Voici quelques exemples :
- o un adversaire peut fabriquer de faux paquets de contrôle ;
- o un adversaire peut modifier les paquets de contrôle en transit ;
- o un adversaire peut répéter des paquets de contrôle ;
- o un adversaire peut étudier un certain nombre de paquets de contrôle et essayer de casser la clé en utilisant des outils cryptographiques. Si l'algorithme de hachage/chiffrement utilisé a des faiblesses connues, il devient alors facile à l'adversaire de découvrir la clé en utilisant des outils simples.

Cette Section spécifie un mécanisme de sécurité fondés sur IPsec pour LMP.

15.1 Exigences pour la sécurité

Les exigences suivantes s'appliquent au mécanisme décrit dans cette section.

- La sécurité de LMP DOIT être capable d'assurer l'authentification, la protection de l'intégrité, et contre la répétition.
- Pour le trafic LMP, la confidentialité n'est pas nécessaire. Seule l'authentification est nécessaire pour assurer que les paquets de contrôle (paquets envoyés sur le canal de contrôle LMP) sont générés au bon endroit et n'ont pas été modifiés dans le transit. Les paquets LMP Test échangés sur les liaisons de données n'ont pas besoin d'être protégés.
- Pour le trafic LMP, protéger l'identité des points d'extrémité LMP n'est généralement pas exigé.
- Le mécanisme de sécurité devrait assurer des schémas de gestion de clé bien définis. Les schémas de gestion de clé devraient être bien analysés pour être cryptographiquement sûrs. Les schémas de gestion de clé devraient être adaptables. De plus, le système de gestion de clé devrait être automatique.
- Les algorithmes utilisés pour l'authentification DOIVENT être cryptographiquement cohérents. Le protocole de sécurité DOIT aussi permettre la négociation et l'utilisation de différents algorithmes d'authentification.

15.2 Mécanismes de sécurité

IPsec est une suite de protocoles qui est utilisée pour sécuriser la communication entre deux homologues à la couche réseau. Ce protocole se compose du document d'architecture de la sécurité IP [RFC2401], de IKE [RFC2409], de IPsec AH [RFC2402], et de IPsec ESP [RFC2406]. IKE est le protocole de gestion de clé pour les réseaux IP, tandis que AH et ESP sont utilisés pour protéger le trafic IP. IKE est défini pour le domaine d'interprétation spécifique de IP.

Considérant les exigences décrites au paragraphe 15.1, il est recommandé que, lorsque la sécurité est nécessaire pour LMP, les mises en œuvre utilisent IPsec comme décrit ci-dessous :

1. Les mises en œuvre de LMP sur le protocole IPsec DEVRAIENT prendre en charge le mode de gestion de clé manuel. Le mode de gestion de clé manuel donne un moyen facile pour établir et diagnostiquer la fonction IPsec.
   Cependant, on note que le mode de gestion de clé manuel ne peut pas efficacement prendre en charge des dispositifs comme la protection contre la répétition et le changement de clé automatique. Une mise en œuvre qui utilise des clés manuelles doit être consciente de ces limites. Il est recommandé qu'une mise en œuvre utilise la gestion de clé manuelle aux seules fins de diagnostic et utilise un protocole de gestion de clé dynamique pour utiliser des dispositifs comme la protection contre la répétition et le changement de clé automatique.

2. IPsec ESP avec authentification d'en-queue en mode tunnel DOIT être pris en charge.

3. Les mises en œuvre DOIVENT prendre en charge les protocoles d'échange de clé authentifié. IKE [RFC2409] DOIT être utilisé comme protocole d'échange de clé si les clés sont négociées de façon dynamique entre les homologues.

4. Les mises en œuvre DOIVENT utiliser le domaine d'interprétation IPsec [RFC2407].

5. Pour le protocole IKE, les identités des associations de sécurité (SA) négociées en mode rapide représentent le trafic que les homologues s'accordent pour protéger et se composent des informations d'espace d'adresse, de protocole, et d'accès. Pour LMP sur IPsec, il est recommandé que la charge utile d'identité pour le mode rapide contienne les informations suivantes : les identités DOIVENT être du type adresse IP et la valeur des identités DEVRAIT être l'adresse IP des homologues communicants. Le champ Protocole DOIT être UDP. Le champ Accès DEVRAIT être réglé à zéro pour indiquer que les champs Accès devraient être ignorés. Cela implique que tout le trafic UDP entre les homologues doit être envoyés à travers le tunnel IPsec. Si une mise en œuvre prend en charge les sélecteurs fondés sur l'accès, elle peut opter pour un sélecteur de granularité plus fine en spécifiant le champ Accès comme accès LMP. Si, cependant, l'homologue n'utilise pas de sélecteurs fondés sur l'accès, la mise en œuvre DOIT revenir à l'utilisation d'une valeur de sélecteur de 0.

6. Le mode agressif de négociation IKE DOIT être pris en charge. Lorsque IPsec est configuré à être utilisé avec un homologue, tous les messages LMP sont supposés être envoyés sur le tunnel IPsec (crypto canal). De même, un receveur LMP configuré à utiliser IPsec avec un homologue devrait rejeter tout trafic LMP qui ne vient pas à travers le crypto canal. Le crypto canal peut être pré établi avec le voisin LMP, ou le premier message LMP envoyé à l'homologue peut déclencher la création du tunnel IPsec. Un ensemble de canaux de contrôle peut partager le même crypto canal. Lorsque des Hello LMP sont utilisés pour surveiller l'état du canal de contrôle, il est important de se souvenir que la défaillance de garde en vie dans un canal de contrôle peut aussi être due à une défaillance dans le crypto canal. La méthode suivante est recommandée pour s'assurer qu'un chemin de communication LMP entre deux homologues fonctionne correctement.
   - Si les Hello LMP détectent une défaillance sur un canal de contrôle, passer à un autre canal de contrôle et/ou essayer d'établir un nouveau canal de contrôle.
   - S'assurer de la bonne santé des canaux de contrôle qui utilisent les Hello LMP. Si tous les canaux de contrôle

indiquent une défaillance et qu'il n'est pas possible d'activer un nouveau canal de contrôle, supprimer tous les canaux de contrôle existants. Aussi, supprimer le crypto canal (la SA IKE et les SA IPsec).

- o Rétablir le crypto canal. L'échec d'établissement d'un crypto canal indique une défaillance fatale pour la communication LMP.
- o Activer le canal de contrôle. L'échec d'activation du canal de contrôle indique une défaillance fatale pour la communication LMP.

Lorsque les homologues LMP sont découverts de façon dynamique (en particulier l'initiateur) les points suivants devraient être notés :
Lorsque on utilise l'authentification de clés pré partagées en mode de protection d'identité (mode principal) la clé pré partagée est obligée de calculer la valeur de SKEYID (utilisée pour déduire les clés pour chiffrer les messages durant l'échange de clés). En mode principal de IKE, la clé pré partagée à utiliser doit être identifiée avant de recevoir la charge utile d'identité de l'homologue. La clé pré partagée est requise pour le calcul de SKEYID. La seule information disponible sur l'homologue à ce point est son adresse IP à partir de laquelle est venue la négociation. Il n'est pas possible de déchiffrer l'adresse IP d'un homologue pour obtenir la clé pré partagée car les adresses sont dynamiques et ne sont pas connues à l'avance.

Le mode agressif d'échange de clé peut être utilisé car les charges utiles d'identification sont envoyées dans le premier message.

Noter cependant que le mode agressif est enclin à des attaques passives de déni de service. Utiliser un secret partagé (secret partagé de groupe) parmi un certain nombre d'homologues est fortement déconseillé parce que cela ouvre la porte à des attaques par interposition.

L'authentification fondée sur la signature numérique ne pose pas de tels problèmes. Il est RECOMMANDÉ qu'un mécanisme d'authentification fondé sur la signature numérique soit utilisé lorsque possible. Si une authentification fondée sur la clé pré partagée est exigée, le mode agressif DEVRAIT alors être utilisé. Les valeurs de clé d'authentification pré partagées de IKE DEVRAIENT être protégées d'une manière similaire à celle du mot de passe de compte d'utilisateur.

16. Considérations relatives à l'IANA

L'IANA a alloué le numéro d'accès 701 à LMP.

Des lignes directrices sont données ci-après à l'IANA pour l'allocation de chaque espace de noms LMP. Des gammes sont spécifiées pour usage privé, pour être allouées par revue d'expert, et pour être allouées par action de normalisation (comme défini dans la [RFC2434].

Les allocations faites à partir des espaces de noms LMP, mis à part les espaces de noms pour usage privé (c'est-à-dire, pour les extensions propriétaires) n'ont pas besoin d'être documentées. Les mises en œuvre de LMP indépendantes qui utilisent les mêmes codets d'usage privé ne vont en général pas interopérer, de sorte qu'il faut faire attention quand on utilise ces codets dans un réseau multi fabricants.

Les allocations faites dans les espaces de noms LMP par revue d'expert doivent être revues par un expert désigné par l'IESG. L'intention de ce document est que les codets provenant de ces gammes soient utilisés pour des extensions expérimentales ; à ce titre, les allocations DOIVENT être accompagnées de RFC expérimentales. Si le déploiement suggère que ces extensions sont utiles, elles devraient alors être décrites dans des RFC sur la voie de la normalisation, et de nouveaux codets tirés des gammes d'action de normalisation DOIVENT être alloués.

Les allocations faites dans les espaces de noms LMP par action de normalisation DOIVENT être documentées par une RFC sur la voie de la normalisation, normalement soumise à un groupe de travail de l'IETF, mais suivant dans tous les cas les procédures usuelles de l'IETF pour les propositions de normes.

Les bits réservés de l'en-tête commun LMP devraient être alloués par action de normalisation, conformément aux politiques décrites dans la [RFC2434].

LMP définit les espaces de noms suivants qui doivent être gérés :

- Type de message LMP.
- Classe d'objet LMP.
- Type de classe d'objet LMP (C-Type). Ils sont uniques au sein de la classe d'objet.
- Type de classe de sous objet LMP (Type). Ils sont uniques au sein de la classe d'objet.

L'espace de nom de type de message LMP devrait être alloué comme suit : conformément aux politiques mentionnées dans la [RFC2434], les numéros dans la gamme 0-127 sont alloués par action de normalisation, 128-240 sont alloués par revue d'expert, et 241-255 sont réservés pour usage privé.

L'espace de nom de classe d'objet LMP devrait être alloué comme suit : conformément aux politiques mentionnées dans la [RFC2434], les numéros dans la gamme 0-127 sont alloués par action de normalisation, 128-240 sont alloués par revue d'expert, et 241-255 sont réservés pour usage privé.

La politique d'allocation des valeurs hors de l'espace de noms de la classe d'objet LMP fait partie de la définition de l'instance spécifique de classe. Quand une classe est définie, sa définition doit aussi inclure une description de la politique selon laquelle les noms de classe d'objet sont alloués.

La politique pour allouer les valeurs hors de l'espace de noms de classe de sous objet LMP fait partie de la définition de l'instance de classe spécifique. Lorsque une classe est définie, sa définition doit aussi inclure une description de la politique selon laquelle les sous objets sont alloués.

Les espaces de noms suivants ont été alloués par l'IANA :

Espace de noms de type de message LMP :

message Config	(type de message = 1)
message ConfigAck	(type de message = 2)
message ConfigNack	(type de message = 3)
message Hello	(type de message = 4)
message BeginVerify	(type de message = 5)
message BeginVerifyAck	(type de message = 6)
message BeginVerifyNack	(type de message = 7)
message EndVerify	(type de message = 8)
message EndVerifyAck	(type de message = 9)
message Test	(type de message = 10)
message TestStatusSuccess	(type de message = 11)
message TestStatusFailure	(type de message = 12)
message TestStatusAck	(type de message = 13)
message LinkSummary	(type de message = 14)
message LinkSummaryAck	(type de message = 15)
message LinkSummaryNack	(type de message = 16)
message ChannelStatus	(type de message = 17)
message ChannelStatusAck	(type de message = 18)
message ChannelStatusRequest	(type de message = 19)
message ChannelStatusResponse	(type de message = 20)

Espace de noms de classe d'objet LMP et type de classe (C-Type)
o CCID : Nom de classe (1)
L'espace de noms de type de classe d'objet CCID devrait être alloué comme suit : conformément aux politiques mentionnées dans la [RFC2434], les numéros dans la gamme 0-111 sont alloués par action de normalisation, 112-119 sont alloués sur revue par expert, et 120-127 sont réservés pour usage privé.
- LOCAL_CCID : (C-Type = 1)
- REMOTE_CCID : (C-Type = 2)

o NODE_ID : Nom de classe (2)
L'espace de noms de type de classe d'objet NODE ID devrait être alloué comme suit : conformément aux politiques mentionnées dans la [RFC2434], les numéros dans la gamme 0-111 sont alloués par action de normalisation, 112-119 sont alloués sur revue par expert, et 120-127 sont réservés pour usage privé.
- LOCAL_NODE_ID : (C-Type = 1)
- REMOTE_NODE_ID : (C-Type = 2)

o LINK_ID : Nom de classe (3)
L'espace de noms de type de classe d'objet LINK_ID devrait être alloué comme suit : conformément aux politiques mentionnées dans la [RFC2434], les numéros dans la gamme 0-111 sont alloués par action de normalisation, 112-119 sont alloués sur revue par expert, et 120-127 sont réservés pour usage privé.
- LOCAL_LINK_ID IPv4 : (C-Type = 1)
- REMOTE_LINK_ID IPv4 : (C-Type = 2)
- LOCAL_LINK_ID IPv6 : (C-Type = 3)
- REMOTE_LINK_ID IPv6 : (C-Type = 4)

- LOCAL_LINK_ID non numéroté : (C-Type = 5)
- REMOTE_LINK_ID non numéroté : (C-Type = 6)

o INTERFACE_ID : Nom de classe (4)
L'espace de noms de type de classe d'objet INTERFACE_ID devrait être alloué comme suit : conformément aux politiques mentionnées dans la [RFC2434], les numéros dans la gamme 0-111 sont alloués par action de normalisation, 112-119 sont alloués sur revue par expert, et 120-127 sont réservés pour usage privé.
- LOCAL_INTERFACE_ID IPv4 : (C-Type = 1)
- REMOTE_INTERFACE_ID IPv4 : (C-Type = 2)
- LOCAL_INTERFACE_ID IPv6 : (C-Type = 3)
- REMOTE_INTERFACE_ID IPv6 : (C-Type = 4)
- LOCAL_INTERFACE_ID non numéroté : (C-Type = 5)
- REMOTE_INTERFACE_ID non numéroté : (C-Type = 6)

o MESSAGE_ID : Nom de classe (5)
L'espace de noms de type de classe d'objet MESSAGE_ID devrait être alloué comme suit : conformément aux politiques mentionnées dans la [RFC2434], les numéros dans la gamme 0-111 sont alloués par action de normalisation, 112-119 sont alloués sur revue par expert, et 120-127 sont réservés pour usage privé.
- MESSAGE_ID : (C-Type = 1)
- MESSAGE_ID_ACK : (C-Type = 2)

o CONFIG : Nom de classe (6)
L'espace de noms de type de classe d'objet CONFIG devrait être alloué comme suit : conformément aux politiques mentionnées dans la [RFC2434], les numéros dans la gamme 0-111 sont alloués par action de normalisation, 112-119 sont alloués sur revue par expert, et 120-127 sont réservés pour usage privé.
- HELLO_CONFIG : (C-Type = 1)

o HELLO : Nom de classe (7)
L'espace de noms de type de classe d'objet HELLO devrait être alloué comme suit : conformément aux politiques mentionnées dans la [RFC2434], les numéros dans la gamme 0-111 sont alloués par action de normalisation, 112-119 sont alloués sur revue par expert, et 120-127 sont réservés pour usage privé.
- HELLO : (C-Type = 1)

o BEGIN_VERIFY : Nom de classe (8)
L'espace de noms de type de classe d'objet BEGIN_VERIFY devrait être alloué comme suit : conformément aux politiques mentionnées dans la [RFC2434], les numéros dans la gamme 0-111 sont alloués par action de normalisation, 112-119 sont alloués sur revue par expert, et 120-127 sont réservés pour usage privé.
- Type 1 : (C-Type = 1)

o BEGIN_VERIFY_ACK : Nom de classe (9)
L'espace de noms de type de classe d'objet BEGIN_VERIFY_ACK devrait être alloué comme suit : conformément aux politiques mentionnées dans la [RFC2434], les numéros dans la gamme 0-111 sont alloués par action de normalisation, 112-119 sont alloués sur revue par expert, et 120-127 sont réservés pour usage privé.
- Type 1 : (C-Type = 1)

o VERIFY_ID : Nom de classe (10)
L'espace de noms de type de classe d'objet VERIFY_ID devrait être alloué comme suit : conformément aux politiques mentionnées dans la [RFC2434], les numéros dans la gamme 0-111 sont alloués par action de normalisation, 112-119 sont alloués sur revue par expert, et 120-127 sont réservés pour usage privé.- Type 1 : (C-Type = 1)

o TE_LINK : Nom de classe (11)
L'espace de noms de type de classe d'objet TE_LINK devrait être alloué comme suit : conformément aux politiques mentionnées dans la [RFC2434], les numéros dans la gamme 0-111 sont alloués par action de normalisation, 112-119 sont alloués sur revue par expert, et 120-127 sont réservés pour usage privé.
- TE_LINK IPv4 : (C-Type = 1)
- TE_LINK IPv6 : (C-Type = 2)
- TE_LINK non numéroté : (C-Type = 3)

o DATA_LINK : Nom de classe (12)
L'espace de noms de type de classe d'objet DATA_LINK devrait être alloué comme suit : conformément aux politiques mentionnées dans la [RFC2434], les numéros dans la gamme 0-111 sont alloués par action de normalisation, 112-119 sont alloués sur revue par expert, et 120-127 sont réservés pour usage privé.
- DATA_LINK IPv4 : (C-Type = 1)

- DATA_LINK IPv6 : (C-Type = 2)
- DATA_LINK non numéroté : (C-Type = 3)

L'espace de noms de classe de sous objet DATA_LINK devrait être alloué comme suit : conformément aux politiques mentionnées dans la [RFC2434], les numéros dans la gamme of 0-127 sont alloués par action de normalisation, 128-247 sont alloués sur revue par expert, et 248-255 sont réservés pour usage privé.
- Type de commutation d'interface : (Type de sous objet = 1)
- Longueur d'onde : (Type de sous objet = 2)

o CHANNEL_STATUS : Nom de classe (13)
L'espace de noms de type de classe d'objet CHANNEL_STATUS devrait être alloué comme suit : conformément aux politiques mentionnées dans la [RFC2434], les numéros dans la gamme 0-111 sont alloués par action de normalisation, 112-119 sont alloués sur revue par expert, et 120-127 sont réservés pour usage privé.
- INTERFACE_ID IPv4 : (C-Type = 1)
- INTERFACE_ID IPv6 : (C-Type = 2)
- INTERFACE_ID non numéroté : (C-Type = 3)

o CHANNEL_STATUS_REQUEST : Nom de classe (14)
L'espace de noms de type de classe d'objet CHANNEL_STATUS_REQUEST devrait être alloué comme suit : conformément aux politiques mentionnées dans la [RFC2434], les numéros dans la gamme 0-111 sont alloués par action de normalisation, 112-119 sont alloués sur revue par expert, et 120-127 sont réservés pour usage privé.
- INTERFACE_ID IPv4 : (C-Type = 1)
- INTERFACE_ID IPv6 : (C-Type = 2)
- INTERFACE_ID non numéroté : (C-Type = 3)

o ERROR_CODE : Nom de classe (20)
L'espace de noms de type de classe d'objet ERROR_CODE devrait être alloué comme suit : conformément aux politiques mentionnées dans la [RFC2434], les numéros dans la gamme 0-111 sont alloués par action de normalisation, 112-119 sont alloués sur revue par expert, et 120-127 sont réservés pour utilisation privée.
- BEGIN_VERIFY_ERROR : (C-Type = 1)
- LINK_SUMMARY_ERROR : (C-Type = 2)

17. Remerciements

Les auteurs tiennent à remercier André Fredette de ses nombreuses contributions au présent document, ainsi que Ayan Banerjee, George Swallow, Adrian Farrel, Dimitri Papadimitriou, Vinay Ravuri, et David Drysdale pour leurs précieux commentaires et suggestions. Merci aussi à John Yu, Suresh Katukam, et Greg Bernstein de leurs utiles suggestions sur l'applicabilité du canal de contrôle dans la bande.

18. Contributeurs

Jonathan P. Lang
Sonos, Inc.
223 E. De La Guerra St.
Santa Barbara, CA 93101
mél : jplang@ieee.org

John Drake
Calient Networks
5853 Rue Ferrari
San Jose, CA 95138
téléphone : (408) 972-3720
mél : jdrake@calient.net

Krishna Mitra
Independent Consultant
mél : kmitra@earthlink.net

Kireeti Kompella
Juniper Networks, Inc.
1194 North Mathilda Avenue
Sunnyvale, CA 94089
mél : kireeti@juniper.net

Yakov Rekhter
Juniper Networks, Inc.
1194 North Mathilda
AvenueSunnyvale, CA 94089
mél : yakov@juniper.net

Lou Berger
Movaz Networks
mél : lberger@movaz.com

Debanjan Saha
IBM Watson Research Center
mél : dsaha@us.ibm.com

Debashis Basak
Accelight Networks
70 Abele Road, Suite 1201
Bridgeville, PA 15017-3470
mél : dbasak@accelight.com

Hal Sandick
Shepard M.S.
2401 Dakota Street
Durham, NC 27705
mél : sandick@nc.rr.com

Bala Rajagopalan
Intel Corp.
2111 NE 25th Ave
Hillsboro, OR 97123
mél : bala.rajagopalan@intel.com

Alex Zinin
Alcatel
mél : alex.zinin@alcatel.com

Sankar Ramamoorthi
Juniper Networks, Inc.
1194 North Mathilda Avenue
Sunnyvale, CA 94089
mél : sankarr@juniper.net

Adresse de contact

Jonathan P. Lang
Sonos, Inc.
829 De La Vina, Suite 220
Santa Barbara, CA 93101

mél : jplang@ieee.org

Déclaration complète de droits de reproduction

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