Groupe de travail Réseau
Request for Comments : 1819
RFC rendues obsolètes : 1190, IEN 119
Catégorie : Expérimentale

Groupe de travail ST2
L. Delgrossi et L. Berger, éditeurs
août 1995
Traduction Claude Brière de L'Isle

Spécification du protocole de flux Internet version 2 (ST2) - Version ST2+

Statut de ce mémoire
Le présent mémoire définit un protocole expérimental pour la communauté de l'Internet. Il ne constitue pas une norme de l'Internet. Les discussions et les suggestions pour son amélioration sont les bienvenues. La distribution du présent mémoire n'est soumise à aucune restriction

Note de l'IESG
Le présent document est une révision de la RFC 1190. Le mandat donné à ses éditeurs était de préciser, simplifier et corriger les erreurs de la RFC1190 pour assurer l'interopérabilité des mises en œuvre.

NOTE : Ni la version du protocole décrit dans le présent document ni sa version précédente ne sont des normes de l'Internet et ce statut n'est pas envisagé pour elles. Depuis la publication de la version originale de ce protocole sont survenus des développements significatifs de l'état de l'art. Les lecteurs devraient noter que des normes et des technologies adoptant d'autres approches du problème de la réservation de ressource sont en cours de développement au sein de l'IETF.

Résumé
Le présent mémoire contient une spécification révisée du protocole de flux Internet version 2 (ST2). ST2 est un protocole de réservation de ressource expérimental destiné à fournir une garantie de temps réel de bout en bout sur un internet. Il permet aux applications de construire des flux de données unidirectionnels multi-destination avec la qualité de service désirée. La version révisée de ST2 spécifiée dans ce mémoire est appelée ST2+.

Cette spécification est un produit du groupe de travail STream Protocol de l'équipe d'ingénierie de l'Internet (IETF).

Table des matières

1. Introduction....2
1.1 Qu'est ce que ST2 ?....2
1.2 ST2 et IP....3
1.3 Historique du protocole....3
1.4 Modules de prise en charge pour ST2....4
1.5 Concepts de base de ST2....6
1.6 Organisation du document....8
2. Description du service d'utilisateur ST2....8
2.1 Opérations de flux et fonctions de primitives....8
2.2 Diagrammes d'état....9
2.3 Tableaux de transition d'état....10
3. Protocole ST2 de transfert de données....11
3.1 Transfert de données avec ST....11
3.2 Fonctions du protocole ST....12
4. Description fonctionnelle de SCMP....12
4.1 Types de flux....13
4.2 PDU de contrôle....14
4.3 Fiabilité de SCMP....15
4.4 Options de flux....15
4.5 Établissement de flux....16
4.6 Modification d'un flux existant....19
4.7 Suppression de flux....21
5. Cas exceptionnels....21
5.1 Messages ST trop longs....21
5.2 Défaillance de temporisation....22
5.3 Échec d'établissement dû à une défaillance d'acheminement....24
5.4 Problèmes dus à des incohérences d'acheminement....25
5.5 Problèmes de réservation de ressources....25

5.6 Problèmes causés par des messages CHANGE....26
5.7 Cibles inconnues dans DISCONNECT et CHANGE....27
6. Détection de défaillance et récupération....27
6.1 Détection de défaillance....27
6.2 Récupération des défaillances....29
6.3 Préemption de flux....30
7. Groupe de flux....31
7.1 Relations de groupe de base....31
7.2 Orthogonalité des relations....33
8. Fonctions auxiliaires....33
8.1 Génération de l'identifiant de flux....33
8.2 Générateur de nom de groupe....33
8.3 Calcul de somme de contrôle....33
8.4 Identification d'agent ST voisin et collecte d'informations....34
8.5 Estimation du délai d'aller-retour....34
8.6 Découverte de la MTU de réseau....34
8.7 Encapsulation IP de ST....35
8.8 Diffusion groupée IP....35
9. Spécification du flux ST2+....36
9.1 FlowSpec version n° 0 - (FlowSpec nul)....36
9.2 FlowSpec version n° 7 - FlowSpec ST2+....37
10. Spécification des unités de données de protocole ST2....39
10.1 PDU de données....39
10.2 PDU de contrôle....40
10.3 Éléments SCMP communs....40
10.4 PDU de message de contrôle ST....44
10.5 Constantes de protocole suggérées....52
10.6 Notations des données....54
11. Références....55
12. Considérations pour la sécurité....56
13. Remerciements et adresse des auteurs....56

1. Introduction

1.1 Qu'est ce que ST2 ?

Le protocole de flux Internet, version 2 (ST2) est un protocole expérimental d'inter-réseautage orienté connexion qui fonctionne à la même couche que IP sans connexion. Il a été développé pour assurer la livraison efficace de flux de données à des destinations uniques ou multiples dans des applications qui exigent une qualité de service garantie. ST2 fait partie de la famille de protocoles IP et sert d'ajout, et non de remplacement, à IP. Les principaux domaines d'application du protocole sont le transport en temps réel de données multimédia, par exemple, des flux de paquets d'audio et vidéo numérique, et les simulations/jeux répartis à travers des internets.

ST2 peut être utilisé pour réserver de la bande passante pour des flux en temps réel à travers des chemins de réseau. Cette réservation, conjointement avec les mécanismes appropriés d'accès réseau et de programmation de paquets dans tous les nœuds qui acceptent le protocole, garantit une qualité de service (QS) bien définie aux applications ST2. Elle assure que les paquets en temps réel sont livrés dans les délais, c'est-à-dire, au moment ou il est nécessaire qu'ils soient présentés. Cela facilite une livraison en douceur des données qui est essentielle pour les applications sensibles au délai, mais qui ne peut normalement être fournie pour la communication IP au mieux.

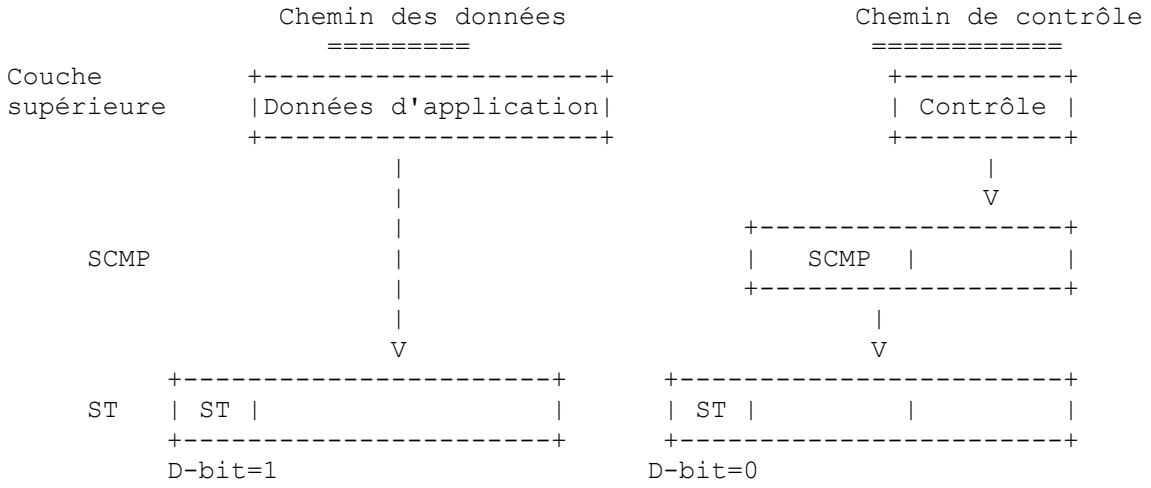


Figure 1 : Chemin des données et du contrôle ST2

Tout comme IP, ST2 consiste en fait en deux protocoles : ST pour le transport des données et SCMP, le protocole de message de contrôle de flux *(Stream Control Message Protocol)* pour toutes les fonctions de contrôle. ST est simple et ne contient qu'un seul format de PDU qui est conçu pour une transmission rapide et efficace des données afin de réaliser de faibles délais de communication. SCMP est cependant plus complexe que l'ICMP de IP. Comme avec ICMP et IP, les paquets SCMP sont transférés au sein de paquets ST comme indiqué à la Figure 1.

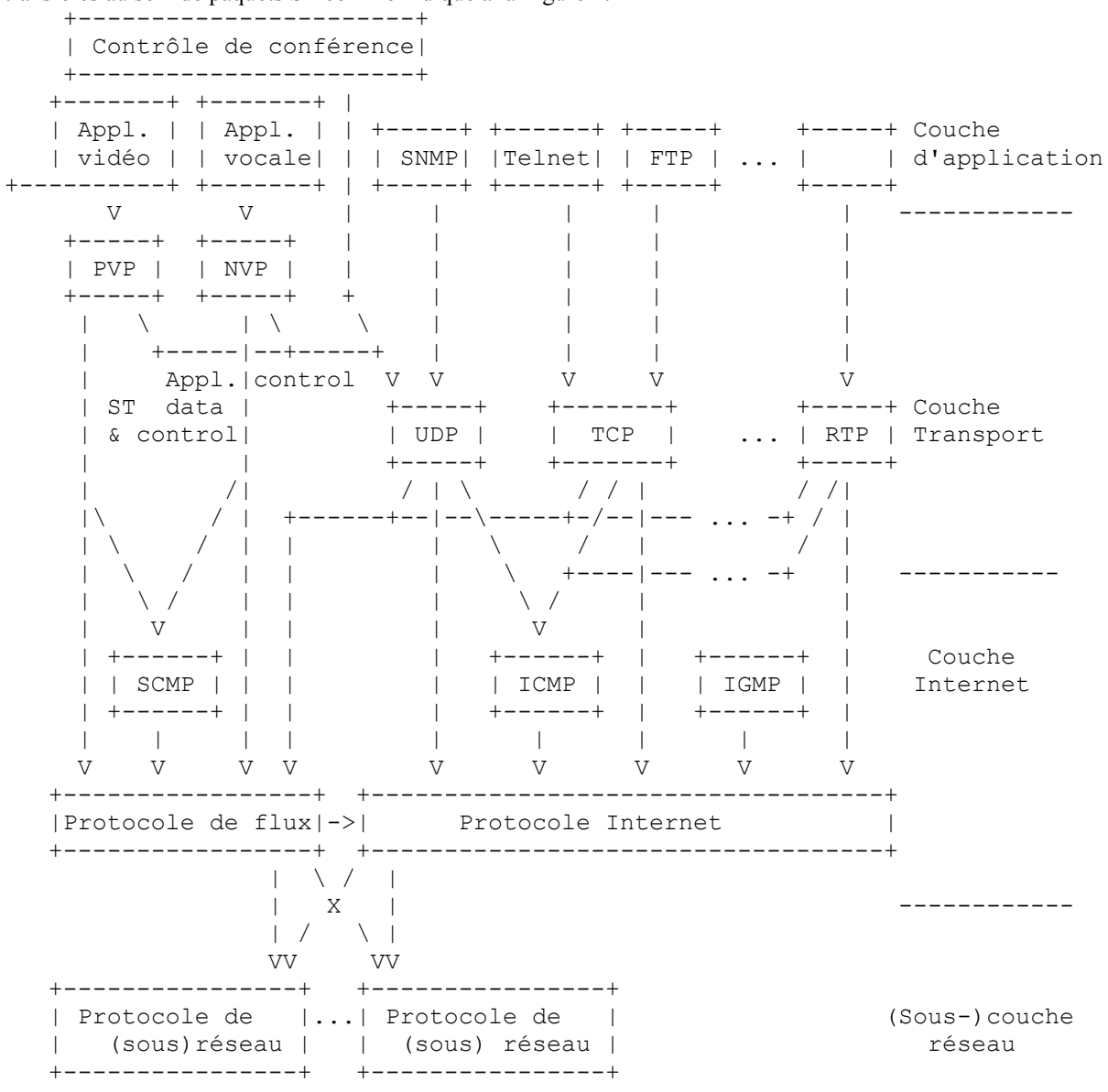


Figure 2 : Relations entre les protocoles

1.2 ST2 et IP

ST2 est conçu pour coexister avec IP sur chaque nœud. Une application multimédia distribuée normale utiliserait les deux protocoles : IP pour le transfert de données traditionnelles et les information de contrôle, et ST2 pour le transfert des données en temps réel. Alors que l'on accède normalement à IP à partir de TCP ou d'UDP, on accèdera à ST2 via de nouveaux protocoles en temps réel de bout en bout. La position de ST2 par rapport aux autres protocoles de la famille Internet est représentée à la Figure 2.

ST2 et IP appliquent tous deux les mêmes schémas d'adressage pour identifier les différent hôtes. Les paquets ST2 et IP diffèrent dans les quatre premiers bits, qui contiennent le numéro de version de protocole d'inter-réseautage : le numéro 5 est réservé pour ST2 (IP a quant à lui le numéro de version 4). En tant que protocole de couche réseau, comme IP, ST2 fonctionne indépendamment de ses sous-réseaux sous-jacents. Les mises en œuvre existantes utilisent ARP pour la résolution d'adresse, et utilisent les mêmes SAP de couche 2 que IP.

À titre de fonction spéciale, les messages ST2 peuvent être encapsulés dans les paquets IP. Cela est représenté à la Figure 2 par une liaison entre ST2 et IP. Cette liaison permet aux messages ST2 de passer à travers les routeurs qui ne traitent pas ST2. La gestion de ressource n'est normalement pas disponible pour ces segments de chemin IP. L'encapsulation dans IP n'est donc suggérée que pour les portions du réseau qui ne constituent pas un goulot d'étranglement du système.

À la Figure 2, le protocole RTP est montré comme exemple de couche transport par dessus ST2. D'autres exemples seraient le protocole de paquet vidéo (PVP, *Packet Video Protocol*) [Cole81], le protocole de la voix sur le réseau (NVP, *Network Voice Protocol*) [Cohe81], et d'autres tels que le protocole de transport de Heidelberg (HeiTP, *Heidelberg Transport Protocol*) [DHHS92].

1.3 Historique du protocole

La première version de ST a été publiée à la fin des années 1970 et a été utilisé pendant la décennie 80 pour de la transmission expérimentale de voix, vidéo, et de la simulation répartie. L'expérience gagnée dans ces applications a conduit au développement de la version révisée du protocole ST2. La révision étend le protocole original pour le rendre plus complet et plus applicable aux environnements multimédia émergeants. La spécification de cette version du protocole est contenue dans la RFC 1190 qui a été publiée en octobre 1990 [RFC1190].

L'intérêt pour ST2 a crû ces dernières années avec des développements de plus en plus nombreux d'applications commerciales multimédia distribuées en cours et avec l'insatisfaction croissante à l'égard de la qualité de la transmission pour l'audio et la vidéo sur IP dans le MBONE. Les compagnies ont des produits disponibles qui incorporent le protocole. Le projet BERKOM MMTS des PTT allemandes [DeAl92] utilise ST2 comme protocole central pour la fourniture de téléservices multimédia tels que la téléconférence et la messagerie. De plus, des mises en œuvre de ST2 par des plates-formes Digital Equipment, IBM, NeXT, Macintosh, PC, Silicon Graphics, et Sun sont disponibles.

En 1993, l'IETF a lancé un nouveau groupe de travail sur ST2 au titre des efforts en cours pour développer des protocoles qui traitent les questions de réservation de ressources. La mission du groupe été de revoir la spécification du protocole existant pour assurer une meilleure interopérabilité entre les mises en œuvre existantes et émergeantes. Elle était aussi de produire une mise à jour de la spécification du protocole expérimental qui reflète l'expérience acquise avec les mises en œuvre et applications existantes de ST2. C'est ce qui a conduit à la spécification du protocole ST2+ contenue dans le présent document.

1.3.1 Différences majeures entre ST de la RFC 1190 et ST2+

Les changements apportés au protocole depuis la RFC1190 ont été motivés par sa simplification, par le besoin d'éclaircir certains points, et la codification des extensions dans les mises en œuvre existantes. Ce paragraphe donne la liste des différences majeures, et ne présente probablement d'intérêt que pour ceux qui connaissent la RFC1190. Les différences majeures entre ces deux versions sont :

- Élimination des "Identifiants de bond" ou HID. Les HID ajoutaient beaucoup de complexité au protocole et ont été considérés comme un empêchement majeur à l'interopérabilité. Les HID ont été remplacés par des identifiants uniques au monde appelés "Identifiants de flux" ou SID.

- Élimination d'un certain nombre d'options de flux. Un certain nombre d'options ne se trouvent utilisées par aucune mise en œuvre, ou sont estimées apporter plus de complexité que de valeur ajoutée. Ces options ont été retirées. Les options retirées sont le point à point, le full duplex, l'inversion des charges, et la route de source.

- Élimination du concept de "sous-ensemble" de mise en œuvre. La RFC1190 permettait des sous-ensembles de mise en

œuvre, pour permettre une mise en œuvre et une expérimentation faciles. Cela a conduit à des problèmes d'interopérabilité. Les agents qui mettent en œuvre le protocole spécifié dans ce document DOIVENT mettre en œuvre le protocole complet. Un certain nombre des fonctions du protocole sont au mieux. Il est prévu que quelques mises en œuvre fassent mieux que les autres pour satisfaire des demandes particulières du protocole.

- Précision sur la capacité des cibles à demander à rejoindre un flux. La RFC1190 peut être interprétée comme prenant en charge les demandes des cibles, mais la plupart des mises en œuvre ne comprenaient pas cela et elles n'ajoutaient pas cette capacité. L'absence de cette capacité a été estimé une limitation significative de la capacité d'échelonner le nombre des participants dans un seul flux ST. Cette précision se fonde sur les travaux de IBM Heidelberg.

- Séparation des fonctions entre ST et modules de support. Un effort a été fait pour améliorer la séparation des fonctions fournies par ST et celles fournies par les autres modules. Ceci est reflété dans la réorganisation du texte et de certains formats de PDU. ST a aussi été rendu indépendant de FlowSpec, bien qu'il définisse une FlowSpec pour les besoins des essais et de l'interopérabilité.

- Réorganisation et réécriture générale de la spécification. Ce document a été réorganisé afin d'améliorer la lisibilité et la clarté. Certaines sections ont été ajoutées, et un effort d'amélioration de l'introduction des concepts a été fait.

1.4 Modules de prise en charge pour ST2

ST2 est un élément d'une grande mosaïque. Ce paragraphe présente l'architecture globale de communication et précise le rôle de ST2 par rapport à ses modules de prise en charge.

ST2 propose un modèle de communication en deux étapes. Dans la première sont construits les canaux en temps réel pour les transferts de données ultérieurs. C'est ce qu'on appelle l'établissement de flux. Il inclut le choix des chemins vers les destinations et la réservation des ressources correspondantes. Dans la seconde étape, les données sont transmises sur les flux précédemment établis. C'est ce qu'on appelle le transfert de données. Alors que l'établissement des flux n'a pas a être réalisé en temps réel, le transfert des données a des exigences strictes de temps réel. L'architecture utilisée pour décrire le modèle de communication ST2 comporte :

- un protocole de transfert de données pour la transmission de données en temps réel sur les flux établis,
- un protocole d'établissement pour installer les flux en temps réel sur la base de la spécification de flux,
- une spécification de flux pour exprimer les exigences de temps réel de l'utilisateur,
- une fonction d'acheminement pour choisir les chemins sur l'Internet,
- un gestionnaire de ressources locales pour traiter de façon appropriée les ressources impliquées dans la communication.

Le présent document définit un protocole de données (ST), un protocole d'établissement (SCMP), et une spécification de flux (ST2+ FlowSpec). Il ne définit pas de fonction d'acheminement ni de gestionnaire de ressources locales. Cependant, ST2 suppose leur existence.

Des architectures de remplacement sont possibles, voir dans la [RFC1633] un exemple d'architecture de remplacement qui pourrait être utilisée lors de la mise en œuvre de ST2.

1.4.1 Protocole de transfert de données

Le protocole de transfert de données définit le format des paquets de données qui appartiennent au flux. Les paquets de données sont délivrés aux cibles le long des chemins de flux précédemment constitués par le protocole d'établissement. Les paquets de données sont délivrés avec la qualité de service associée au flux.

Les paquets de données contiennent un identifiant de flux unique au monde qui indique à quel flux ils appartiennent.
L'identifiant de flux est aussi connu par le protocole d'établissement, qui l'utilise durant l'établissement du flux. Le protocole de transfert de données pour ST2, appelé simplement ST, est défini de façon complète par le présent document.

1.4.2 Protocole d'établissement

Le protocole d'établissement est chargé d'établir, entretenir et libérer les flux en temps réel. Il s'appuis sur la fonction d'acheminement pour choisir les chemins de la source aux destinations. À chaque hôte/routeur sur ces chemins, il présente la spécification de flux associée au flux au gestionnaire de ressource local. Cela conduit les gestionnaires de ressource à réserver les ressources appropriées pour le flux. Le protocole d'établissement pour ST2 est appelé protocole de message de contrôle de flux (SCMP, *Stream Control Message Protocol*) et il est défini de façon complète par le présent document.

1.4.3 Spécification de flux

La spécification de flux est une structure de données qui inclut les exigences de qualité de service (QS) des applications ST2. À chaque hôte/routeur, elle est utilisée par le gestionnaire de ressource local pour traiter les ressources de la façon appropriée pour satisfaire ces exigences. La distribution de la spécification de flux à tous les gestionnaires de ressource le long des chemins de communication est la tâche du protocole d'établissement. Cependant, le contenu de la spécification de flux est transparent pour le protocole d'établissement, qui transporte simplement la spécification de flux. Toutes les opérations sur la spécification de flux, y compris la mise à jour des champs internes et la comparaison des spécifications de flux sont effectuées par les gestionnaires de ressource.

Le présent document définit un format spécifique de spécification de flux qui permet l'interopérabilité parmi les mises en œuvre de ST2. Cette spécification de flux est destinée à prendre en charge un flux avec un seul taux de transmission pour toutes les destinations du flux. Les mises en œuvre peuvent prendre en charge plus d'un format de spécification de flux et les moyens d'ajouter de nouveaux formats seront fournis lorsqu'ils seront définis à l'avenir. Cependant, le format de spécification de flux doit être cohérent tout au long du flux, c'est-à-dire qu'il n'est pas possible d'utiliser des formats de spécification de flux différents pour des parties différentes du même flux.

1.4.4 Fonction d'acheminement

La fonction d'acheminement est une capacité de génération de chemin externe en envoi individuel. Elle fournit le protocole d'acheminement avec le chemin pour atteindre chacune des destinations désirées. La fonction d'acheminement est appelée bond par bond et fournit les informations sur le prochain bond. Une fois qu'un chemin est choisi par la fonction d'acheminement, il persiste pour toute la durée de vie du flux. La fonction d'acheminement peut essayer d'optimiser, sur la base du nombre de cibles, les ressources demandées, ou utiliser les capacités de diffusion groupée ou de bande passante du réseau local. Autrement, la fonction d'acheminement peut même être fondée sur de simples informations de connexité.

Le protocole d'acheminement n'est pas nécessairement au courant des critères utilisés par la fonction d'acheminement pour le choix des chemins. Cela fonctionne avec tout algorithme de fonction d'acheminement. L'algorithme adopté est une affaire locale pour chaque hôte/routeur et des hôtes/routeurs différents peuvent utiliser des algorithmes différents. L'interface entre protocole d'établissement et fonction d'acheminement est aussi une affaire locale et n'est donc pas spécifiée ici.

La présente version de ST ne prend pas en charge l'acheminement de source. Elle ne prend pas en charge l'enregistrement de chemin. Elle ne comporte pas de dispositions qui permettent l'identification des voisins à capacité ST. L'identification des hôtes/routeurs ST distants n'est pas spécifiquement traitée.

1.4.5 Gestionnaire de ressource local

À chaque hôte/routeur traversé par un flux, le gestionnaire de ressource local (LRM, *Local Resource Manager*) est chargé du traitement des ressources locales. Le LRM sait quelles ressources sont sur le système et quelle capacité elles peuvent fournir. Les ressources comportent :

- o les CPU sur les systèmes d'extrémité et les routeurs pour exécuter le logiciel d'application et de protocole,
- o l'espace de mémoire centrale pour ce logiciel (comme dans tous les systèmes en temps réel, le code devrait être logé dans la mémoire centrale, car l'en sortir se ferait au détriment des performances du système),
- o de l'espace de mémoire tampon pour mémoriser les données, par exemple, paquets de communication, passage à travers les nœuds,
- o adaptateurs de réseau,
- o réseaux de transmission entre les nœuds. Les réseaux peuvent être aussi simples que des liaisons en point à point ou aussi complexes que des réseaux commutés tels que les réseaux en relais de trame et en ATM.

Durant l'établissement et la modification de flux, le LRM est présenté par le protocole d'établissement avec la spécification de flux associée au flux. Pour chaque ressource qu'il détient, le LRM est supposé effectuer les fonctions suivantes :

- o Contrôle d'admission de flux : il vérifie si, étant donnée la spécification de flux, il y a des ressources suffisantes pour traiter le nouveau flux de données. Si les ressources disponibles sont insuffisantes, le nouveau flux de données doit être rejeté.
- o Calcul de la qualité de service : il calcule les meilleures performances possibles que peuvent fournir les ressources pour le nouveau flux de données dans les conditions de trafic actuelles, par exemple, les valeurs de débit et de délai sont calculées.
- o Réservation de ressource : il réserve les capacités de ressources requises pour satisfaire à la QS désirée.

Durant le transfert des données, le LRM est chargé de :

- o Mettre la QS en application : il met en application les exigences de QS en planifiant de façon appropriée l'accès aux ressources. Par exemple, les paquets de données provenant d'une application avec une garantie de court délai doivent être servis avant les données d'une application qui a des limites de délai moins strictes.

Le LRM peut aussi fournir les fonctions supplémentaires suivantes :

- o Régulation des données : pour lisser le trafic de données d'un flux, par exemple, comme avec l'algorithme du baquet qui fuit.
- o Policer : pour empêcher que les applications excèdent leur QS négociée, par exemple, d'envoyer des données à un débit supérieur à celui indiqué dans la spécification de flux.
- o Préemption de flux : pour libérer des ressources pour d'autres flux avec une priorité ou importance supérieure.

Les stratégies adoptées par les LRM pour traiter les ressources dépendent des ressources elles mêmes et peuvent varier selon l'hôte/routeur. Cependant, il est nécessaire que tous les LRM aient la même compréhension de la spécification de flux. L'interface entre le protocole d'établissement et le LRM est une affaire locale pour chaque hôte et n'est donc pas spécifiée dans le présent document. Un exemple de LRM est la technique d'administration de ressource de Heidelberg (HeiRAT, *Heidelberg Resource Administration Technique*) [VoHN93].

On suppose aussi que le LRM fournit des fonctions de comparaison des spécifications de flux, c'est-à-dire, pour décider si une spécification de flux exige une quantité supérieure, égale ou inférieure des capacités de ressource à réserver.

1.5 Concepts de base de ST2

Les paragraphes qui suivent présentent à un niveau introductif quelques uns des concepts fondamentaux de ST2 parmi lesquels les flux, le transfert de données et la spécification de flux.

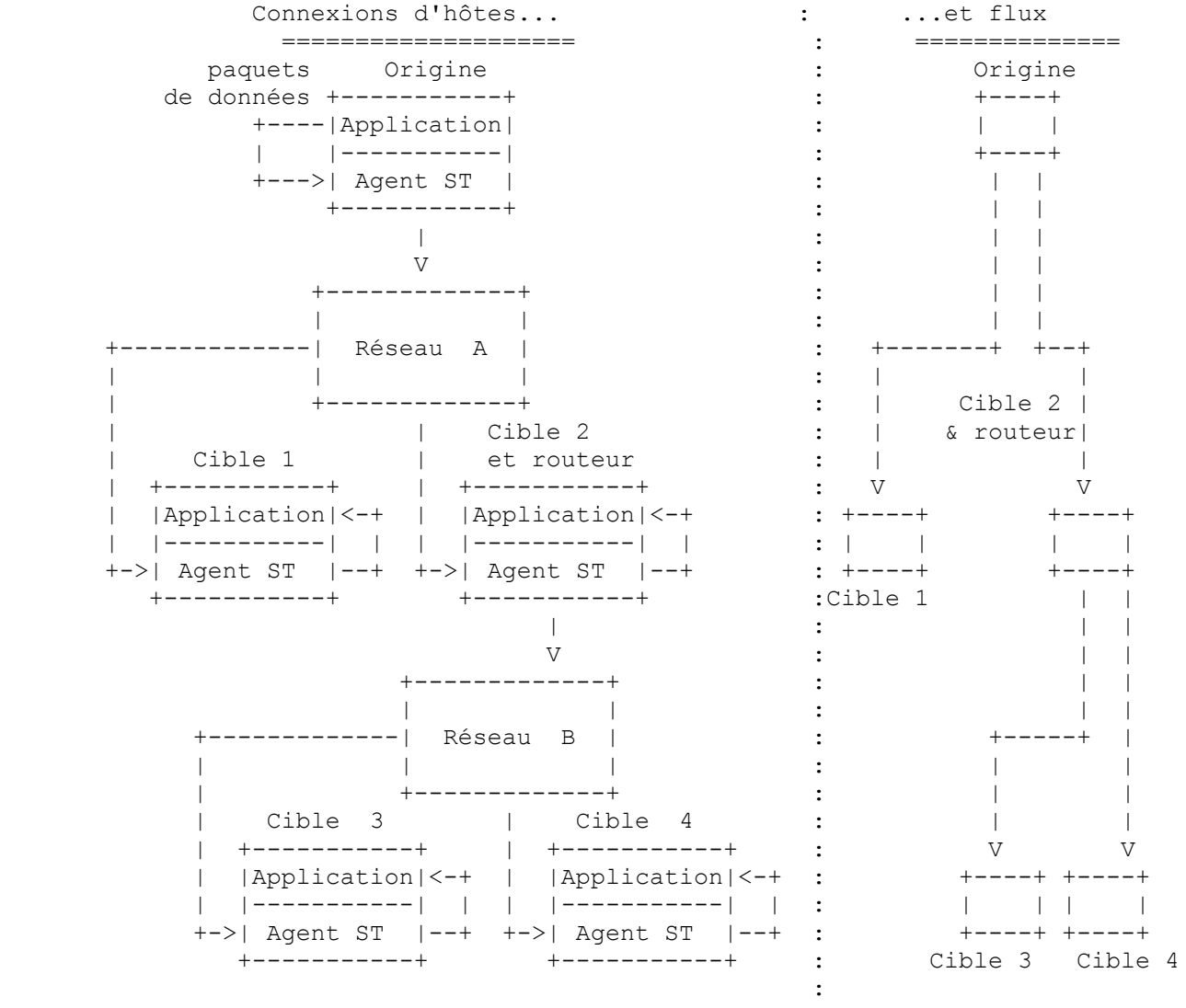


Figure 3 : Le concept de flux

1.5.1 Flux

Les flux forment le concept central de ST2. Ils sont établis entre une origine qui envoie et une ou plusieurs cibles qui reçoivent sous la forme d'une arborescence d'acheminement. Les flux sont unidirectionnels de l'origine aux cibles. Les nœuds dans l'arborescence représentent ce qu'on appelle les agents ST, entités qui exécutent le protocole ST2 ; dans cette arborescence, les

liaisons sont appelées des bonds. Tout nœud dans le milieu de l'arbre est appelé un agent intermédiaire, ou routeur. Un agent peut avoir n'importe quelle combinaison d'origine, cible, ou capacités intermédiaires.

La Figure 3 illustre un flux provenant d'une origine jusqu'à quatre cibles, où l'agent ST sur la cible 2 fonctionne aussi comme agent intermédiaire. Utilisons ce nœud Cible 2/routeur pour expliquer la terminologie de base de ST2 : la direction du flux depuis ce nœud vers les cibles 3 et 4 est appelée aval, la direction vers le nœud d'origine est l'amont. Les agents ST qui sont à un bond d'un nœud donné sont appelés bonds précédents vers l'amont, et bonds précédents vers l'aval.

L'entretien des flux se fait à l'aide des messages SCMP. Les messages SCMP typiques pour construire un flux sont CONNECT et ACCEPT, DISCONNECT et REFUSE pour clore un flux, CHANGE pour modifier la qualité de service associée à un flux, et JOIN pour demander à s'ajouter à un flux.

Chaque agent ST conserve les informations d'état décrivant les flux qui s'écoulent à travers lui. Il peut activement rassembler et distribuer de telles informations. Il peut reconnaître les agents ST voisins défaillants grâce à l'utilisation d'échanges périodiques de messages HELLO. Il peut demander aux autres agents ST l'état d'un flux particulier via un message STATUS. Ces agents ST renvoient alors un message STATUS-RESPONSE. Les messages NOTIFY peuvent être utilisés pour informer les autres agents ST d'événements significatifs.

ST2 offre une grande richesse de fonctionnalités pour la gestion des flux. Les flux peuvent être groupés pour minimiser les ressources allouées ou pour les traiter de la même façon en cas de défaillances. Durant les audioconférences, par exemple, seul un ensemble limité de participants peut parler à la fois. En utilisant le mécanisme de groupe, il est seulement nécessaire de réserver des ressources pour une portion des flux audio du groupe. En utilisant le même concept, un groupe entier de flux audio et vidéo en rapport peut être abandonné si l'un d'entre eux est préempté.

1.5.2 Transmission de données

Le transfert de données dans ST2 est unidirectionnel dans le sens aval. Le transport de données à travers des flux est très simple. ST2 met seulement un petit en-tête devant les données d'utilisateur. L'en-tête contient une identification de protocole qui distingue les paquets ST2 des paquets IP, un numéro de version ST2, un champ de priorité (qui spécifie l'importance relative des flux en cas de conflit), un compteur de longueur, une identification de flux, et une somme de contrôle. Ces éléments forment un en-tête de 12 octets.

L'efficacité est aussi réalisée en évitant la fragmentation et le réassemblage sur tous les agents. L'établissement de flux donne une taille maximum de message pour les paquets de données sur un flux. Cette taille maximum de message est communiquée aux couches supérieures, de sorte qu'elles fournissent à ST2 des paquets de données de taille convenable.

La communication avec les multiples prochains bonds peut être rendue encore plus efficace en utilisant la diffusion groupée de couche MAC lorsque elle est disponible. Si un sous réseau prend en charge la diffusion groupée, un seul paquet en diffusion groupée est suffisant pour atteindre tous les prochains bonds connectés à ce sous-réseau. Cela conduit à une réduction significative des exigences de bande passante d'un flux. Si la diffusion groupée n'est pas fournie, des paquets séparés doivent être envoyés à chaque prochain bond.

Comme ST2 s'appuie sur la réservation, il ne contient pas de mécanisme de correction d'erreurs pour les échanges de données comme celui qu'on trouve dans TCP. On suppose que les données en temps réel, comme l'audio et vidéo numérique, exigent seulement une livraison correcte partielle. Dans de nombreux cas, les paquets retransmis arriveraient trop tard pour satisfaire aux exigences de livraison du temps réel. Aussi, selon le codage des données et les applications particulières, un petit nombre d'erreurs d'un flux de données est acceptable. Dans tous les cas, la fiabilité peut être assurée par des couches au dessus de ST2 quand c'est nécessaire.

1.5.3 Spécification de flux

Au titre de l'établissement d'une connexion, SCMP prend en main la négociation des paramètres de qualité de service pour un flux. Dans la terminologie de ST2, ces paramètres forment une spécification de flux (FlowSpec) qui est associée au flux. Différentes versions de FlowSpecs existent, voir la [RFC1190], [DHHS92] et la [RFC1363], et ils peuvent être distingués par un numéro de version. Normalement, ils contiennent des paramètres tels qu'un débit moyen et maximum, un délai de bout en bout, la variance de délai d'un flux. SCMP lui-même ne fournit que le mécanisme de relais des paramètres de qualité de service.

Trois sortes d'entités participent à la négociation de la qualité de service : les entités d'application aux sites d'origine et de cibles comme utilisateurs du service, les agents ST, et le gestionnaire de ressource local (LRM). L'application d'origine fournit la FlowSpec initiale qui demande une qualité de service particulière. Chaque agent ST qui obtient la FlowSpec au titre d'un message d'établissement de connexion la présente au gestionnaire de ressource local. ST2 ne détermine pas comment les

gestionnaires de ressource font les réservations ni comment les ressources sont programmées suite à ces réservations ; cependant, ST2 suppose que ces mécanismes existent.
Un exemple de la procédure de négociation de FlowSpec est illustré à la Figure 4. Selon le succès de ses réservations locales, le LRM met à jour les champs FlowSpec et retourne la FlowSpec à l'agent ST, qui la passe en aval au titre du message de connexion. Finalement, la FlowSpec est communiquée à l'application à la cible qui peut fonder sur elle sa décision d'acceptation/rejet pour l'établissement de la connexion et peut finalement aussi modifier la FlowSpec. Si une cible accepte la connexion, la FlowSpec (éventuellement modifiée) est renvoyée à l'origine qui peut alors calculer une qualité de service globale pour toutes les cibles. L'entité d'application à l'origine peut ultérieurement demander un CHANGE pour ajuster les réservations.

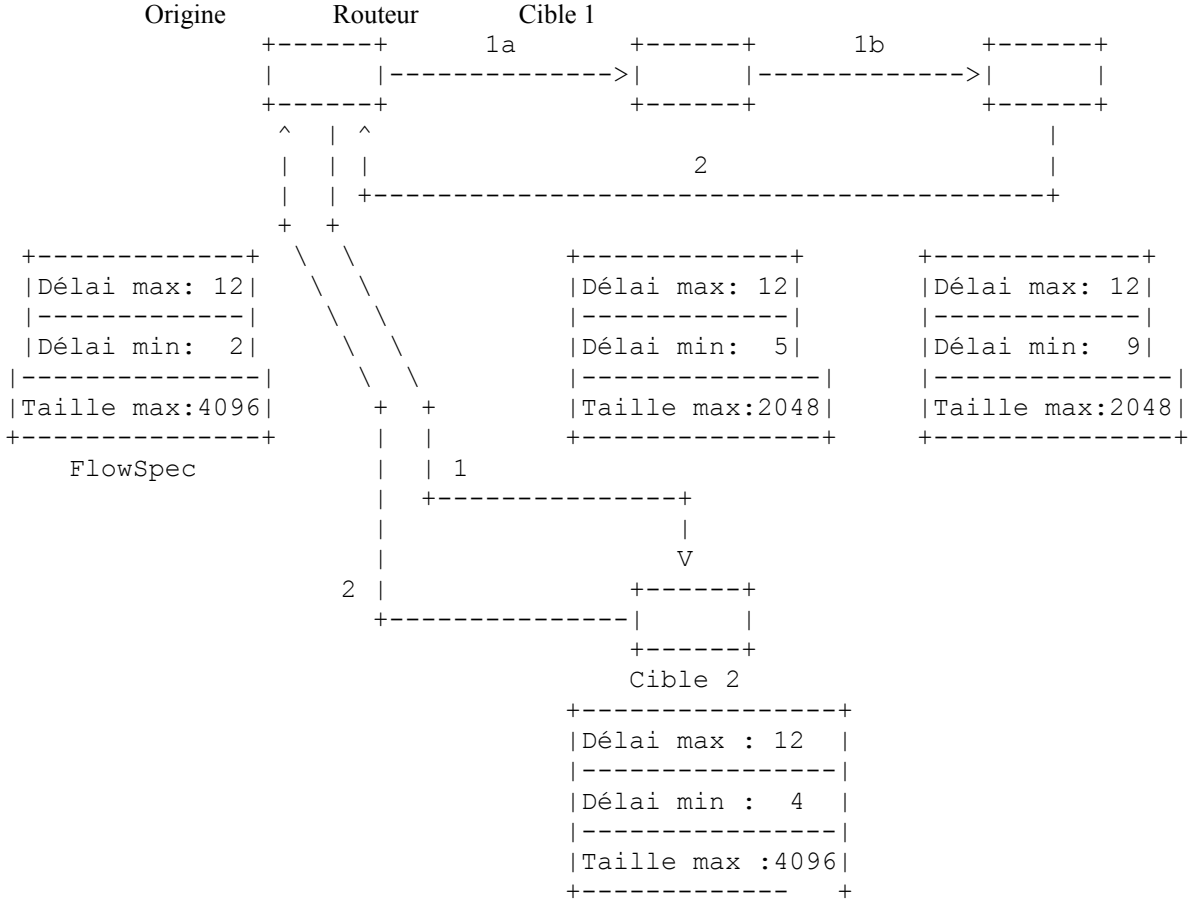


Figure 4 : Négociation de qualité de service avec FlowSpecs

1.6 Organisation du document

Le présent document contient la spécification de la version ST2+ du protocole ST2. Dans la suite de ce document, chaque fois que les termes "ST" ou "ST2" sont utilisés, il se réfèrent à la version ST2+ de ST2.

Le document est organisé comme suit :
- o la Section 2 décrit le servie d'utilisateur ST2 du point de vue d'une application.
- o la Section 3 illustre le protocole de transfert de données de ST2, ST.
- o la Section 4 jusqu'à la Section 8 spécifient le protocole d'établissement de ST2, SCMP.
- o la spécification du flux ST2 est présentée à la Section 9.
- o les formats des éléments de protocole et les PDU sont définis à la Section 10.

2. Description du service d'utilisateur ST2

Cette section décrit le service d'utilisateur ST du point de vue général d'une application. Il définit les opérations et les fonctions primitives du flux ST. Il spécifie quelles opérations sur les flux peuvent être invoquées par les applications construites par dessus ST et quand les fonctions primitives de ST peuvent être exécutées légalement. Noter que les primitives ST présentées ne spécifient pas une API. Elles ne sont utilisées ici que dans le seul but d'illustrer le modèle de service pour ST.

2.1 Opérations de flux et fonctions de primitives

Une application ST peut à l'origine créer, étendre, réduire, changer, envoyer des données et supprimer un flux. Lorsque un flux est étendu, de nouvelles cibles sont ajoutées au flux ; lorsque un flux est réduit, certaines des cibles actuelles sont abandonnées; Lorsque un flux est changé, la qualité de service associée est modifiée.

Une application ST à la cible peut se joindre au flux, en recevoir des données et le quitter. Ceci se traduit par les opérations de flux suivantes :

- o OPEN : crée un nouveau flux [origine], CLOSE : supprime le flux [origine],
- o ADD : étend le flux, c'est-à-dire, y ajoute de nouvelles cibles [origine],
- o DROP : réduit le flux, c'est-à-dire, y ôte des cibles [origine],
- o JOIN : se joindre à un flux [cible], LEAVE : quitte un flux [cible],
- o DATA : envoie des données à travers le flux [origine],
- o CHG : change la QS du flux [origine],

Chaque opération de flux peut exiger l'exécution de plusieurs fonctions primitives. Par exemple, pour ouvrir un nouveau flux, une demande est d'abord produite par l'envoyeur et une indication est générée chez un ou plusieurs receveurs ; puis, les receveurs peuvent chacun accepter ou refuser la demande et les indications correspondantes sont générées chez l'envoyeur. Un cas avec un seul receveur est donnée dans la Figure 5 ci-dessous.

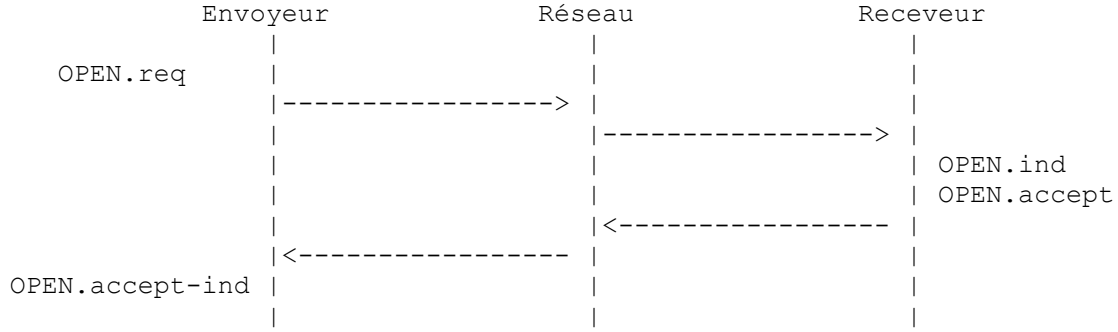


Figure 5 : Primitives pour l'opération de flux OPEN

Le tableau 1 définit les fonctions primitives de service ST associées à chaque opération de flux. La colonne marquée "O/C" indique si la primitive est exécutée à l'origine ou à la cible.

Primitive	Description	O/C|
OPEN.req	ouvre un flux	O
OPEN.ind	indication de demande de connexion	C
OPEN.accept	accepte le flux	C
OPEN.refuse	refuse le flux	C
OPEN.accept-ind	indication d'acceptation de connexion	O
OPEN.refuse-ind	indication de refus de connexion	O
ADD.req	ajoute des cibles au flux	O
ADD.ind	indication de demande d'ajout	C
ADD.accept	accepte le flux	C
ADD.refuse	refuse le flux	C
ADD.accept-ind	indication d'acceptation d'ajout	O
ADD.refuse-ind	indication de refus d'ajout	O
JOIN.req	se joindre à un flux	C
JOIN.ind	indication de demande de jonction	O
JOIN.reject	rejette une jonction	O
JOIN.reject-ind	indication de rejet de jonction	C
DATA.req	envoi de données	O
DATA.ind	indication de données reçues	C
CHG.req	changer la QS du flux	O
CHG.ind	indication de demande de changement	C
CHG.accept	changement accepté	C
CHG.refuse	changement refusé	C
CHG.accept-ind	indication d'acceptation de changement	O
CHG.refuse-ind	indication de refus de changement	O

DROP.req	abandon de cibles	O
DROP.ind	indication de déconnexion	C
LEAVE.req	quitte le flux	C
LEAVE.ind	indication que le flux est quitté	O
CLOSE.req	ferme le flux	O
CLOSE.ind	indication de fermeture du flux	C

Tableau 1 : Primitives de ST

2.2 Diagrammes d'état

Il n'est pas suffisant de définir l'ensemble des opérations de flux ST. Il est aussi nécessaire de spécifier quand les opérations peuvent être légalement exécutées. Pour cette raison, un ensemble d'états est maintenant introduit et les transitions d'un état à l'autre sont spécifiées. Des états sont définis par rapport à un seul flux. Les opérations de flux définies précédemment peuvent être légalement exécutées uniquement à partir d'un état approprié.

Un agent ST peut, par rapport à un flux ST, être dans un des états suivants :
- o IDLE : le flux n'a pas encore été créé.
- o PENDING : le flux est entré dans le processus d'établissement.
- o ACTIVE : le flux est établi et est actif.
- o ADDING : le flux est établi. Une expansion de flux est en cours.
- o CHGING : le flux est établi. Un changement de flux est en cours.

L'expérience antérieure de ST a conduit à limiter les opérations de flux qui peuvent être exécutées simultanément. Ces restrictions sont :

1. Une seule opération ADD ou CHG peut être traitée à la fois. Si un ADD ou CHG est déjà en cours, les demandes ultérieures sont mises en file d'attente par l'agent ST et ne sont traitées qu'après l'achèvement de l'opération précédente. Ceci s'applique aussi aux deux demandes ultérieures de la même sorte, par exemple, deux opérations ADD ou deux opérations CHG. La seconde opération n'est pas exécutée tant que la première n'est pas achevée.

2. Supprimer un flux, quitter un flux, ou abandonner des cibles d'un flux n'est possible qu'après que l'établissement du flux a été achevé. Un flux est considéré comme établi lorsque tous les prochains bonds de l'origine ont soit accepté soit refusé le flux. Noter que le refus de flux est automatiquement forcé après la fin de temporisation si aucune réponse ne vient d'un prochain bond.

3. Un agent ST ne transmet des données que le long des chemins déjà établis vers les cibles (voir aussi le paragraphe 3.1). Un chemin est considéré comme établi lorsque le prochain bons sur le chemin a explicitement accepté le flux. Cela implique que la cible et tous les autres agents ST intermédiaires sont prêts à traiter les paquets de données entrants. En aucun cas, un agent ST ne transmettra de données à un agent ST de prochain bons qui n'a pas explicitement accepté le flux. Pour être sûr que toutes les cibles reçoivent les données, une application devrait n'envoyer les données qu'après que tous les chemins ont été établis, c'est-à-dire, que le flux est établi.

4. Il est permis d'envoyer des données à partir des états CHGING et ADDING. Lorsque on envoie des données à partir de l'état CHGING, la qualité de service sur les cibles affectées par le changement devrait être supposée plus restrictive. Lorsque on envoie des données à partir de l'état ADDING, les cibles qui reçoivent les données comportent au moins toutes les cibles qui faisaient déjà partie du flux au moment où l'opération ADD a été invoquée.

Les règles introduites ci-dessus exigent que les agents ST mettent en file d'attente les demandes entrantes lorsque l'état en cours ne permet pas de les traiter immédiatement. Afin de préserver la sémantique, les agents ST doivent conserver l'ordre des demandes, c'est-à-dire, mettre en œuvre la méthode FIFO. Exceptionnellement, la demande CLOSE par l'origine, et la demande LEAVE de la cible peuvent être traitées immédiatement : dans ces cas, la file d'attente est supprimée et il est possible que les demandes en file d'attente ne soient pas traitées.

Le diagramme d'état suivant définit le service ST. Des diagrammes séparés sont présentés pour l'origine et les cibles.

Le symbole (a/r)* indique que toutes les cibles dans la liste des cibles ont explicitement accepté ou refusé le flux, ou que le refus a été forcé après expiration de la temporisation. Si la liste des cibles est vide, c'est-à-dire, qu'elle ne contient pas de cible, la condition (a/r)* est immédiatement satisfaite, le flux vide est créé et il entre dans l'état ESTBL.

Les primitives OPEN et ADD séparées à la cible ne sont que conceptuelles. La cible est en fait incapable de distinguer un OPEN et un ADD. Ceci est reflété à la Figure 7 et au Tableau 3 par la notation OPEN/ADD.

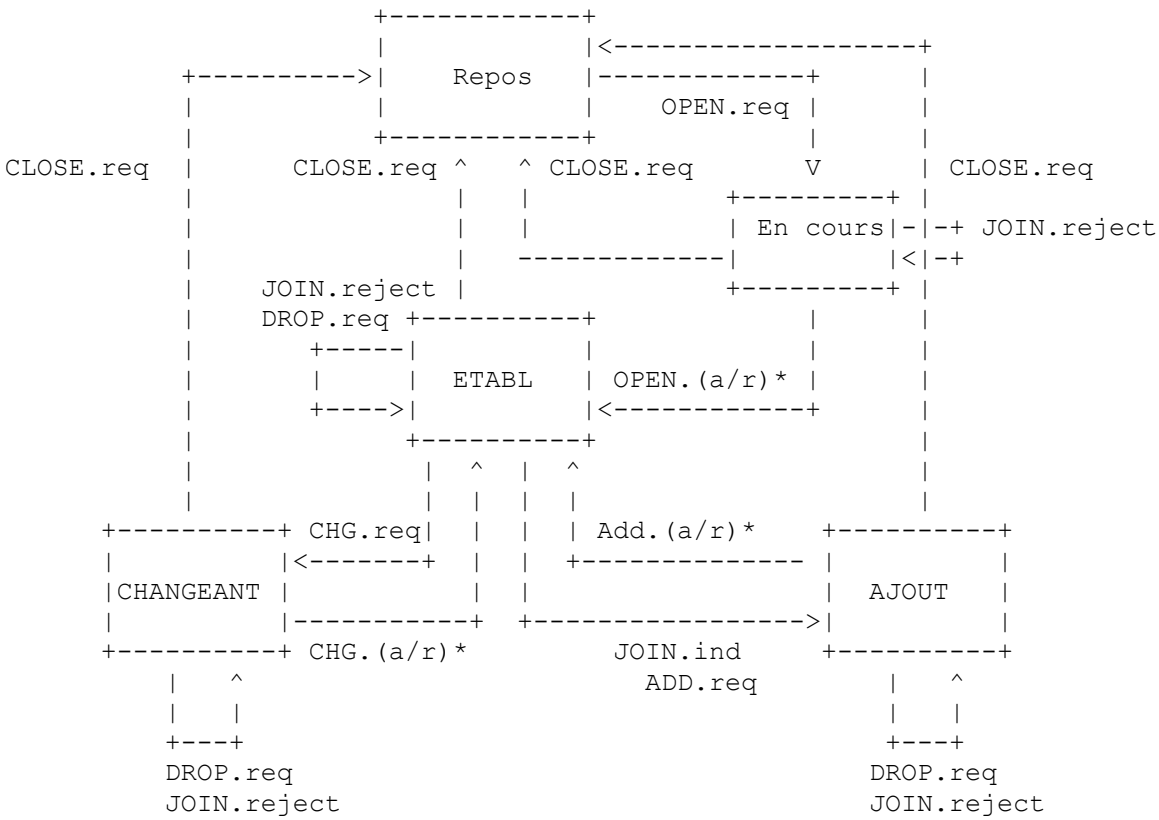


Figure 6 : Service ST à l'origine

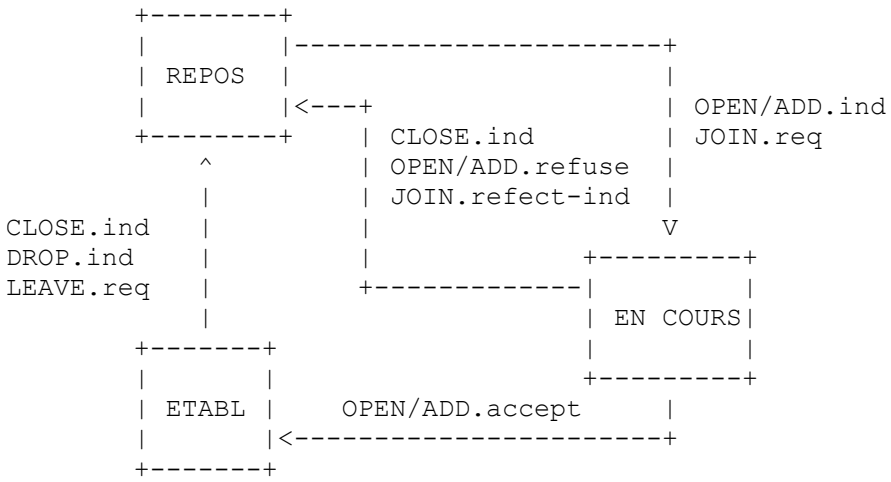


Figure 7 : Service ST à la cible

2.3 Tableaux de transition d'état

Les Tableaux 2 et 3 définissent quelles primitives peuvent être traitées à partir de quels états et les transitions d'état possibles.

Primitive	REPOS	EN COURS	ETABL	CHANGEANT	AJOUT
OPEN.req	ok	-	-	-	-
OPEN.accept-ind	-	si (a,r)*->ETAB	-	-	-
OPEN.refuse-ind	-	si (a,r)*->ETAB	-	-	-
ADD.req	-	file d'attente	->AJOUT	file d'attente	file d'attente
ADD.accept-ind	-	-	-	-	si (a,r)*
-	-	-	-	->ETABL	

ADD.refuse-ind	-	-	-	-	if(a,r)*
-	-	-	-	->ETABL	
JOIN.ind	-	file d'attente	->ADDING	file d'attente	file d'attente
JOIN.reject	-	OK	ok	ok	ok
DATA.req	-	-	ok	ok	ok
CHG.req	-	file d'attente	->CHGEANT	file d'attente	file d'attente
CHG.accept-ind	-	-	-	si (a,r)*	-
-	-	-	->ETABL	-	
CHG.refuse.ind	-	-	-	si (a,r)*	-
-	-	-	->ETABL	-	
DROP.req	-	-	ok	ok	ok
LEAVE.ind	-	OK	ok	ok	ok
CLOSE.req	-	OK	ok	ok	ok

Tableau 2 : Primitives et états à l'origine

Primitive	REPOS	EN COURS	ETABL
OPEN/ADD.ind	->EN COURS	-	-
OPEN/ADD.accept	-	->ETABL	-
OPEN/ADD.refuse	-	->REPOS	-
JOIN.req	->EN COURS	-	-
JOIN.reject-ind	-	-REPOS	-
DATA.ind	-	-	ok
CHG.ind	-	-	ok
CHG.accept	-	-	ok
DROP.ind	-	ok	ok
LEAVE.req	-	ok	ok
CLOSE.ind	-	ok	ok
CHG.ind	-	-	ok

Tableau 3 : Primitives et états à la cible

3. Protocole ST2 de transfert de données

Cette section présente le protocole de transfert de données ST2, ST. Le transfert des données est décrit au paragraphe 3.1, puis les fonctions du protocole de transfert des données sont illustrées au paragraphe 3.2.

3.1 Transfert de données avec ST

La transmission des données avec ST n'est pas fiable. Une application n'a pas la garantie que les données atteindront les destinations et ST ne fait aucune tentative de récupération des paquets perdus, par exemple, du fait du réseau sous-jacent. Cependant, si les données atteignent leur destination, il devrait agir conformément à la qualité de service associée au flux.

De plus, ST peut livrer des données corrompues dans la transmission. De nombreux types de données en temps réel, comme l'audio et la vidéo numérique, exigent seulement une livraison partiellement correcte. Dans de nombreux cas, les paquets retransmis arriveraient trop tard pour satisfaire à leurs exigences de livraison en temps réel. D'un autre côté, selon le codage des données et l'application particulière, un petit nombre d'erreurs dans le flux des données est acceptable. Dans tous les cas, la fiabilité peut être fournie par les couches au dessus de ST2 si nécessaire.

Aucune fragmentation des données n'est prise en charge durant la phase de transfert des données. L'application est supposée segmenter ses PDU de données conformément à la MTU minimum sur tous les chemins dans le flux. L'application reçoit des informations sur les MTU qui se rapportent aux chemins vers les cibles au titre du message ACCEPT (voir au paragraphe 8.6). La MTU minimum sur tous les chemins peut être calculée à partir des MTU relatives aux chemins individuels. Les agents ST éliminent en silence les paquets de données trop longs (voir aussi au paragraphe 5.1.1).

Un agent ST ne transmet les données que sur les chemins déjà établis vers les cibles. Un chemin est considéré comme établi une fois que l'agent ST du prochain bond sur le chemin a envoyé un message ACCEPT (voir au paragraphe 2.2). Cela implique que la cible et les autres agents ST intermédiaires sur le chemin vers la cible sont prêts à traiter les paquets de données entrants. En aucun cas un agent ST ne transmettra de données à un agent ST de prochain bond qui n'a pas explicitement accepté le flux.

Pour être raisonnablement sûr que toutes les cibles reçoivent les données avec la qualité de service désirée, une application devrait n'envoyer les données qu'après que le flux complet a été établi. Selon l'API locale, une application peut n'être pas empêchée d'envoyer des données avant l'achèvement de l'établissement du flux, mais elle devrait être consciente du fait que des données pourraient être perdues ou ne pas atteindre les cibles prévues. Ce comportement peut en fait être désirable pour des applications comme celles dont les cibles multiples peuvent chacune traiter aussitôt les données disponibles (par exemple, une conférence ou un jeu réparti).

Il est souhaitable que les mises en œuvre tirent parti des réseaux qui prennent en charge la diffusion groupée. Si un réseau ne prend pas en charge la diffusion groupée, ou dans le cas où les prochains bonds sont sur des réseaux différents, plusieurs copies du paquet de données doivent être envoyées.

3.2 Fonctions du protocole ST

Le protocole ST fournit deux fonctions :
- o l'identification du flux
- o la priorité des données

3.2.1 Identification de flux

Les paquets de données ST sont encapsulés par un en-tête ST qui contient l'Identifiant de flux (SID, *Stream IDentifier*). Ce SID est choisi à l'origine de telle sorte qu'il soit unique au monde sur l'Internet. Le SID doit être connu aussi du protocole d'établissement. Au moment de l'établissement du flux, le protocole d'établissement construit, chez chaque agent traversé par le flux, une entrée dans sa base de données locale qui contient les informations de flux. Le SID peut être utilisé comme référence dans cette base de données, pour obtenir rapidement les informations nécessaires de duplication et de transmission.

Les identifiants de flux sont destinés à être utilisés pour rendre plus efficaces la tâche de transmission des paquets. L'opération en temps critique est celle ou un agent ST intermédiaire reçoit un paquet de l'agent ST du bond précédent et le transmet aux agents ST du prochain bond.

Le format des PDU de données qui comportent le SID est défini au paragraphe 10.1. La génération des identifiants de flux est exposée au paragraphe 8.1.

3.2.2 Élimination de paquet sur la base de la priorité des données

ST fournit une qualité de service bien définie à ses applications. Cependant, il peut y avoir des cas où le réseau est temporairement encombré et où les agents ST doivent éliminer certains paquets pour minimiser l'impact global sur d'autres flux. Le protocole ST fournit un mécanisme pour éliminer les paquets de données sur la base du champ Priorité dans la PDU de données (voir au paragraphe 10.1). L'application alloue à chaque paquet de données un niveau de priorité à l'élimination, qui est porté dans le champ Priorité. Les agents ST tenteront d'éliminer d'abord les paquets de priorité inférieure durant les périodes d'encombrement du réseau. Les applications peuvent choisir d'envoyer des données sur plusieurs niveaux de priorité afin que les données les moins importantes puissent être éliminées en premier.

4. Description fonctionnelle de SCMP

Les agents ST créent et gèrent des flux à l'aide du protocole de message de contrôle de flux (SCMP, *ST Control Message Protocol*). Conceptuellement, SCMP se tient immédiatement au-dessus de ST (comme le fait ICMP au dessus de IP). SCMP suit un modèle de demande-réponse. Les messages SCMP sont rendus fiables par l'utilisation de la retransmission après expiration d'une temporisation.

Cette section contient une description fonctionnelle de la gestion de flux avec SCMP. Pour aider à préciser les échanges SCMP utilisés pour l'établissement et la conservation des flux ST, un exemple est inclus sur une topologie de réseau simple, représenté à la Figure 8. En utilisant les messages SCMP décrits dans cette section, il sera possible à une application ST :
- o de créer un flux de A vers les homologues en B, C et D,

- o d'ajouter un homologue en E,
- o d'abandonner les homologues B et C,
- o de laisser F se joindre au flux,
- o de supprimer le flux.

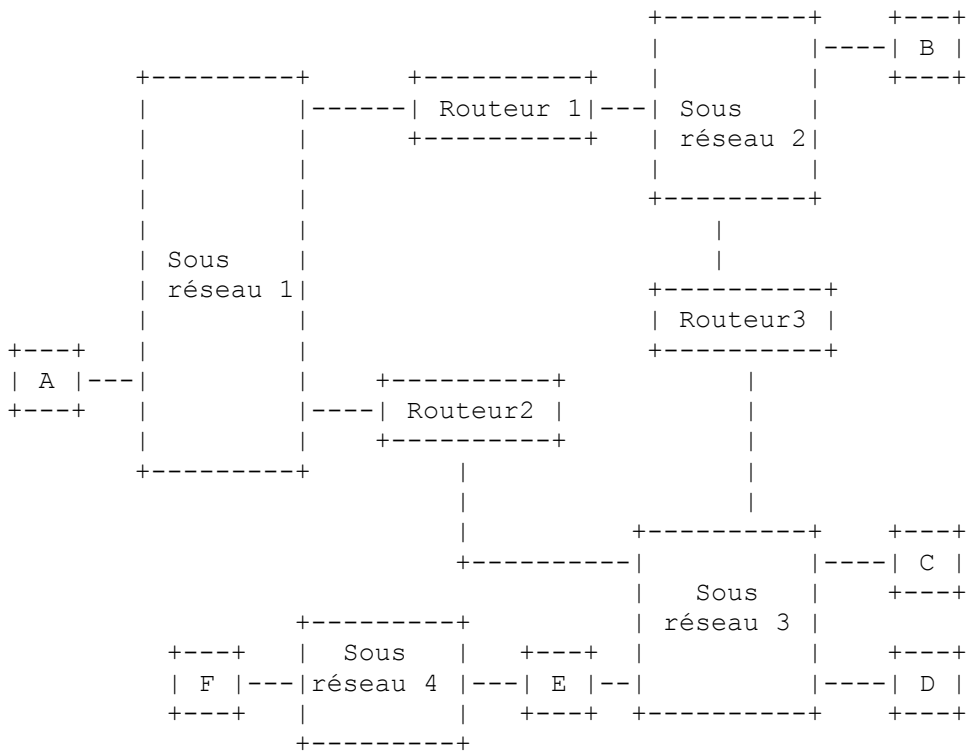


Figure 8 : Exemple de topologie d'un flux ST

On décrit d'abord les types de flux possibles au paragraphe 4.1 ; le paragraphe 4.2 introduit les types de messages de contrôle SCMP ; la fiabilité de SCMP est discutée au paragraphe 4.3 ; les options de flux sont traitées au paragraphe 4.4 ; l'établissement des flux est présenté au paragraphe 4.5 ; le paragraphe 4.6 illustre la modification d'un flux, y compris l'expansion, la réduction, les changements de qualité de service associée au flux. Enfin, la suppression de flux est traitée au paragraphe 4.7.

4.1 Types de flux

SCMP permet l'établissement et la gestion de différents types de flux. Les flux diffèrent dans la façon dont ils sont construits et dans celle dont les informations sont conservées sur les cibles connectées.

4.1.1 Construction de flux

Les flux peuvent être construits de façon orientée envoyeur, orientée receveur, ou avec une approche mixte :

- o Dans la façon orientée envoyeur, l'application à l'origine fournit à l'agent ST la liste des receveurs pour le flux. Les nouvelles cibles, s'il en est, sont aussi ajoutées à partir de l'origine.

- o Dans l'approche orientée receveur, l'application à l'origine crée les flux et ferme les flux qui ne contiennent pas de cible. Chaque cible se joint alors au flux de façon autonome.

- o Dans l'approche mixte, l'application à l'origine crée un flux qui contient des cibles et d'autres cibles se joignent au flux de façon autonome.

ST2 fournit des options de flux pour prendre en charge les approches de flux orienté envoyeur et mixte. Les flux orientés receveur peuvent être émulés grâce à l'utilisation de flux mixtes. La façon dont les cibles peuvent être ajoutées à un flux particulier est contrôlée selon des niveaux d'autorisation d'adhésion. Les niveaux d'autorisation d'adhésion sont décrits au paragraphe 4.4.2.

4.1.2 Connaissance des receveurs

Lorsque des flux sont construits de la façon orienté envoyeur, tous les agents ST auront les informations complètes sur toutes les cibles en aval d'un agent particulier. Dans ce cas, les informations de cible sont relayées vers l'aval d'agent à agent durant l'établissement du flux.

Lorsque les cibles s'ajoutent d'elles-mêmes à des flux d'approche mixte, les agents ST vers l'amont peuvent être informés ou non. La propagation des informations sur les cibles qui "se joignent" à un flux est aussi contrôlée via des niveaux d'autorisation d'adhésion. Comme mentionné précédemment, les niveaux d'autorisation d'adhésion sont décrits au paragraphe 4.4.2.

Ceci conduit à deux types de flux :

- Les informations complètes de cibles sont propagées dans un flux en état plein. Pour un tel flux, tous les agents sont avertis de toutes les cibles connectées au flux vers l'aval. il en résulte que les informations de cibles sont conservées à l'origine et chez les agents intermédiaires. Les opérations sur des cibles seules sont toujours possibles, c'est-à-dire, changer une certaine cible, ou sortir cette cible du flux. Il est toujours possible à tout agent ST de tenter la récupération de toutes les cibles vers l'aval.

- Dans les flux plus légers, il est possible que les agents d'origine et d'autres éléments vers l'amont n'aient aucune connaissance de certaines cibles. Il en résulte des états moins conservés et une gestion de flux plus facile, mais cela limite les opérations sur des cibles spécifiques. Des actions particulières peuvent être exigées pour la prise en charge des opérations de changement et d'abandon sur des cibles inconnues (voie au paragraphe 5.7). Aussi, la récupération de flux peut n'être pas possible. Bien sûr, des fonctions génériques comme la suppression de tout le flux sont encore possibles. Il est prévu que les applications qui auront un grand nombre de cibles vont utiliser des flux légers afin de limiter les états chez les agents et le nombre de cibles par message de contrôle.

Les flux à états pleins servent bien des applications comme la visioconférence ou les jeux répartis en ligne, dans lesquels il est important d'avoir la connaissance des receveurs connectés, par exemple, de limiter les participants. Les flux légers peuvent être exploités par les applications telles que la lecture à distance ou les applications de diffusion radio et télévision en différé dans lesquelles il n'est pas nécessaire que l'envoyeur connaisse les receveurs. Le paragraphe 4.4.2 définit les niveaux d'autorisation d'adhésion, qui prennent en charge deux types de flux à états pleins et un type de flux léger.

4.2 PDU de contrôle

SCMP définit les PDU suivantes (le principale objet de chaque PDU est aussi indiqué) :

1.	ACCEPT	accepter un nouveau flux
2.	ACK	accuser réception d'un message entrant
3.	CHANGE	changer la qualité de service associée à un flux
4.	CONNECT	établir un nouveau flux ou ajouter de nouvelles cibles à un flux existant
5.	DISCONNECT	retirer certaines cibles du flux, ou toutes
6.	ERROR	indiquer une erreur contenue dans un message entrant
7.	HELLO	détecter des défaillances des agents ST voisins
8.	JOIN	demander à se joindre au flux de la part d'une cible
9.	JOIN-REJECT	rejeter la demande d'une cible de se joindre au flux
10.	NOTIFY	informer un agent ST d'un événement significatif
11.	REFUSE	refuser l'établissement d'un nouveau flux
12.	STATUS	interroger un agent ST sur un flux spécifique
13.	STATUS-RESPONSE	répondre aux interrogations sur un flux spécifique

SCMP suit un modèle de demande-réponse dont toutes les demandes attendent des réponses. La retransmission après l'expiration d'une temporisation est utilisée pour les messages perdus ou ignorés. Les messages de contrôle ne s'étendent pas à travers les limites de paquet ; si un message de contrôle est trop grand pour la MTU à un bond, ses informations sont partagées et un message de contrôle par partie est envoyé, comme décrit au paragraphe 5.1.2.

On répond aux messages de demande CONNECT et CHANGE par des messages ACCEPT qui indiquent la réussite, et par des messages REFUSE qui indiquent l'échec. On répond aux messages JOIN soit par un message CONNECT qui indique la réussite, soit par un message JOIN-REJECT qui indique l'échec. Les cibles peuvent être retirées d'un flux soit par l'origine soit par la cible via les messages DISCONNECT et REFUSE.

Les messages ACCEPT, CHANGE, CONNECT, DISCONNECT, JOIN, JOIN-REJECT, NOTIFY et REFUSE doivent toujours recevoir des accusés de réception explicites :

- o avec un message ACK, si le message a été reçu correctement et si il a été possible d'analyser et extraire et interpréter correctement son en-tête, ses champs et paramètres,
- o avec un message ERROR, si une erreur de syntaxe a été détectée dans l'en-tête, les champs ou les paramètres inclus dans le message. La PDU erronée peut facultativement être retournée au titre du message ERROR. Un message ERROR indique seulement une erreur de syntaxe. Si d'autres erreurs sont détectées, il est nécessaire d'accuser d'abord réception avec ACK puis de prendre ensuite les actions appropriées. Par exemple, supposons un message CHANGE qui contient un SID inconnu : d'abord, il faut envoyer un message ACK, puis suit un message REFUSE avec le code de cause (SIDInconnu).

Si aucun message ACK ou ERROR n'est reçu avant que le temporisateur correspondant n'arrive à expiration, un échec de temporisation survient. La façon dont un agent ST devrait traiter les échecs de temporisation est décrite au paragraphe 5.2.

Les messages ACK, ERROR, et STATUS-RESPONSE ne reçoivent jamais d'accusé de réception.

Les messages HELLO sont un cas particulier. Si ils contiennent une erreur de syntaxe, un message ERROR devrait être généré en réponse. Autrement, aucun accusé de réception ni réponse ne devrait être généré. L'utilisation des messages HELLO est exposée au paragraphe 6.1.2.

Les messages STATUS qui contiennent une erreur de syntaxe devraient avoir en réponse un message ERROR. Autrement, un message STATUS-RESPONSE devrait être envoyé en réponse. L'utilisation de STATUS et de STATUS-RESPONSE est exposée au paragraphe 8.4.

4.3 Fiabilité de SCMP

SCMP est rendu fiable par l'utilisation de la retransmission lorsque une réponse n'est pas reçue à temps. Les messages ACCEPT, CHANGE, CONNECT, DISCONNECT, JOIN, JOIN-REJECT, NOTIFY et REFUSE doivent tous recevoir en réponse un message ACK (voir au paragraphe 4.2). En général, lors de l'envoi d'un message SCMP qui exige une réponse ACK, l'agent ST qui envoie doit régler le temporisateur Toxxxx (où xxxx est le type de message SCMP, par exemple, ToConnect). Si il ne reçoit pas un ACK avant l'arrivée à expiration du temporisateur Toxxxx, l'agent ST devrait retransmettre le message SCMP. Si aucun ACK n'a été reçu en Nxxxx retransmissions, une condition de fin de temporisation SCMP survient et l'agent ST entre dans son état de récupération de fin de temporisation SCMP. Les actions effectuées par l'agent ST en résultat de la condition de fin de temporisation SCMP diffèrent selon les différents messages SCMP et sont décrites au paragraphe 5.2.

Pour certains messages SCMP (CONNECT, CHANGE, JOIN et STATUS) l'agent ST qui envoie attend aussi une réponse en retour (ACCEPT/REFUSE, CONNECT/JOIN- REJECT) après la réception de ACK. Pour ces cas, l'agent ST doit lancer le temporisateur ToxxxxResp dès la réception du ACK. (Comme précédemment, xxxx est le type de message SCMP initiateur, par exemple, ToConnectResp). Si il ne reçoit pas la réponse appropriée en retour à l'expiration de ToxxxxResp, l'agent ST met à jour son état et effectue les actions de récupération appropriées décrites au paragraphe 5.2. Les constantes suggérées sont données au paragraphe 10.5.4.

L'algorithme de fin de temporisation et de retransmission dépend de la mise en œuvre et sort du domaine d'application du présent document. La plupart des algorithmes existants se fondent sur une estimation du délai d'aller-retour (RTT, *Round Trip Time*) entre les deux agents. SCMP contient donc un mécanisme (voir au paragraphe 8.5) pour estimer ce RTT. Noter que les noms des variables qui se rapportent à la fin de temporisation décrits ci-dessus ne servent que de référence et les développeurs peuvent choisir de combiner certaines variables.

4.4 Options de flux

Une application peut choisir entre plusieurs options de flux. Les options désirées sont indiquées à l'agent ST de l'origine lorsque un nouveau flux est créé. Les options s'appliquent à des flux seuls et sont valides durant toute la durée de vie du flux. Les options choisies par l'application à l'origine sont incluses dans le message CONNECT initial (voir au paragraphe 4.5.3). Lorsque un message CONNECT atteint une cible, l'application à la cible est notifiée des options du flux qui ont été choisies (voir au paragraphe 4.5.5).

4.4.1 Pas de récupération

Lorsque une défaillance de flux est détectée, un agent ST va normalement tenter une récupération de flux, comme décrit au paragraphe 6.2. L'option NoRecovery est utilisée pour indiquer aux agents ST de ne pas tenter la récupération du flux. Le comportement du protocole dans le cas où l'option NoRecovery a été choisie est illustré au paragraphe 6.2. L'option NoRecovery est spécifiée en établissant le bit S dans le message CONNECT (voir au paragraphe 10.4.4). Le bit S ne peut être mis que par l'origine et il n'est jamais modifié par les agents ST intermédiaires et de cible.

4.4.2 Niveau d'autorisation d'adhésion

Lorsque un nouveau flux est créé, il est nécessaire de définir le niveau d'autorisation d'adhésion associé au flux. Ce niveau détermine le comportement du protocole dans le cas d'une adhésion au flux (voir aux paragraphes 4.1 et 4.6.3). Le niveau d'autorisation d'adhésion pour un flux est défini par le bit J et le bit N dans l'en-tête du message CONNECT (voir au paragraphe 10.4.4). Un des niveaux d'autorisation d'adhésion suivants doit être choisi :

- Niveau 0 - Refuse l'adhésion (JN = 00) : Aucune cible n'est admise à se joindre au flux.
- Niveau 1 - OK, Notifier l'origine (JN = 01) : Les cibles peuvent se joindre au flux. L'origine reçoit notification de l'adhésion de la cible.
- Niveau 2 - OK (JN = 10) : Les cibles peuvent se joindre au flux. Aucune notification n'est envoyée à l'origine du flux.

Certaines applications peuvent choisir de conserver un contrôle étroit sur leurs flux et ne permettront aucune connexion sans la permission de l'origine. Pour de tels flux, les applications cibles peuvent demander leur adhésion en envoyant une demande hors bande, c'est-à-dire, via l'IP normal, à l'origine. L'origine, si c'est son choix, peut alors ajouter la cible suivant le processus décrit au paragraphe 4.6.1.

Le niveau d'autorisation choisi a un impact sur le traitement du flux et sur l'état qui est conservé pour ce flux, comme décrit au paragraphe 4.1.

4.4.3 Enregistrement de chemin

L'option RecordRoute peut être utilisée pour demander que le chemin entre l'origine et une cible soit enregistré et livré à l'application. Cette option peut être utilisée pendant la connexion, l'acceptation, le changement, ou le refus d'un flux. Le résultat d'une option RecordRoute demandée par l'origine, c'est-à-dire, au titre des messages CONNECT ou CHANGE, est livré à la cible. Le résultat d'une option RecordRoute demandée par la cible, c'est-à-dire, au titre des messages ACCEPT ou REFUSE, est livré à l'origine.

L'option RecordRoute est spécifiée en ajoutant le paramètre RecordRoute aux messages SCMP mentionnés. Le format du paramètre RecordRoute est indiqué au paragraphe 10.3.5. Lorsque il ajoute ce paramètre, l'agent ST de l'origine doit déterminer le nombre d'entrées qui peuvent être enregistrées comme expliqué au paragraphe 10.3.5.

4.4.4 Données d'utilisateur

L'option Données d'utilisateur *(UserData)* peut être utilisée par les applications pour transporter des données spécifiques de l'application avec des messages de contrôle SCMP. Cette option peut être incluse dans les messages ACCEPT, CHANGE, CONNECT, DISCONNECT et REFUSE. Le format du paramètre Données d'utilisateur est indiqué au paragraphe 10.3.7. Cette option peut être incluse par l'origine, ou par la cible, en ajoutant le paramètre Données d'utilisateur aux messages SCMP mentionnés. Cette option ne peut être incluse qu'une seule fois par message SCMP.

4.5 Établissement de flux

Ce paragraphe présent une description de l'établissement de flux. Par souci de simplicité, on supposera que tout réussit, par exemple, toutes les ressources demandées sont disponibles, les messages sont correctement livrés, et l'acheminement est correct. Les défaillances possibles dans la phase d'établissement sont traitées au paragraphe 5.2.

4.5.1 Information provenant de l'application

Avant que l'établissement de flux ne puisse commencer, l'application doit collecter les informations nécessaires pour déterminer les caractéristiques de la connexion. Cela inclut d'identifier les participants et de sélectionner les paramètres de QS du flux de données. Les informations passées à l'agent ST par l'application incluent :

- la liste des cibles du flux (paragraphe 10.3.6). La liste peut être vide (paragraphe 4.5.3.1),
- la spécification du flux qui contient la qualité de service désirée pour le flux (Section 9),
- les informations sur les groupes dont le flux est membre, s'il en est (Section 7),
- les informations sur les options choisies pour le flux (paragraphe 4.4).

4.5.2 Établissement initial à l'origine

L'agent ST à l'origine effectue ensuite les opérations suivantes :

- allouer un identifiant de flux (SID) au flux (paragraphe 8.1),

- o invoquer la fonction d'acheminement pour déterminer l'ensemble des prochains bonds du flux (paragraphe 4.5.2.1),
- o invoquer le gestionnaire de ressource local (LRM) pour réserver les ressources (paragraphe 4.5.2.2),
- o créer des entrées de base de données locales pour mémoriser les informations sur le nouveau flux,
- o propager la demande de création de flux aux prochains bonds déterminés par la fonction d'acheminement (paragraphe 4.5.3).

4.5.2.1 Invocation de la fonction d'acheminement

Un agent ST qui est en train d'établir un flux invoque la fonction d'acheminement pour trouver le prochain bond à atteindre chacune des cibles spécifiées par la liste des cibles fournies par l'application. Ceci est similaire à la décision d'acheminement dans IP. Cependant, dans ce cas, le chemin est une multitude de cibles avec des exigences de QS plutôt qu'une seule destination.

Le résultat de la fonction d'acheminement est un ensemble d'agents ST de prochain bond. L'ensemble des prochains bonds choisis par la fonction d'acheminement n'est pas nécessairement le même que l'ensemble des prochains bonds que IP choisirait étant donné un certain nombre de datagrammes IP indépendants pour les mêmes destinations. L'algorithme d'acheminement peut essayer d'optimiser des paramètres autres que le nombre de bonds qu'il faudra aux paquets, tels que le délai, la consommation de bande passante du réseau local, ou la consommation totale de bande passante internet. Autrement, l'algorithme d'acheminement peut utiliser une simple recherche de chemin pour chaque cible.

Une fois qu'un prochain bond est choisi par la fonction d'acheminement, il persiste pour toute la durée de vie du flux, à moins que ne survienne une défaillance du réseau.

4.5.2.2 Réservation de ressources

L'agent ST invoque le gestionnaire de ressource locale (LRM) pour effectuer les réservations appropriées. L'agent ST présente au LRM des informations qui incluent :

- o la spécification de flux avec la qualité de service souhaitée pour le flux (Section 9),
- o le numéro de version associé à la spécification du flux (Section 9).
- o les informations sur les groupes dont est membre le flux, s'il en est (Section 7),

La spécification de flux contient les informations dont le LRM a besoin pour allouer les ressources. Le LRM met à jour les informations contenues dans la spécification du flux avant de la retourner à l'agent ST. Le paragraphe 9.2.3 définit les champs de la spécification de flux que met à jour le LRM.

L'adhésion à un flux dans un groupe peut affecter la quantité de ressources que le LRM a à allouer, voir la Section 7.

4.5.3 Envoi du message CONNECT

L'agent ST envoie un message CONNECT à chacun des agents ST de prochain bond identifiés par la fonction d'acheminement. Chaque message CONNECT contient le SID, les options de flux choisies, la FlowSpec, et une liste de cibles *(TargetList)*. Le format du message CONNECT est défini au paragraphe 10.4.4. En général, la FlowSpec et la Liste des cibles dépendent à la fois du prochain bond et du réseau qui intervient. Chaque liste des cibles est un sous-ensemble de la liste des cibles de l'origine, qui identifie les cibles qui seront atteintes à travers le prochain bond et à qui le message CONNECT est à envoyer.

La liste des cibles peut être vide, voir au paragraphe 4.5.3.1 ; si la liste des cibles cause la génération d'un message CONNECT trop long, ce message est partagé comme expliqué au paragraphe 5.1.2. Si plusieurs prochains bonds sont à atteindre à travers un réseau qui prend en charge la diffusion groupée de niveau réseau, un message CONNECT différent doit néanmoins être envoyée à chacun des prochains bonds car chacun va avoir une liste des cibles différente.

4.5.3.1 Liste de cibles vide

Une application à l'origine peut demander que l'agent ST local crée un flux vide. Il le fait en passant une Liste des cibles vide à l'agent ST local durant l'établissement initial du flux. Lorsque l'agent ST local reçoit une demande de création d'un flux vide, il alloue l'identifiant de flux (SID), met à jour ses entrées de base de données locale pour mémoriser les informations sur le nouveau flux et notifie à l'application que l'établissement du flux est achevé. L'agent ST local ne génère aucun message CONNECT pour les flux avec une Liste des cibles vide. Des cibles peuvent être ajoutées ultérieurement par l'origine (voir au paragraphe 4.6.1) ou elles peuvent se joindre au flux de façon autonome (voir au paragraphe 4.6.3).

4.5.4 Traitement de CONNECT par un agent ST intermédiaire

Un agent ST qui reçoit un message CONNECT, supposant qu'il ne comporte pas d'erreurs, répond au bond précédent par un ACK. Le message ACK doit identifier le message CONNECT auquel il correspond en incluant le numéro de référence indiqué par le champ Référence du message CONNECT. L'agent ST intermédiaire appelle la fonction d'acheminement, invoque le LRM pour réserver des ressources, puis propage les messages CONNECT à ses prochains bonds, comme décrit aux

paragraphes précédents.

4.5.5 Traitement de CONNECT aux cibles

Un agent ST qui est la cible d'un message CONNECT, supposant qu'il n'y a pas d'erreurs, répond au bond précédent par un ACK. L'agent ST invoque le LRM pour réserver les ressources locales puis interroge le processus d'application spécifié pour savoir si elle veut ou non accepter la connexion.

Les paramètres provenant du message CONNECT sont présentés à l'application, y compris le SID, les options de flux choisies, Origine, FlowSpec, Liste des cibles, et Groupe, s'il en est, pour qu'ils servent de base de décision. L'application est identifiée par une combinaison du champ NextPcol, provenant du paramètre Origine, et du champ point d'accès de service (SAP) inclus dans la cible correspondante (généralement la seule qui reste) de la liste des cibles. Le contenu du champ SAP peut spécifier l'identifiant d'accès ou autre à utiliser par la couche de protocole au dessus de la couche ST de l'hôte. Les paquets de données ultérieurement reçus porteront le SID, qui peut être transposé dans ces informations et être utilisé pour leur livraison.

Finalement, sur la base de la décision de l'application, l'agent ST envoie au bond précédent d'où le message CONNECT a été reçu un message soit ACCEPT soit REFUSE. Comme le bond précédent doit accuser réception du message ACCEPT (ou REFUSE), il lui est alloué un nouveau numéro de référence qui sera retourné dans le ACK. Le message CONNECT auquel le ACCEPT (ou REFUSE) est la réponse est identifié en plaçant le numéro de référence du message CONNECT dans le champ LnkReference du ACCEPT (ou REFUSE). Le message ACCEPT contient la FlowSpec comme acceptée par l'application à la cible.

4.5.6 Traitement de ACCEPT par un agent ST intermédiaire

Lorsque un agent ST intermédiaire reçoit un message ACCEPT, il vérifie d'abord que le message est une réponse à un CONNECT antérieur. Sinon, il répond à l'agent ST de prochain bond par un message ERROR, avec le code de cause (LnkRefUnknown). Autrement, il répond à l'agent ST du prochain bond par un ACK, et propage le message ACCEPT individuel au bond précédent le long du même chemin tracé par le CONNECT mais en direction inverse vers l'origine.

La FlowSpec est incluse dans le message ACCEPT de telle sorte que les agents ST d'origine et intermédiaires obtiennent l'accès aux informations qui ont été accumulées lorsque le message CONNECT a traversé l'internet. Noter que les ressources, comme spécifiées dans la FlowSpec du message ACCEPT, peuvent différer des ressources qui ont été réservées lorsque le CONNECT a été originellement traité. Donc, l'agent ST présente au LRM la FlowSpec incluse dans le message ACCEPT. On s'attend à ce que chaque LRM ajuste les réservations locales en libérant toutes les ressources excédentaires. Le LRM peut choisir de ne pas ajuster les réservations locales lorsque cet ajustement pourrait résulter en la perte de ressources nécessaires. Il peut aussi choisir d'attendre pour ajuster les ressources allouées jusqu'à ce que toutes les cibles en transit aient été acceptées ou refusées.

Dans le cas où l'agent ST intermédiaire agit comme origine par rapport à cette cible (voir au paragraphe 4.6.3.1) le message ACCEPT n'est pas propagé vers l'amont.

4.5.7 Traitement de ACCEPT par l'origine

L'origine va finalement recevoir un message ACCEPT (ou REFUSE) de chacune des cibles. Lorsque tous les ACCEPT sont reçus, l'application est notifiée des cibles et des ressources qui ont été bien allouées le long du chemin vers elle, comme spécifié dans la FlowSpec contenue dans le message ACCEPT. L'application peut alors utiliser les informations pour adopter ou terminer la portion de flux vers chaque cible.

Lorsque un ACCEPT est reçu par l'origine, le chemin vers la cible est considéré comme établi et l'agent ST est autorisé à transmettre les données le long de ce chemin comme expliqué à la Section 2 et au paragraphe 3.1.

4.5.8 Traitement de REFUSE par l'agent ST intermédiaire

Si une application d'une cible ne souhaite pas participer au flux, elle renvoie un message REFUSE à l'origine avec le code de cause (ApplDisconnect). Un agent ST intermédiaire qui reçoit un message REFUSE avec le code de cause (ApplDisconnect) en accuse réception en envoyant un ACK au prochain bond, invoque le LRM pour ajuster les réservations comme approprié, supprime l'entrée de cible de la base de données interne, et propage le message REFUSE en retour à l'agent ST du bond précédent.

Dans le cas où l'agent ST intermédiaire agit comme origine par rapport à cette cible (voir au paragraphe 4.6.3.1) le message REFUSE est seulement propagé vers l'amont lorsqu'il n'y a plus d'agent vers l'aval participant au flux. Dans ce cas, l'agent

indique que l'agent est à retirer du flux en propageant le message REFUSE avec le bit G établi (à 1).

4.5.9 Traitement de REFUSE par l'origine

Lorsque le message REFUSE atteint l'origine, l'agent ST de l'origine envoie un ACK et notifie à l'application que la cible ne fait plus partie du flux et également si le flux n'a plus de cibles restantes. Si il ne reste plus de cibles, l'application peut souhaiter terminer le flux, ou garder le flux actif pour permettre l'ajout de cibles ou de flux comme décrit au paragraphe 4.6.3.

4.5.10 Autres fonctions durant l'établissement de flux

Certaines autres fonctions doivent être accomplies par un agent ST lorsque les messages CONNECT voyagent vers l'aval et que les messages ACCEPT (ou REFUSE) voyagent vers l'amont durant la phase d'établissement du flux. Elles n'ont pas été mentionnées dans les paragraphes précédents pour rendre l'exposé aussi simple que possible. Ces fonctions incluent :

- Le calcul de la taille de l'unité maximum de transmission sur les chemins des cibles, au titre du mécanisme de découverte de la MTU présenté au paragraphe 8.6. Ceci est fait en mettant à jour le champ MaxMsgSize du message CONNECT (voir au paragraphe 10.4.4). Cette valeur est rapportée à l'origine dans le champs MaxMsgSize du message ACCEPT (voir au paragraphe 10.4.1).

- Le compte du nombre de nuages IP à traverser pour atteindre chacune des cibles, s'il en est. Les nuages IP sont traversés lorsque le mécanisme d'encapsulation IP est utilisé. Ce mécanisme est décrit au paragraphe 8.7. Les agents qui encapsulent mettent à jour le champ Bonds IP du message CONNECT (voir au paragraphe 10.4.4). La valeur résultante est rapportée à l'origine dans le champ Bonds IP du message ACCEPT (voir au paragraphe 10.4.1).

- La mise à jour de la valeur Temporisateur de récupération pour le flux sur la base de ce que l'agent peut prendre en charge. Ceci fait partie du mécanisme de récupération de flux du paragraphe 6.2. Ceci est fait en mettant à jour le champ Temporisateur de récupération du message CONNECT (voir au paragraphe 10.4.4). Cette valeur est rapportée à l'origine dans le champ Temporisateur de récupération du message ACCEPT (voir au paragraphe 10.4.1).

4.6 Modification d'un flux existant

Certaines applications peuvent souhaiter modifier un flux après qu'il ait été créé. Les changements possibles incluent l'expansion d'un flux, sa réduction, et le changement de sa FlowSpec. L'origine peut ajouter ou retirer des cibles, comme décrit aux paragraphes 4.6.1 et 4.6.2. Les cibles peuvent demander à se joindre au flux, comme décrit au paragraphe 4.6.3, ou elles peuvent décider de quitter un flux, comme décrit au paragraphe 4.6.4. Le paragraphe 4.6.5 explique comment changer la FlowSpec d'un flux.

Comme défini à la Section 2, un agent ST ne peut traiter qu'une seule modification de flux à la fois. Si une opération de modification d'un flux est déjà en cours, les autres demandes sont mises en file d'attente et traitées lorsque les opérations précédentes ont été achevées. Ceci s'applique aussi aux deux demandes suivantes de la même espèce, par exemple, deux changements successifs à la FlowSpec.

4.6.1 Ajout de nouvelles cibles par l'origine

Il est possible qu'une application à l'origine ajoute de nouvelles cibles à un flux existant à tout moment après l'établissement du flux. Avant que de nouvelles cibles ne soient ajoutées, l'application doit collecter les informations nécessaires sur les nouvelles cibles. De telles informations sont passées à l'agent ST de l'origine.

L'agent ST de l'origine produit un message CONNECT qui contient le SID, la FlowSpec, et la Liste des cibles qui spécifie les nouvelles cibles. Ceci est similaire à l'envoi d'un message CONNECT durant l'établissement du flux, avec les exceptions suivantes : l'origine vérifie que a) le SID est valide, b) les cibles ne sont pas déjà membres du flux, c) que le LRM évalue la FlowSpec de la nouvelle cible comme étant la même que la FlowSpec du flux existant, c'est-à-dire, elle exige qu'une quantité égale ou inférieure de ressources soit allouée. Si la FlowSpec de la nouvelle cible ne correspond pas à la FlowSpec du flux existant, une erreur est générée avec le code de cause (FlowSpecMismatch). Les fonctions pour comparer les spécifications de flux sont fournies par le LRM (voir au paragraphe 1.4.5).

Un agent ST intermédiaire qui est déjà un participant au flux regarde le SID et le StreamCreationTime *(heure de création du flux)*, et vérifie que le flux est le même. Il vérifie ensuite si l'intersection de la liste des cibles et des cibles du flux établi est vide. Si ce n'est pas le cas, il répond par un message REFUSE avec le code de cause (TargetExists) qui contient une liste des cibles qui sont dupliquées. Pour indiquer que le flux existe et inclure les cibles de la liste, l'agent ST règle à un (1) le bit E du

message REFUSE (voir au paragraphe 10.4.11). L'agent procède alors au traitement de chaque nouvelle cible de la Liste des cibles.

Pour chaque nouvelle cible de la Liste des cibles, le traitement est tout à fait le même que pour le CONNECT original. Le CONNECT reçoit un accusé de réception, est propagé et les ressources réseau sont réservées. Les agents ST intermédiaires ou de cible qui ne sont pas déjà des participants au flux se comportent comme dans le cas de l'établissement du flux (voir aux paragraphes 4.5.4 et 4.5.5).

4.6.2 Retrait d'une cible par l'origine

Il est possible qu'une application à l'origine retire des cibles existantes d'un flux à tout moment après que les cibles ont accepté le flux. L'application à l'origine spécifie l'ensemble des cibles à retirer et informe l'agent ST local. Sur la base de ces informations, l'agent ST envoie des messages DISCONNECT avec le code de cause (ApplDisconnect) aux prochains bonds qui se rapportent aux cibles.

Un agent ST qui reçoit un message DISCONNECT doit accuser réception en envoyant un ACK au bond précédent. L'agent ST met à jour son état et notifie le LRM de la suppression de la cible afin qu'il puisse modifier les réservations comme approprié. Lorsque le message DISCONNECT atteint la cible, l'agent ST notifie aussi à l'application que la cible ne fait plus partie du flux. Lorsque il ne reste plus de cibles qui puissent être atteintes à travers ce prochain bond particulier, l'agent ST informe le LRM et il supprime le prochain bond de son ensemble des prochains bonds.

SCMP fournit aussi un mécanisme d'arrosage pour supprimer les cibles qui se sont jointes au flux sans le notifier à l'origine. Le cas particulier de la suppression de cibles via l'arrosage est décrit au paragraphe 5.7.

4.6.3 Une cible se joint à un flux

Une application peut demander à se joindre à un flux existant. Il doit collecter des informations sur le flux incluant l'identifiant du flux (SID) et l'adresse IP de l'origine du flux. Cela peut être fait hors bande, par exemple, via l'IP normal. Les informations sont alors passées à l'agent ST local. L'agent ST génère un message JOIN qui contient la demande de l'application de se joindre au flux et il l'envoie vers l'origine du flux.

Un agent ST qui reçoit un a message JOIN, supposant qu'il n'y a pas d'erreurs, répond par un ACK. Le message ACK doit identifier le message JOIN auquel il correspond en incluant le numéro de référence indiqué par le champ Référence du message JOIN. Si l'agent ST n'est pas traversé par le flux qui doit être rejoint, il propage le message JOIN vers l'origine du flux. Une fois que le message JOIN a reçu un accusé de réception, les agents ST ne conservent aucune information d'état se rapportant au message JOIN.

Finalement, un agent ST traversé par le flux ou de l'origine du flux elle-même est atteint. Cet agent doit d'abord répondre à un message JOIN reçu par un ACK à l'agent ST duquel le message a été reçu, puis, il produit soit un message CONNECT soit un message JOIN-REJECT et l'envoie vers la cible. La réponse à la demande d'adhésion se fonde sur le niveau d'autorisation d'adhésion associé au flux (voir au paragraphe 4.4.2) :

- Si le flux a le niveau d'autorisation n° 0 (refus d'adhésion) : l'agent ST envoie un message JOIN-REJECT vers la cible avec le code de cause (JoinAuthFailure).

- Si le flux a le niveau d'autorisation n° 1 (ok, notifier l'origine) : l'agent ST envoie un message CONNECT vers la cible avec une liste de cibles incluant la cible qui demandait à se joindre au flux.

  Cela aboutit finalement à ajouter la cible au flux. Lorsque l'agent ST reçoit le message ACCEPT qui indique que la nouvelle cible a été ajoutée, il ne propage pas le message ACCEPT vers l'arrière (paragraphe 4.5.6). Il produit plutôt un message NOTIFY avec le code de cause (TargetJoined) de sorte que les agents amont, y compris l'origine, puissent ajouter la nouvelle cible aux informations d'état conservées. Le message NOTIFY inclut toutes les informations spécifiques de la cible.

- Si le flux a le niveau d'autorisation n° 2 (ok) : l'agent ST envoie un message CONNECT vers la cible avec une liste de cibles incluant la cible qui demandait à se joindre au flux. Cela aboutit finalement à ajouter la cible au flux. Lorsque l'agent ST reçoit le message ACCEPT qui indique que la nouvelle cible a été ajoutée, il ne propage pas le message ACCEPT vers l'arrière (paragraphe 4.5.6), ni ne le notifie à l'origine. Un message NOTIFY est généré avec le code de cause (TargetJoined) si les informations spécifiques de la cible ont besoin d'être propagées vers l'origine. Un exemple de telles informations est un changement de la MTU (voir au paragraphe 8.6).

4.6.3.1 Agent intermédiaire (routeur) comme origine

Lorsque un flux a le niveau d'autorisation d'adhésion n° 2 (voir au paragraphe 4.4.2) il est possible que l'origine du flux ne soit pas au courant de la participation de certaines cibles au flux. Dans ce cas, l'agent ST intermédiaire qui a le premier envoyé un message CONNECT à cette cible doit agir comme origine du flux pour cette cible. Cela inclut :

- o si tout le flux est supprimé, l'agent intermédiaire doit déconnecter la cible,
- o si la FlowSpec du flux est changée, l'agent intermédiaire doit changer la FlowSpec pour la cible comme approprié,
- o de traiter de façon appropriée les messages ACCEPT et REFUSE, sans propagation aux agents ST vers l'amont,
- o de générer des messages NOTIFY lorsque nécessaire (comme décrit plus haut).

L'agent intermédiaire se comporte normalement pour toutes les autres cibles ajoutées au flux en conséquence d'un message CONNECT produit par l'origine.

4.6.4 Cible se supprimant elle-même

L'application à la cible peut informer l'agent ST local qu'elle veut être retirée du flux. L'agent ST forme alors un message REFUSE avec la cible elle-même comme seule entrée dans la liste des cibles et avec le code de cause (ApplDisconnect). Le message REFUSE est renvoyé à l'origine via le bond précédent. Si un flux a plusieurs cibles et qu'une cible quitte le flux en utilisant ce mécanisme REFUSE, le flux sur les autres cibles n'est pas affecté ; le flux continue d'exister.

Un agent ST qui reçoit un message REFUSE en accuse réception en envoyant un ACK au prochain bond. La cible est supprimée et le LRM en est notifié de telle sorte qu'il ajuste les réservations comme approprié. Le message REFUSE est aussi propagé vers l'arrière à l'agent ST du bond précédent sauf dans le cas où l'agent agit comme origine. Dans ce cas, un message NOTIFY peut être propagé à la place (voir au paragraphe 4.6.3).

Lorsque le REFUSE atteint l'origine, l'origine envoie un ACK et notifie à l'application que la cible ne fait plus partie du flux.

4.6.5 Changement du FlowSpec d'un flux

L'application à l'origine peut souhaiter changer la FlowSpec d'un flux établi. Changer la FlowSpec est une opération critique et elle peut même conduire dans certains cas à la suppression des cibles affectées. Les problèmes possibles avec les changements de FlowSpec sont exposés au paragraphe 5.6.

Pour changer la FlowSpec du flux, l'application informe l'agent ST à l'origine de la nouvelle FlowSpec et de la liste des cibles qui se rapportent au changement. L'agent ST à l'origine produit alors un message CHANGE par prochain bond incluant la nouvelle FlowSpec et l'envoie aux agents ST de prochain bond pertinents. Si le champ G-bit du message CHANGE est établi (à 1), le changement affecte toutes les cibles dans le flux.

Le message CHANGE contient un bit appelé bit I (voir au paragraphe 10.4.3). Par défaut, le bit I est mis à zéro (0) pour indiquer qu'on s'attend à ce que le LRM essaye d'effectuer le changement de FlowSpec demandé sans risquer de supprimer le flux. Les applications qui désirent une plus forte probabilité de succès et qui acceptent de prendre le risque de casser le flux peuvent l'indiquer en mettant le bit I à un (1). Les applications qui exigent la modification demandée afin de continuer à fonctionner sont supposées établir ce bit *(à 1)*.

Un agent ST intermédiaire qui reçoit un message CHANGE envoie d'abord un ACK au bond précédent puis il fournit la FlowSpec au LRM. Si le LRM peut effectuer le changement, l'agent ST propage les messages CHANGE le long des chemins établis.

Si le processus entier réussit, les messages CHANGE vont finalement atteindre les cibles. Les cibles répondent par un message ACCEPT (ou REFUSE) qui est propagé en retour à l'origine. En traitant le message ACCEPT sur le chemin de retour vers l'origine, les ressources excédentaires peuvent être libérées par le LRM comme décrit au paragraphe 4.5.6. Le message REFUSE doit avoir le code de cause (ApplRefused).

SCMP fournit aussi un mécanisme d'arrosage pour changer les cibles qui se sont jointes au flux sans en notifier l'origine. Le cas particulier du changement de cible via l'arrosage est décrit au paragraphe 5.7.

4.7 Suppression de flux

Un flux est généralement terminé par l'origine lorsque elle n'a plus d'autres données à envoyer. Un flux est aussi supprimé si l'application doit se terminer de façon anormale ou si certaines défaillances du réseau sont rencontrées. Dans ce cas, le traitement est identique aux descriptions précédentes excepté que le code de cause (ApplAbort, NetworkFailure, etc.) est différent.

Lorsque toutes les cibles ont quitté un flux, l'origine notifie ce fait à l'application, et l'application est alors chargée de mettre un terme au flux. Noter cependant que l'application peut décider d'ajouter des cibles au flux au lieu d'y mettre un terme, ou de seulement laisser le flux ouvert sans cibles afin de permettre les adhésions au flux.

5. Cas exceptionnels

Les descriptions précédentes couvraient le cas simple où tout fonctionne. Nous allons maintenant exposer ce qui se passe lorsque les choses ne vont pas. Cela inclut les situations où les messages excèdent la MTU d'un réseau, sont perdus, où les ressources demandées sont indisponibles, où l'acheminement échoue ou est incohérent.

5.1 Messages ST trop longs

Il est possible qu'un agent ST, ou une application, ait besoin d'envoyer un message qui excède l'unité maximum de transmission (MTU) d'un réseau. Ce cas doit être traité mais pas via la fragmentation générique, car ST2 ne prend pas en charge la fragmentation générique des messages de données ni de contrôle.

5.1.1 Traitement des longs paquets de données

Les agents ST éliminent les paquets de données qui excèdent la MTU du réseau de prochain bond. Aucun message d'erreur n'est généré. Les applications devrait éviter d'envoyer des paquets de données plus longs que la MTU minimum supportée par un flux donné. L'application, aussi bien d'origine que de cibles, peut apprendre la MTU minimum du flux à travers le mécanisme de découverte de MTU décrit au paragraphe 8.6.

5.1.2 Traitement des longs paquets de contrôle

Chaque agent ST connaît la MTU des réseaux auxquels il est connecté, et ces MTU restreignent la taille des messages SCMP qu'il peut envoyer. Une taille de message SCMP peut dépasser la MTU d'un réseau donné pour un certain nombre de raisons :

- le paramètre Liste des cibles (paragraphe 10.3.6) peut être trop long ;
- le paramètre RecordRoute (paragraphe 10.3.5) peut être trop long ;
- le paramètre Données d'utilisateur (paragraphe 10.3.7) peut être trop long ;
- le champ PDUInError du message ERROR (paragraphe 10.4.6) peut être trop long.

Un agent ST qui reçoit ou génère un message SCMP trop long devrait :

- couper le message en plusieurs messages, portant chacun une partie de la liste des cibles. Tous les paramètres RecordRoute et Données d'utilisateur sont dupliqués dans chaque message pour livraison à toutes les cibles. Les applications qui acceptent un grand nombre de cibles peuvent éviter d'utiliser de longs paramètres Liste de cibles et ils sont supposés le faire, en exploitant les fonctions d'adhésion au flux (voir au paragraphe 4.6.3). Une exception existe à cette règle. Dans le cas d'un paramètre Liste de cibles long à inclure dans un message STATUS-RESPONSE, le paramètre Liste de cibles est simplement tronqué au point où la liste peut tenir dans un seul message (voir au paragraphe 8.4).

- pour les agents d'aval : si le paramètre Liste de cibles contient un seul élément de cible et si la taille du message est encore trop longue, l'agent ST devrait produire un message REFUSE avec le code de cause (RecordRouteSize) si la taille du paramètre RecordRoute est cause que la taille du message SCMP excède la MTU du réseau, ou avec le code de cause (UserDataSize) si la taille du paramètre Données d'utilisateur est cause que la taille du message SCMP excède la MTU du réseau. Si les paramètres RecordRoute et Données d'utilisateur sont tous deux présents, le code de cause (UserDataSize) devrait être envoyé. Pour les messages générés à la cible : l'agent ST de la cible doit vérifier les messages SCMP qui pourraient excéder la MTU sur le chemin complet de la cible à l'origine, et informer l'application que des messages SCMP trop longs ont été générés. Le format du rapport d'erreur est une question de mise en œuvre locale. Les codes d'erreur sont les mêmes qu'indiqué précédemment.

Les agents ST qui génèrent de trop longs messages ERROR tronquent simplement le champ PDUInError au point où le message est plus petit que la MTU du réseau.

5.2 Défaillance de temporisation

Comme décrit au paragraphe 4.3, la livraison de messages SCMP est rendu fiable par l'utilisation des accusés de réception, des temporisations et de la retransmission. Les messages ACCEPT, CHANGE, CONNECT, DISCONNECT, JOIN, JOIN-

REJECT, NOTIFY, et REFUSE doivent toujours recevoir un accusé de réception (voir au paragraphe 4.2). De plus, pour certains messages SCMP (CHANGE, CONNECT, JOIN) l'agent ST qui envoie s'attend aussi à une réponse en retour (ACCEPT/REFUSE, CONNECT/JOIN-REJECT) après réception d'un ACK. Le message STATUS doit aussi recevoir un message STATUS-RESPONSE en retour.

Les paragraphes qui suivent décrivent le traitement de chacun des cas de défaillance possibles dus aux situations de fin de temporisation lors de l'attente d'un accusé de réception ou d'une réponse. Les variables qui se rapportent à la fin de temporisation, et leurs noms, utilisés dans les paragraphes suivant ne servent que de référence. Ce sont des éléments spécifiques de la mise en œuvre. Des mises en œuvre différentes ne sont pas obligées d'avoir les même noms de variables, ou les mêmes mécanismes par lesquels le comportement de fin de temporisation ou de retransmission sont mis en œuvre.

5.2.1 Défaillance due à une fin de temporisation d'accusé de réception de ACCEPT

Un agent ST qui envoie un message ACCEPT vers l'amont s'attend à un ACK provenant de l'agent ST du bond précédent . Si aucun ACK n'est reçu avant l'arrivée à expiration du temporisateur ToAccept, l'agent ST devrait réessayer et envoyer le message ACCEPT à nouveau. Après NAccept réessais sans succès, l'agent ST envoie un message REFUSE vers l'origine, et un message DISCONNECT vers les cibles. REFUSE et DISCONNECT doivent tous deux identifier les cibles affectées et spécifier le code de cause (RetransTimeout).

5.2.2 Défaillance due à une fin de temporisation d'accusé de réception de CHANGE

Un agent ST qui envoie un message CHANGE vers l'aval attend un ACK de la part de l'agent ST du prochain bond. Si aucun ACK n'est reçu avant l'arrivée à expiration du temporisateur ToChange, l'agent ST devrait réessayer et envoyer le message CHANGE à nouveau. Après NChange essais sans succès, l'agent ST abandonne la tentative de modification en envoyant un message REFUSE vers l'origine, et un message DISCONNECT vers les cibles. REFUSE et DISCONNECT doivent tous deux identifier les cibles affectées et spécifier le code de cause (RetransTimeout).

5.2.3 Défaillance due à une fin de temporisation de réponse à CHANGE

Seul l'agent ST de l'origine met en œuvre ce temporisateur. Après avoir correctement reçu le ACK à un message CHANGE, un agent ST s'attend à recevoir un message ACCEPT ou REFUSE en réponse. Si un de ces messages n'est pas reçu avant l'arrivée à expiration du temporisateur ToChangeResp, l'agent ST de l'origine abandonne la tentative de modification, et se comporte comme si un message REFUSE avec le bit E mis et un code de cause (ResponseTimeout) était reçu.

5.2.4 Défaillance due à une fin de temporisation d'accusé de réception de CONNECT

Un agent ST qui envoie un message CONNECT vers l'aval s'attend à un ACK de la part de l'agent ST de prochain bond. Si aucun ACK n'est reçu avant l'arrivée à expiration du temporisateur ToConnect, l'agent ST devrait réessayer et envoyer le message CONNECT à nouveau. Après NConnect essais sans succès, l'agent ST envoie un message REFUSE vers l'origine, et un message DISCONNECT vers les cibles. REFUSE et DISCONNECT doivent identifier les cibles affectées et spécifier le code de cause (RetransTimeout).

5.2.5 Défaillance due à une fin de temporisation de réponse à CONNECT

Seul l'agent ST de l'origine met en œuvre ce temporisateur. Après avoir correctement reçu le ACK à un message CONNECT, un agent ST s'attend à recevoir un message ACCEPT ou REFUSE en réponse. Si un de ces messages n'est pas reçu avant l'arrivée à expiration du temporisateur ToConnectResp, l'agent ST de l'origine abandonne la tentative de modification, agit comme si un message REFUSE avait été reçu, et envoie un message DISCONNECT vers les cibles. REFUSE et DISCONNECT doivent identifier les cibles affectées et spécifier le code de cause (ResponseTimeout).

5.2.6 Défaillance due à une fin de temporisation d'accusé de réception de DISCONNECT

Un agent ST qui envoie un message DISCONNECT vers l'aval s'attend à un ACK de la part de l'agent ST du prochain bond. Si aucun ACK n'est reçu avant l'arrivée à expiration du temporisateur ToDisconnect, l'agent ST devrait réessayer et envoyer le message DISCONNECT à nouveau. Après NDisconnect essais sans succès, l'agent ST abandonne simplement et il suppose que l'agent ST du prochain bond ne fait plus partie du flux.

5.2.7 Défaillance due à une fin de temporisation d'accusé de réception de JOIN

Un agent ST qui envoie un message JOIN vers l'origine attend un ACK de la part de l'agent ST voisin. Si aucun ACK n'est

reçu avant l'arrivée à expiration du temporisateur ToJoin, l'agent ST devrait réessayer et envoyer le message JOIN à nouveau. Après NJoin essais sans succès, l'agent ST renvoie un message JOIN-REJECT dans la direction de la cible avec le code de cause (RetransTimeout).

5.2.8 Défaillance due à une fin de temporisation de réponse à JOIN

Seul l'agent de la cible met en œuvre ce temporisateur. Après avoir correctement reçu le ACK à un message JOIN, l'agent ST de la cible s'attend à recevoir un message CONNECT ou un message JOIN-REJECT en réponse. Si un de ces messages n'est pas reçu avant l'arrivée à expiration du temporisateur ToJoinResp, l'agent ST abandonne la tentative de se joindre au flux et retourne une erreur correspondante avec le code de cause (RetransTimeout) à l'application.

Noter que, après avoir correctement reçu le ACK à un message JOIN, les agents ST intermédiaires ne conservent aucun état sur la tentative d'adhésion au flux. Par conséquent, ils ne lancent pas le temporisateur ToJoinResp et n'attendent pas de message CONNECT ou JOIN-REJECT. Ceci est décrit au paragraphe 4.6.3.

5.2.9 Défaillance due à une fin de temporisation d'accusé de réception de JOIN-REJECT

Un agent ST qui envoie un message JOIN-REJECT vers la cible attend un ACK de la part de l'agent ST voisin. Si aucun ACK n'est reçu avant l'arrivée à expiration du temporisateur ToJoinReject, l'agent ST devrait réessayer et envoyer le message JOIN-REJECT à nouveau. Après NJoinReject essais sans succès, l'agent ST abandonne simplement.

5.2.10 Défaillance due à une fin de temporisation d'accusé de réception de NOTIFY

Un agent ST qui envoie un message NOTIFY à un agent ST voisin s'attend à un ACK de la part de cet agent ST. Si aucun ACK n'est reçu avant l'arrivée à expiration du temporisateur ToNotify, l'agent ST devrait réessayer et envoyer le message NOTIFY à nouveau. Après NNotify essais sans succès, l'agent ST abandonne simplement et se comporte comme si le message ACK avait été reçu.

5.2.11 Défaillance due à une fin de temporisation d'accusé de réception de REFUSE

Un agent ST qui envoie un message REFUSE vers l'amont s'attend à recevoir un ACK de la part de l'agent ST du bond précédent. Si aucun ACK n'est reçu avant l'arrivée à expiration du temporisateur ToRefuse, l'agent ST devrait réessayer et envoyer le message REFUSE à nouveau. Après NRefuse essais sans succès, l'agent ST abandonne et suppose qu'il ne fait plus partie du flux.

5.2.12 Défaillance due à une fin de temporisation de réponse à STATUS

Après l'envoi d'un message STATUS à un agent ST voisin, un agent ST s'attend à recevoir un message STATUS-RESPONSE en réponse. Si ce message n'est pas reçu avant l'arrivée à expiration du temporisateur ToStatusResp, l'agent ST envoie à nouveau le message STATUS. Après NStatus essais sans succès, l'agent ST abandonne et suppose que l'agent ST voisin n'est plus actif.

5.3 Échec d'établissement dû à une défaillance d'acheminement

Il est possible qu'un agent ST reçoive un message CONNECT qui contient un SID connu, mais d'un agent ST autre que l'agent ST du bond précédent du flux avec ce SID. Cela peut être :

1. que deux branches de l'arborescence formant le flux se sont jointes ensemble,
2. le résultat d'une tentative de récupération d'un flux partiellement échoué, ou
3. une boucle d'acheminement.

La liste des cibles contenue dans le message CONNECT est utilisée pour distinguer les différents cas en comparant chaque cible nouvelle reçue avec celles du flux existant précédemment :

- si l'adresse IP de la ou des cibles diffère, c'est le cas n° 1 ;
- si la cible correspond à une cible dans le flux existant, cela peut être le cas n° 2 ou n° 3.

Le cas n° 1 est traité au paragraphe 5.3.1, alors que les autres cas sont traités au paragraphe 5.3.2.

5.3.1 Convergence de chemin

Il est possible qu'un agent ST reçoive un message CONNECT qui contient un SID connu, mais d'un agent ST autre que l'agent ST du bond précédent du flux avec ce SID. Cela peut résulter de ce que deux branches de l'arborescence qui forme le flux se sont jointes. La détection de ce cas et les autres sources possibles sont exposées au paragraphe 5.2.

SCMP ne permet pas que des flux aient des chemins convergents, c'est-à-dire que les flux sont toujours en forme d'arborescence et jamais en forme de graphe. Au point de convergence, l'agent ST qui détecte cette condition génère un message REFUSE avec le code de cause (PathConvergence). Aussi, pour aider l'agent ST amont, l'agent qui la détecte place l'adresse IP de l'une des cibles connectées au flux dans le champ ValidTargetIPAddress du message REFUSE. Cette adresse IP sera utilisée par les agents ST amont pour éviter de partager le flux.

Un agent ST amont qui reçoit le message REFUSE avec le code de cause (PathConvergence) va vérifier pour voir si l'adresse IP de la liste est une des cibles connues du flux. Si non, le message REFUSE est propagé à l'agent du bond précédent. Si l'adresse IP de la liste est connue par l'agent ST amont, cet agent ST est l'agent ST qui a causé le partage dans le flux. (Cet agent peut même être l'origine.) Cet agent évite alors de partager le flux en utilisant le prochain bond de cette cible connue comme prochain bond pour les cibles refusées. Il envoie un message CONNECT avec les cibles affectées au prochain bond valide existant.

Le processus ci-dessus va se poursuivre, bond par bond, jusqu'à ce que la ValidTargetIPAddress corresponde à l'adresse IP d'une cible connue. Le seul cas où ce processus échouera est quand la cible connue est supprimée avant que le REFUSE se propage jusqu'à l'origine. Dans ce cas, l'origine peut juste réémettre le CONNECT et recommencer tout le processus.

5.3.2 Autres cas

Il n'est pas facile de distinguer les cas restants, y compris ceux de l'échec partiel d'un flux et d'une boucle d'acheminement. Pour tenter la récupération d'un flux défaillant, un agent ST peut produire de nouveaux messages CONNECT auprès des cibles affectées. Un tel CONNECT peut atteindre un agent ST à l'aval de la défaillance avant que cet agent ST ait reçu un DISCONNECT du voisinage de la défaillance. Tant que cet agent ST n'a pas reçu le DISCONNECT, il ne peut pas distinguer entre une récupération de défaillance et une boucle d'acheminement erronée. Cet agent ST doit donc répondre au message CONNECT par un message REFUSE avec les cibles affectées spécifiées dans la liste des cibles et un code de cause approprié (StreamExists).

L'agent ST qui précède immédiatement ce point, c'est-à-dire, le dernier agent ST à avoir envoyé le message CONNECT, va recevoir le message REFUSE. Il doit libérer toutes les ressources réservées exclusivement pour le trafic des cibles de la liste. Si cet agent ST n'était pas celui qui a tenté la récupération du flux, il ne peut alors pas distinguer entre une récupération sur défaillance et une boucle d'acheminement erronée. Il devrait répéter le CONNECT après une expiration du temporisateur ToConnect (voir au paragraphe 5.2.4). Si après NConnect retransmissions il continue de recevoir des messages REFUSE, il devrait propager le message REFUSE vers l'origine, avec la liste des cibles qui spécifie les cibles affectées, mais avec un code de cause différent (RouteLoop).

Le message REFUSE avec ce code de cause (RouteLoop) est propagé par chaque agent ST sans retransmettre de message CONNECT. À chaque agent ST, il cause la libération de toutes les ressources réservées exclusivement aux cibles de la liste. Le REFUSE sera propagé à l'origine dans le cas d'une boucle d'acheminement erronée. Dans le cas de récupération de flux, il sera propagé à l'agent ST qui tente la récupération, qui peut être un agent ST intermédiaire ou l'origine elle-même. Dans le cas d'une récupération de flux, l'agent ST qui tente la récupération peut produire un nouveau message CONNECT aux mêmes prochains bonds ou à des prochains bonds différents.

Si un agent ST reçoit à la fois un message REFUSE et un message DISCONNECT avec une cible en commun, il peut alors, pour chaque cible en commun, libérer les ressources pertinentes et ne propager ni le REFUSE ni le DISCONNECT.

Si l'origine reçoit un tel message REFUSE, elle devrait essayer d'envoyer un nouveau CONNECT à toutes les cibles affectées. Comme les erreurs d'acheminement dans l'internet sont supposées être temporaires, les nouveaux CONNECT finiront par trouver des chemins acceptables pour les cibles, s'il en existe. Si aucun autre chemin n'existe après NRetryRoute essais, l'application devrait en être informée de sorte qu'elle puisse prendre toute mesure qui semblerait nécessaire.

5.4 Problèmes dus à des incohérences d'acheminement

Lorsque un agent ST intermédiaire reçoit un message CONNECT, il invoque l'algorithme d'acheminement pour choisir les agents ST du prochain bond sur la base de la liste des cibles et des réseaux auxquels il souhaite être connecté. Si le prochain bond qui en résulte vers une des cibles est à travers le même réseau que celui dont il a reçu le CONNECT (mais pas le bond précédent lui-même), il peut y avoir un problème d'acheminement. Cependant, l'algorithme d'acheminement au bond précédent

peut optimiser différemment de ce que l'algorithme local ferait dans la même situation. Comme l'agent ST local ne peut pas distinguer les deux cas, il devrait permettre l'établissement mais renvoyer à l'agent ST du bond précédent un message NOTIFY informatif avec le code de cause approprié (RouteBack), la liste des cibles pertinente, et dans l'élément NextHopIPAddress l'adresse de l'agent ST du prochain bond retournée par son algorithme d'acheminement.

L'agent ST qui reçoit un tel NOTIFY devrait en accuser réception par un ACK. Si l'agent ST utilise un algorithme qui va produire un tel comportement, aucune autre action n'est entreprise ; sinon, l'agent ST devrait envoyer un DISCONNECT à l'agent ST du prochain bond pour corriger le problème.

Autrement, si le prochain bond retourné par la fonction d'acheminement est en fait le bond précédent, une incohérence d'acheminement a été détectée. Dans ce cas, un REFUSE est renvoyé à l'agent ST du bond précédent contenant un code de cause approprié (RouteInconsist), la liste des cibles pertinente, et dans l'élément NextHopIPAddress l'adresse du bond précédent. Lorsque le bond précédent reçoit le REFUSE, il va recalculer le prochain bond pour les cibles affectées. Si il y a une différence dans les bases de données d'acheminement chez les deux agents ST, ils peuvent échanger à nouveau des messages CONNECT et REFUSE. Comme de telles erreurs d'acheminement sont supposées être temporaires dans l'Internet, la situation devrait finalement se stabiliser.

5.5 Problèmes de réservation de ressources

Comme mentionné au paragraphe 1.4.5, la réservation de ressources est traitée par le LRM. Le LRM peut n'être pas capable de satisfaire une demande particulière durant l'établissement ou la modification d'un flux pour un certain nombre de raisons, y compris une discordance de FlowSpec, un numéro de version de FlowSpec inconnu, une erreur de traitement de FlowSpec, et une incapacité à allouer les ressources demandées. La présente section discute de ces cas et spécifie les codes de cause qui devraient être utilisés lorsque ces cas d'erreur se rencontrent.

5.5.1 FlowSpecs discordants

Dans certains cas, le LRM peut exiger qu'une FlowSpec demandée corresponde à une FlowSpec existante, par exemple, lors de l'ajout de nouvelles cibles à un flux existant (voir au paragraphe 4.6.1). Dans le cas d'une discordance de FlowSpec, le LRM la notifie à l'agent ST de traitement qui devrait répondre avec le code de cause (FlowSpecMismatch).

5.5.2 Version de FlowSpec inconnue

Lorsque le LRM est invoqué, les informations qui lui sont passées comportent la version de la FlowSpec (voir au paragraphe 4.5.2.2). Si cette version n'est pas connue du LRM, celui-ci le notifie à l'agent ST. L'agent ST devrait répondre par un message REFUSE avec le code de cause (FlowVerUnknown).

5.5.3 LRM incapable de traiter le FlowSpec

Le LRM peut rencontrer une erreur spécifique du LRM ou de la FlowSpec lorsqu'il tente de satisfaire une demande. Un exemple d'une telle erreur est donné au paragraphe 9.2.1. Ces erreurs sont spécifiques de la mise en œuvre et ne seront pas énumérées avec les codes de cause ST. Elles sont couvertes par un seul code de cause générique. Lorsque un LRM rencontre une telle erreur, il devrait en notifier l'agent ST qui devrait répondre par le code de cause générique (FlowSpecError).

5.5.4 Ressources insuffisantes

Si le LRM ne peut pas faire les réservations nécessaires à cause de l'insuffisance des ressources disponibles, un agent ST peut :

- essayer des chemins de remplacement vers les cibles : l'agent ST invoque la fonction d'acheminement pour trouver un chemin différent vers les cibles. Si un chemin de remplacement est trouvé, l'établissement de la connexion du flux continue de la façon normale, comme décrit au paragraphe 4.5.

- refuser d'établir le flux le long de ce chemin : l'agent ST d'origine informe l'application de l'échec de l'établissement du flux ; les agents ST intermédiaire et de cible produisent un message REFUSE (comme décrit au paragraphe 4.5.8) avec le code de cause (CantGetResrc).

Il dépend des mises en œuvre locales de savoir si un agent ST essaye des chemins de remplacement ou refuse d'établir le flux. Dans tous les cas, si des ressources suffisantes ne peuvent être trouvées sur des chemins différents, l'agent ST doit explicitement refuser d'établir le flux.

5.6 Problèmes causés par des messages CHANGE

Un message CHANGE peut échouer pour diverses raisons, parmi lesquelles :

- o des ressources insuffisantes : la demande peut concerner une quantité de ressources supérieure à celles qui sont disponibles ; le code de cause sera (CantGetResrc) ;

- o une application cible n'est pas d'accord avec le changement ; le code de cause sera (ApplRefused);

Le flux affecté peut être laissé dans un des deux états par suite d'un échec de modification :
a) le flux peut revenir à l'état où il était avant le traitement du message CHANGE, ou
b) le flux peut être supprimé.

Le cas de défaillance courant auquel on s'attend va être celui où le changement demandé ne peut être satisfait, mais les ressources allouées avant le changement restent allouées et disponibles pour l'usage du flux. Dans ce cas, l'agent ST du point où s'est produit la défaillance doit informer les agents ST en amont de la défaillance. (Dans le cas où cet agent ST est la cible, il peut n'y avoir en réalité pas de défaillance, et l'application peut tout simplement n'être pas d'accord avec le changement). L'agent ST informe les agents ST de l'amont en envoyant un message REFUSE avec le code de cause (CantGetResrc ou ApplRefused). Pour indiquer que la FlowSpec d'avant le changement est toujours disponible et que le flux existe toujours, l'agent ST règle le bit E du message REFUSE à un (1) (voir au paragraphe 10.4.11). Les agents ST de l'amont qui reçoivent le message REFUSE informent le LRM de sorte qu'il puisse essayer de revenir à la FlowSpec d'avant le changement. Il est permis, mais pas souhaitable, que les ressources en excès restent allouées.

Pour le cas où la tentative de changement du flux résulte en la perte des ressources réservées précédemment, le flux est supprimé. Cela peut arriver, par exemple, lorsque le bit I est établi (paragraphe 4.6.5) et que le LRM libère les ressources du flux d'avant le changement avant que les nouvelles ne soit réservées, et que ni les nouvelles ni les anciennes ressources ne sont disponibles. Dans ce cas, l'agent ST de l'endroit où la défaillance s'est produite doit informer les autres agents ST de la rupture de la portion du flux affectée. Ceci est fait par l'agent ST en envoyant un message REFUSE vers l'amont et un message DISCONNECT vers l'aval, tous deux avec le code de cause (CantGetResrc). Pour indiquer que les ressources du flux antérieures au changement ont été perdues, le bit E du message REFUSE est réglé à zéro (0).

Noter qu'un échec de changement des ressources demandées pour des cibles spécifiques ne devrait pas être cause que d'autres cibles dans le flux sont supprimées.

5.7 Cibles inconnues dans DISCONNECT et CHANGE

Le traitement des cibles inconnues dans la liste d'un message DISCONNECT ou CHANGE dépend du niveau d'autorisation d'adhésion du flux (voir au paragraphe 4.4.2). Pour les flux avec les niveaux d'autorisation d'adhésion n° 0 et n° 1 (voir au paragraphe 4.4.2) toutes les cibles doivent être connues. Dans ce cas, lors du traitement d'un message CHANGE, l'agent devrait générer un message REFUSE avec le code de cause (TargetUnknown). Lors du traitement d'un message DISCONNECT, il est possible que le DISCONNECT soit une duplication d'une ancienne demande, de sorte que l'agent devrait répondre comme si il avait réussi à déconnecter la cible. C'est-à-dire qu'il devrait répondre par un message ACK.

Pour les flux avec le niveau d'autorisation d'adhésion n° 2, il est possible que l'origine ne soit pas avertie de certaines cibles qui participent au flux. L'origine peut supprimer ou changer ces cibles via le mécanisme d'arrosage suivant.

Si aucun agent ST du prochain bond ne peut être associé à une cible, le message CHANGE/DISCONNECT qui comporte la cible est dupliqué à tous les agents ST de prochain bond. Ceci a pour effet de propager le message CHANGE/ DISCONNECT à tous les agents ST vers l'aval. Finalement, l'agent ST qui agit comme origine pour la cible (paragraphe 4.6.3.1) est atteint et la cible est supprimée.

La suppression/modification de cibles via l'arrosage n'est pas supposée être le cas normal. Elle est incluse pour présenter aux applications des capacités uniformes pour tous les types de flux. L'arrosage ne s'applique qu'aux flux qui ont le niveau d'autorisation d'adhésion n° 2.

6. Détection de défaillance et récupération

6.1 Détection de défaillance

Le mécanisme de détection de défaillance SCMP se fonde sur deux hypothèses :

1. Si un voisin d'un agent ST est actif, et a été actif sans interruption, et s'il n'a pas notifié à l'agent ST de problème avec les flux qui les traversent tous deux, l'agent ST peut alors supposer qu'il n'y a aucun problème avec ces flux.

2. Un réseau au travers duquel un agent ST a acheminé un flux va notifier à l'agent ST si un problème affecte les paquets de données du flux mais n'affecte pas les paquets de contrôle.

L'objet du protocole de robustesse défini ici est que les agents ST déterminent si le flux à travers un voisin a été coupé par la défaillance du voisin ou du réseau intervenant. Ce protocole devrait détecter l'immense majorité des défaillances qui peuvent survenir. Une fois qu'une défaillance est détectée, les procédures de récupération décrites au paragraphe 6.2 sont initiées par les agents ST.

6.1.1 Défaillances du réseau

Un agent ST peut détecter les défaillances du réseau par deux mécanismes :
- o le réseau peut rapporter une défaillance, ou
- o l'agent ST peut découvrir une défaillance par lui-même.

Elles diffèrent par la quantité d'informations qui sont disponibles à un agent ST afin de prendre une décision de récupération. Par exemple, un réseau peut être capable de rapporter que la bande passante réservée a été perdue et la cause de la perte, et il peut aussi rapporter que la connexité avec l'agent ST voisin reste intacte. D'un autre côté, un agent ST peut découvrir que la communication avec un agent ST voisin a cessé parce qu'il n'a reçu aucun trafic de la part de ce voisin depuis un certain temps. Si un agent ST détecte une défaillance, il peut n'être pas capable de déterminer si la défaillance était dans le réseau alors que le voisin reste disponible, ou si le voisin a eu une défaillance alors que le réseau reste intact.

6.1.2 Détection des défaillances des agents ST

Chaque agent ST envoie périodiquement à chaque voisin avec lequel il partage un ou plusieurs flux un message HELLO. Cet échange de messages est entre les agents ST, et non entre les entités qui représentent les flux ou les applications. C'est-à-dire qu'un agent ST a seulement besoin d'envoyer un seul message HELLO à un voisin, sans considération du nombre de flux qui s'écoulent entre eux. Tous les agents ST (hôtes aussi bien qu'intermédiaires) doivent participer à cet échange. Cependant, seuls les agents ST qui partagent des flux actifs peuvent participer à cet échange et c'est une erreur que d'envoyer un message HELLO à un agent ST voisin avec lequel il n'y a pas de flux en commun, par exemple, pour vérifier si il est actif. Les messages STATUS peuvent être utilisés pour interroger l'état des agents ST voisins (voir au paragraphe 8.4).

Pour les besoins de l'échange de messages HELLO, l'existence des flux est bordée par le traitement des messages ACCEPT et DISCONNECT/REFUSE et est définie aussi bien pour le cas vers l'amont que vers l'aval. Un flux vers un bond précédent est défini comme débutant une fois qu'un message ACCEPT a été transmis vers l'amont. Un flux pour un prochain bond est défini comme débutant une fois que le message ACCEPT reçu a eu son accusé de réception. Un flux est défini comme se terminant une fois qu'un accusé de réception a été envoyé pour un message DISCONNECT ou REFUSE reçu, et qu'un accusé de réception a été reçu pour un message DISCONNECT ou REFUSE envoyé.

Le message HELLO a deux champs :
- o un champ HelloTimer qui est en unités de millisecondes modulo le maximum pour la taille du champ, et
- o un bit Restarted qui spécifie si l'agent ST a été redémarré récemment.

Le HelloTimer doit apparaître comme étant incrémenté à chaque milliseconde qu'un message HELLO soit envoyé ou non. Le HelloTimer revient à zéro après avoir atteint la valeur maximum. Chaque fois qu'un agent ST subit un événement catastrophique qui peut avoir pour résultat la perte des informations d'état ST, il doit remettre son HelloTimer à zéro et il doit régler le bit Restarted dans tous les messages HELLO envoyés dans les HelloTimerHoldDown secondes suivantes.

Si un agent ST reçoit un message HELLO qui contient le bit Restarted établi, il doit supposer que l'agent ST qui l'envoie a perdu son état. Si il partage des flux avec ce voisin, il doit initier l'activité de récupération de flux (voir au paragraphe 6.2). Si il ne partage pas de flux avec ce voisin, il ne devrait pas tenter d'en créer un jusqu'à ce que ce bit ne soit plus établi. Si un agent ST reçoit un message CONNECT d'un voisin dont le bit Restarted est établi, l'agent doit répondre par un message ERROR avec le code de cause approprié (RestartRemote). Si un agent reçoit un message CONNECT alors que le propre bit Restarted

de l'agent est établi, l'agent doit répondre par un message ERROR avec le code de cause approprié (RestartLocal).

Chaque flux ST a une valeur de RecoveryTimeout associée. Cette valeur est allouée par l'origine et portée dans le message CONNECT (voir au paragraphe 4.5.10). Chaque agent vérifie pour voir si il peut prendre en charge la valeur demandée. Si il ne le peut pas, il met à jour la valeur au plus petit intervalle de temporisation qu'il peut accepter. Le RecoveryTimeout utilisé par un flux particulier est obtenu à partir du message ACCEPT (voir au paragraphe 4.5.10) et c'est la plus petite valeur vue à travers tous les messages ACCEPT provenant des cibles participantes.

Un agent ST doit envoyer des messages HELLO à ses voisins avec une période plus courte que la plus petite RecoveryTimeout de tous les flux actifs qui passent entre les deux agents ST, sans considération de la direction. Cette période doit être plus courte d'un facteur, appelé HelloLossFactor,qui est au moins aussi grand que le plus grand nombre de messages HELLO consécutifs dont on peut envisager de façon crédible qu'ils pourraient être perdus alors que la communication entre les deux agents ST est encore viable.

Un agent ST peut envoyer des messages HELLO simultanés à tous ses voisins au débit nécessaire pour prendre en charge le plus petit RecoveryTimeout de tout flux actif. Autrement, il peut envoyer des messages HELLO aux différents voisins indépendamment à des taux différents correspondant aux RecoveryTimeouts des flux individuels.

Un agent ST doit s'attendre à recevoir au moins un nouveau message HELLO de chaque voisin au moins aussi fréquemment que le plus petit RecoveryTimeout de tout flux actif en commun avec ce voisin. L'agent peut détecter les messages HELLO dupliqués ou retardés en comparant le champ HelloTimer du message HELLO valide le plus récent de ce voisin avec le champ HelloTimer d'un message HELLO entrant. Les messages HELLO entrant valides auront un champ HelloTimer supérieur au champ contenu dans le message HELLO valide reçu précédemment du temps écoulé depuis la réception du message précédent. L'évaluation réelle du temps écoulé devrait tenir compte de la variance du délai maximum vraisemblable depuis ce voisin.

Si l'agent ST ne reçoit pas un message HELLO valide dans l'intervalle RecoveryTimeout d'un flux, il doit supposer que l'agent ST voisin, ou la liaison de communication entre les deux, est défaillant et il doit initier l'activité de récupération de flux, comme décrit ci-dessous au paragraphe 6.2.

6.2 Récupération des défaillances

Si un agent ST intermédiaire ou un réseau, ou une partie d'un réseau a une défaillance, l'agent ST du bond précédant les divers agents ST du prochain bond va découvrir ce fait par le mécanisme de détection de défaillances décrit au paragraphe 6.1.

La récupération d'un flux ST est un effort relativement complexe et long parce qu'il est conçu d'une manière générale pour fonctionner à travers un grand nombre de réseaux avec des caractéristiques diverses. Donc, il peut exiger des informations largement réparties, et des temporisateurs relativement longs. D'un autre côté, comme un réseau est normalement un système homogène, la récupération des défaillances dans le réseau peut être une opération relativement plus rapide et plus simple. Donc un agent ST qui détecte une défaillance devrait tenter de réparer la défaillance du réseau avant de tenter la récupération du flux ST. Si le flux qui existait entre deux agents ST avant la défaillance ne peut pas être reconstruit par les seuls mécanismes de récupération du réseau, le mécanisme de récupération du flux ST doit être invoqué.

Si la récupération de flux est nécessaire, les différents agents ST devront effectuer différentes fonctions, selon leur relation avec la défaillance :

- Un agent ST qui est un prochain bond à partir d'une défaillance devrait d'abord vérifier qu'il y a bien une défaillance. Il peut le faire en utilisant les messages STATUS pour interroger ses voisins en amont. Si il ne peut pas communiquer avec ce voisin, pour chaque flux actif à partir de ce voisin, il devrait alors envoyer d'abord un message REFUSE vers l'amont avec le code de cause approprié (STAgentFailure). On fait cela sur le voisin pour accélérer la récupération de défaillance au cas où le bond serait unidirectionnel, c'est-à-dire que le voisin peut entendre l'agent ST mais l'agent ST ne peut pas entendre le voisin. L'agent ST qui détecte la défaillance doit alors, pour chaque flux actif à partir de ce voisin, envoyer un message DISCONNECT avec le même code de cause vers les cibles. Tous les agents ST vers l'aval traitent ce message DISCONNECT exactement comme le DISCONNECT qui supprime le flux. Si la récupération réussit, les cibles recevront de nouveaux messages CONNECT.

- Un agent ST qui est le bond précédent avant le composant défaillant vérifie d'abord qu'il y a une défaillance en interrogeant les voisins vers l'aval en utilisant des messages STATUS. Si le voisin a perdu son état mais est disponible, l'agent ST peut alors essayer de reconstruire (ce qui est expliqué ci-dessous) les flux affectés, pour ceux qui n'ont pas l'option NoRecovery marquée. Si il ne peut pas communiquer avec le prochain bond, l'agent ST qui a détecté la défaillance envoie alors un message DISCONNECT, pour chaque flux affecté, avec le code de cause approprié (STAgentFailure) vers les cibles affectées. Il fait cela pour accélérer la récupération de défaillance au cas où la communication serait unidirectionnelle et où

ce message pourrait être délivré avec succès.

Sur la base de l'option NoRecovery, l'agent ST qui était le bond précédent avant le composant défaillant prend les mesures suivantes :

- Si l'option NoRecovery est choisie, l'agent ST envoie alors, pour chaque flux affecté, un message REFUSE avec le code de cause approprié (STAgentFailure) au bond précédent. La liste des cibles dans ces messages contient toutes les cibles qui ont été atteintes à travers la branche brisée. Comme exposé au paragraphe 5.1.2, plusieurs messages REFUSE peuvent être nécessaires si la PDU est trop longue pour la MTU du réseau intervenant. Le message REFUSE est propagé sur tout le chemin jusqu'à l'origine. L'application de l'origine peut essayer la récupération du flux en envoyant un nouveau CONNECT aux cibles affectées. Pour établir les flux, le nouveau CONNECT sera traité par les agents ST intermédiaires comme un ajout de nouvelles cibles dans les flux établis.

- Si l'option NoRecovery n'est pas choisie, l'agent ST peut essayer la récupération des flux affectés. Il le fait flux par flux en produisant un nouveau message CONNECT auprès des cibles affectées. Si l'agent ST ne peut pas trouver de nouveaux chemins pour certaines cibles, ou si le seul chemin pour certaines cibles est à travers le bond précédent, il envoie alors un ou plusieurs messages REFUSE au bond précédent avec le code de cause approprié (CantRecover) spécifiant les cibles affectées dans la liste des cibles. Le bond précédent peut alors essayer la récupération du flux en envoyant un message CONNECT à ces cibles. Si il ne peut trouver un chemin approprié, il propagera le message REFUSE vers l'origine.

Quel que soit l'agent ST qui tente la récupération d'un flux endommagé, il produira un ou plusieurs messages CONNECT auprès des cibles affectées. Ces messages CONNECT sont traités par les agents ST intermédiaires comme des ajouts des nouvelles cibles dans les flux établi. Les FlowSpec des nouveaux messages CONNECT sont les mêmes que ceux contenus dans les messages CONNECT ou CHANGE les plus récents que l'agent ST avait envoyés aux cibles affectées lorsque le flux était opérationnel.

À réception d'un ACCEPT durant une récupération de flux, l'agent qui reconstruit le flux doit s'assurer que la FlowSpec et les autres attributs du flux (par exemple, MaxMsgSize et RecoveryTimeout) du flux rétabli sont égaux, ou moins restrictifs, que ceux du flux avant la défaillance. Si ils sont plus restrictifs, la tentative de récupération doit être abandonnée. Si ils sont égaux, ou sont moins restrictifs, la tentative de récupération est réussie. Lorsque la tentative est réussie, les ACCEPT relatifs à la récupération de la défaillance ne sont pas transmis vers l'amont par l'agent de la récupération.

Tout agent ST qui décide que suffisamment de tentatives de récupération ont été faites, ou que les tentatives de récupération n'ont aucune chance de réussite, peut indiquer qu'aucune autre tentative de récupération ne devrait être faite. Ceci se fait en établissant le bit N dans le message REFUSE (voir au paragraphe 10.4.11). Ce bit doit être établi par les agents, y compris la cible, qui savent qu'il n'y a aucune chance de réussir la récupération. Un agent ST qui reçoit un message REFUSE avec le bit N établi (à 1) ne tentera pas la récupération, sans considération de l'option NoRecovery, et il établira le bit N lorsqu'il propagera le message REFUSE vers l'amont.

6.2.1 Problèmes de la récupération de flux

La reconstruction d'un flux interrompu peut ne pas se passer en douceur. Comme il peut y avoir des délais pendant que les informations concernant la défaillance sont propagées tout autour d'un internet, des erreurs d'acheminement peuvent survenir pendant un certain temps après une défaillance. Il en résulte que l'agent ST qui tente la récupération peut recevoir des messages ERROR pour les nouveaux CONNECT qui sont causés par des erreurs d'acheminement de l'internet. L'agent ST qui tente la récupération devrait être prêt à renvoyer des CONNECT avant de réussir à reconstruire le flux. Si la défaillance provoque une partition de l'internet et qu'un nouvel ensemble de chemins ne peut être trouvé vers les cibles, les messages REFUSE seront finalement propagés à l'origine, qui peut alors informer l'application afin qu'elle décide si elle termine le flux ou si elle continue à tenter sa récupération.

Le nouveau CONNECT peut à un certain point atteindre un agent ST à l'aval de la défaillance avant que ne le fasse le DISCONNECT. Dans ce cas, l'agent ST qui reçoit le CONNECT n'est pas encore averti que le flux a subi une défaillance, et il va interpréter le nouveau CONNECT comme résultant d'une défaillance de l'acheminement. Il va répondre par un message ERROR avec le code de cause approprié (StreamExists). Comme la temporisation des agents ST qui précèdent immédiatement la défaillance et de ceux qui la suivent immédiatement est approximativement la même, il est très vraisemblable que les restes du flux endommagé seront bientôt éliminés par un message DISCONNECT. Donc, l'agent ST qui reçoit le message ERROR avec le code de cause (StreamExists) devrait retransmettre le message CONNECT après l'arrivée à expiration du temporisateur ToConnect. Si cela échoue à nouveau, la demande sera réessayée NConnect fois. C'est seulement si cela échoue encore que l'agent ST envoie un message REFUSE avec le code de cause approprié (RouteLoop) à son bond précédent. Ce message sera propagé en retour à l'agent ST qui tente la récupération du flux endommagé. Cet agent ST peut émettre un nouveau message CONNECT si tel est son choix. Le REFUSE est confronté à un message CONNECT créé par une opération de récupération à travers le champ LnkReference dans le message CONNECT.

Les agents ST qui ont propagé un message CONNECT et ont reçu un message REFUSE devraient conserver ces informations pendant un certain temps. Si un agent ST reçoit un second message CONNECT pour une cible qui a récemment résulté en un REFUSE, cet agent ST peut répondre immédiatement par un REFUSE plutôt que de tenter de propager le CONNECT. Ceci a pour effet d'élaguer l'arborescence qui est formée par la propagation des messages CONNECT sur une cible qui n'est pas joignable par les chemins qui sont sélectionnés en premier. L'arborescence va passer à travers un agent ST donné une seule fois, et la phase d'établissement du flux sera achevée plus vite.

Si un message CONNECT atteint une cible, celle-ci devrait aussi efficacement que possible utiliser l'état qu'elle a sauvegardé avant la défaillance du flux durant la récupération du flux. Elle va alors émettre un message ACCEPT vers l'origine. Le message ACCEPT sera intercepté par l'agent ST qui tente la récupération du flux endommagé, si ce n'est pas l'origine. Si la FlowSpec contenue dans ACCEPT spécifie la même sélection de paramètres que ceux qui étaient en effet avant la défaillance, l'agent ST qui tente la récupération ne propagera alors pas le ACCEPT. La comparaison des FlowSpec est faite par le LRM. Si la sélection des paramètres est différente, l'agent ST qui tente la récupération enverra alors à l'origine un message NOTIFY avec le code de cause approprié (FailureRecovery) qui contient une FlowSpec spécifiant les nouvelles valeurs des paramètres. L'origine peut alors devoir changer ses caractéristiques de génération de données et les paramètres du flux avec un message CHANGE pour utiliser le sous-arbre qui vient d'être récupéré.

6.3 Préemption de flux

Comme mentionné au paragraphe 1.4.5, il est possible que le LRM décide intentionnellement de casser un flux. C'est ce qu'on appelle la préemption de flux. Les flux sont supposés être préemptés afin de libérer des ressources pour un nouveau flux qui a une priorité supérieure.

Si le LRM décide qu'il est nécessaire de préempter un ou plusieurs des flux qui le traversent, il reste à décider quels flux doivent être préemptés. Une application a deux façon d'influencer une telle décision :

1. Sur la base des informations de la FlowSpec. Par exemple, avec la FlowSpec de ST2+, les flux peuvent être affectés d'une valeur de préséance de 0 (la moins importante) à 256 (la plus importante). Cette valeur est portée dans la FlowSpec lors de l'établissement du flux (voir au paragraphe 9.2) de telle sorte que le LRM en soit informé.

2. Avec le mécanisme de groupe. Une application peut spécifier qu'un ensemble de flux sont en relation les uns avec les autres et qu'ils sont tous candidats à la préemption si l'un d'eux est préempté. Cela peut être fait en utilisant la relation de partage de sort définie au paragraphe 7.1.2. Cela aide le LRM à faire un bon choix lorsque plus d'un flux doit être préempté, parce que cela conduit à casser une seule application plutôt que autant d'applications que le nombre de flux préemptés.

Si le LRM préempte un flux, il doit le notifier à l'agent ST local. Les actions suivantes sont effectuées par l'agent ST :

- L'agent ST chez l'hôte où le flux a été préempté envoie des messages DISCONNECT avec le code de cause approprié (StreamPreempted) vers les cibles affectées. Il envoie un message REFUSE avec le code de cause approprié (StreamPreempted) au bond précédent.

- Un agent ST du bond précédent le flux préempté agit comme dans le cas de récupération de défaillance (voir au paragraphe 6.2).

- Un agent ST du prochain bond du flux préempté agit comme dans le cas de récupération de défaillance (voir au paragraphe 6.2).

Noter que, à l'opposé de la récupération de défaillance, il n'est pas nécessaire de vérifier que la défaillance s'est réellement produite, parce que ceci est explicitement indiqué par le code de cause (StreamPreempted).

7. Groupe de flux

Il peut être nécessaire d'associer des flux en rapport. Le mécanisme de groupe est simplement une technique d'association qui permet aux agents ST d'identifier les différents flux qui doivent être associés.

Un groupe consiste en un ensemble de flux et d'une relation. L'ensemble de flux peut être vide. La relation s'applique à tous les membres du groupe. Chaque groupe est identifié par un nom de groupe. Le nom du groupe doit être unique au monde.

Les flux appartiennent au même groupe si ils ont le même nom de groupe dans le champ Nom de groupe du paramètre Groupe

(voir au paragraphe 10.3.2). La relation est définie par le champ Relation. L'appartenance au groupe doit être spécifiée à la création du flux et persister pour toute la durée de vie du flux. Un seul flux peut appartenir à plusieurs groupes.

L'agent ST qui crée un nouveau groupe est appelé l'initiateur du groupe. Tout agent ST peut être un initiateur de groupe. L'initiateur alloue le Nom de groupe et la Relation aux membres du groupe. L'initiateur peut ou non être l'origine d'un flux appartenant au groupe. La génération du Nom de groupe est décrite au paragraphe 8.2.

7.1 Relations de groupe de base

La présente version de ST définit quatre relations de groupe de base. Une mise en œuvre ST2+ doit prendre en charge les quatre relations de base. L'adhésion aux relations spécifiées est normalement au mieux. Les relations de base sont décrites en détail du paragraphe 7.1.1 au paragraphe 7.1.4 ci-dessous.

7.1.1 Partage de bande passante

Les flux associés au même groupe partagent la même bande passante du réseau. L'intention est de prendre en charge des applications telles que les audio conférences où, de tous les participants, seuls certains ont la parole à un moment donné. Dans un tel scénario, l'utilisation globale de la bande passante peut être diminuée en allouant seulement les ressources qui peuvent être utilisées immédiatement, par exemple, il est suffisant de réserver la bande passante pour de petits ensembles de flux audio.

Le concept de base du groupe de partage de la bande passante est que le LRM va allouer jusqu'à un multiple spécifié du flux le plus exigeant connu dans le groupe. Le LRM va allouer les ressources de façon incrémentaire, au fur et à mesure de la réception des demandes d'établissement de flux, jusqu'à ce que le total des exigences du groupe soient satisfaites. Les demandes d'établissement suivantes vont partager les ressources du groupe et n'auront pas besoin d'une allocation de ressources supplémentaires. La procédure va résulter en une allocation standard où un seul flux d'un groupe traverse un agent, et en des allocations partagées où plusieurs flux traversent un agent.

Pour illustrer cela, appelons le multiple mentionné ci-dessus "N", et le flux le plus exigeant que connaisse un agent dans un groupe Bmax. Pour une application qui entend permettre à trois participants de parler en même temps, N a une valeur de trois et chaque LRM va allouer pour le groupe une quantité de bande passante jusqu'à 3*Bmax même lorsqu'il y a beaucoup plus de flux dans le groupe. Le LRM va réserver les ressources par incrément, selon les demandes des flux, jusqu'à ce que N*Bmax ressources soient allouées. Chaque agent peut être traversé par un ensemble et nombre de flux différent appartenant tous au même groupe.

Un agent ST qui reçoit une demande de flux présente au LRM toutes les informations de groupe nécessaires (voir au paragraphe 4.5.2.2). Si la bande passante maximum pour le groupe, N*Bmax, a déjà été allouée et qu'un nouveau flux avec une demande de bande passante inférieure à Bmax est à établir, le LRM n'allouera aucune bande passante supplémentaire.

Si il y a moins de N*Bmax ressources allouées, le LRM va étendre les ressources allouées au groupe de la quantité demandée dans la nouvelle FlowSpec, jusqu'à N*Bmax ressources. Le LRM va mettre à jour la FlowSpec sur la base des ressources disponibles pour le flux, mais pas sur les ressources totales allouées au groupe.

On devrait noter que les agents ST et les LRM n'apprennent les exigences d'un groupe que lorsque les flux qui appartiennent au groupe sont créés. Dans le cas de la relation de partage de la bande passante, une application devrait tenter d'établir d'abord les flux les plus exigeants pour minimiser les efforts d'établissement de flux. Si au contraire les flux les moins exigeants sont construits les premiers, il sera toujours nécessaire d'allouer de la bande passante additionnelle dans les étapes consécutives lorsque les flux les plus exigeants seront construits. Il appartient aussi aux applications de coordonner leurs différentes FlowSpec et de décider de la valeur appropriée pour N.

7.1.2 Partage de sort

Les flux qui appartiennent à ce groupe partagent le même sort. Si un flux est supprimé, les autres membres du groupe sont aussi supprimés. Cela est destiné à prendre en charge la préemption de flux par l'indication des flux qui sont en relation mutuelle. Si la préemption de plusieurs flux est nécessaire, ces informations peuvent être utilisées par le LRM pour supprimer un ensemble de flux en relation, par exemple, avec un impact sur une seule application, au lieu de faire un choix aléatoire avec pour effet possible d'interrompre plusieurs applications différentes. Cet attribut ne s'applique pas à la fermeture normale de flux, c'est-à-dire, avec le code de cause (ApplDisconnect). Sur une déconnexion normale, les autres flux appartenant à un tel groupe restent actifs.

Cette relation donne une indication des flux qui devraient être préemptés. Cependant, le LRM responsable de la préemption n'est pas forcé de se comporter de cette façon , et d'autres flux pourraient être préemptés d'abord, sur la base de critères différents.

7.1.3 Partage de chemin

Les flux qui appartiennent à ce groupe partagent autant que possible les mêmes chemins. Cela peut être souhaitable pour plusieurs raisons, par exemple, pour exploiter les mêmes ressources allouées ou pour tenter de conserver l'ordre de transmission. Un agent ST tente de choisir le même chemin bien que la façon dont ceci est mis en œuvre dépende fortement de l'algorithme d'acheminement utilisé.

Si l'algorithme d'acheminement est assez sophistiqué, un agent ST peut suggérer qu'un flux soit acheminé sur un chemin déjà établi. Autrement, il peut demander à l'algorithme d'acheminement un ensemble de chemins légaux pour la destination et vérifier si le chemin désiré est inclus dans ceux qui sont praticables.

Le partage de chemin est un conseil de l'algorithme d'acheminement utilisé par ST. Ne pas acheminer un flux à travers un chemin partagé ne pourrait pas empêcher la création d'un nouveau flux ou résulter en la suppression d'un flux existant.

7.1.4 Partage de ressources de sous-réseau

Cette relation donne une indication aux fonctions de couche de liaison des données. Les flux qui appartiennent à ce groupe peuvent partager les mêmes ressources de couche MAC. Par exemple, la même adresse de diffusion groupée de couche MAC peut être utilisée pour tous les flux dans un groupe donné. Ce mécanisme permet une meilleure utilisation des adresses de diffusion groupée de couche MAC et elle est particulièrement utile lorsque elle est utilisée avec des adaptateurs de réseau qui offrent un très petit nombre d'adresses de diffusion groupée de couche MAC.

7.2 Orthogonalité des relations

Les quatre relations de base, comme elles ont été définies, sont orthogonales. Cela signifie que toutes les combinaisons des relations de base sont permises. Par exemple, considérons une application qui exige un service bidirectionnel pour un flux avec plusieurs cibles. Supposons aussi que seules N cibles aient la permission de renvoyer en même temps des données à l'origine. Dans ce scénario, tous les flux inverses pourraient appartenir au même groupe. Ils pourraient partager à la fois les chemins et les attributs de bande passante. La relation de partage de chemin et de bande passante est obtenue de l'ensemble de relations de base. Cet exemple est important parce qu'il montre comment un service bidirectionnel peut être efficacement obtenu dans ST.

8. Fonctions auxiliaires

Certaines fonctions sont exigées des mises en œuvre d'hôte et agent intermédiaire ST. De telles fonctions sont décrites dans cette section.

8.1 Génération de l'identifiant de flux

L'identifiant de flux, ou SID, se compose d'un identifiant univoque de 16 bits et de l'adresse IP à 32 bits de l'adresse IP du flux. Les identifiants de flux doivent être uniques au monde. La définition et le format spécifique du champ de 16 bits sont laissés au gré de la mise en œuvre. Le champ est supposé n'avoir qu'une signification locale.

Une mise en œuvre de ST doit fournir une facilité de générateur d'identifiant de flux, de telle sorte qu'un protocole de couche application ou supérieur puisse obtenir des identifiants univoques à partir de la couche ST. C'est un mécanisme pour que l'application demande l'allocation d'un identifiant de flux qui soit indépendant de la demande de création d'un flux. L'identifiant de flux est utilisé par le protocole d'application ou de couche supérieure lors de la création des flux.

Par exemple, les deux fonctions suivantes pourraient être disponibles :
- AllocateStreamID() -> résultat, StreamID
- ReleaseStreamID(StreamID) -> résultat

Une mise en œuvre peut aussi fournir une fonction de suppression d'identifiant de flux.

8.2 Générateur de nom de groupe

La génération du nom de groupe est similaire à celle de l'identifiant de flux. Le nom de groupe comporte un identifiant univoque de 16 bits, un horodatage de création de 32 bits, et une adresse IP de 32 bits. Les noms de groupe sont uniques au

monde. Un nom de groupe comporte l'adresse IP du créateur, de sorte que cela réduit le problème de l'unicité mondiale à un simple problème local. Les définitions et formats spécifiques du champ de 16 bits et de l'horodatage de création de 32 bits sont laissés à l'initiative de la mise en œuvre. Ces champs doivent être localement uniques, et n'ont de signification que localement.

Une mise en œuvre de ST doit fournir une facilité de générateur de nom de groupe, afin qu'un protocole d'application ou de couche supérieure puisse obtenir un nom de groupe unique de la couche ST. C'est un mécanisme pour que l'application demande l'allocation d'un nom de groupe qui soit indépendant de la demande de création d'un flux. Le nom de groupe est utilisé par le protocole d'application ou de couche supérieure lors de la création des flux qui doivent faire partie du groupe.

Par exemple, les deux fonctions suivantes pourraient être disponibles :
- AllocateGroupName() -> résultat, Nom de groupe
- ReleaseGroupName(GroupName) -> résultat

Une mise en œuvre peut aussi fournir une fonction de suppression de nom de groupe.

8.3 Calcul de somme de contrôle

L'algorithme standard de somme de contrôle Internet est utilisé pour ST : "Le champ Somme de contrôle est le complément à un de 16 bits de la somme des compléments à un de tous les mots de 16 bits de l'en-tête. Pour les besoins du calcul de la somme de contrôle, la valeur du champ Somme de contrôle est zéro (0)." Voir les [RFC1071], [RFC1141] et [RFC791] pour trouver des suggestions d'algorithmes efficaces de somme de contrôle.

8.4 Identification d'agent ST voisin et collecte d'informations

Le message STATUS peut être utilisé pour collecter des informations sur les agents ST voisins, les flux que le voisin prend en charge, et les cibles spécifiques des flux que le voisin prend en charge. Un agent qui reçoit un message STATUS fournit les informations demandées via un message STATUS-RESPONSE.

Le message STATUS peut être utilisé pour collecter différentes informations d'un voisin. Il peut être utilisé pour :

- identifier les voisins à capacité ST. Si un agent ST souhaite vérifier si un voisin à la capacité ST, il devrait générer un message STATUS avec un SID qui a tous ses champs mis à zéro. Un agent qui reçoit un message STATUS avec un tel SID devrait répondre par un STATUS-RESPONSE contenant le même SID, et aucune autre information de flux. L'agent ST qui reçoit doit répondre aussitôt que possible pour aider à l'estimation du temps d'aller-retour (voir au paragraphe 8.5) ;

- obtenir des informations sur un flux particulier. Si un agent ST souhaite vérifier les informations générales du voisin relatives à un flux spécifique, il devrait générer un message STATUS contenant le SID du flux. Un agent ST qui reçoit un tel message, va d'abord vérifier si le flux est connu. Si il n'est pas connu, l'agent ST qui reçoit envoie un STATUS-RESPONSE contenant le même SID, et aucune autre information de flux. Si le flux est connu, l'agent ST qui reçoit envoie un STATUS-RESPONSE contenant le SID du flux, les bonds IP, la FlowSpec, les membres du groupe (s'il en est), et autant de cibles qu'il peut en être inclus dans un seul message avec sa limite de MTU (voir au paragraphe 5.1.2). Noter que toutes les cibles peuvent n'être pas incluses dans une réponse à une demande d'informations générales de flux. Si des informations sur une cible spécifique d'un flux sont désirées, le mécanisme décrit ci-après devrait être utilisé.

- obtenir des informations sur des cibles particulières dans un flux. Si un agent ST souhaite vérifier les informations d'un voisin se rapportant à une ou plusieurs cibles spécifiques d'un flux particulier, il devrait générer un message STATUS contenant le SID du flux et un paramètre Liste des cibles faisant la liste des cibles pertinentes. Un agent ST qui reçoit un tel message va d'abord vérifier si le flux et la cible sont connus. Si le flux n'est pas connu, l'agent suit le processus décrit ci-dessus. Si le flux et les cibles sont tous deux connus, l'agent répond par un STATUS-RESPONSE contenant le SID du flux, les bonds IP, la FlowSpec, les membres du groupe (s'il en est), et les cibles demandées qui sont connues. Si le flux est connu mais pas la cible, l'agent répond par un STATUS-RESPONSE contenant le SID du flux, les bonds IP, la FlowSpec, les membres du groupe (s'il en est), mais aucune cible.

Les formats spécifiques des messages STATUS et STATUS-RESPONSE sont définis aux paragraphes 10.4.12 et 10.4.13.

8.5 Estimation du délai d'aller-retour

SCMP est rendu fiable par l'utilisation de la retransmission lorsque un accusé de réception attendu n'est pas reçu à temps. Les

algorithmes de temporisation et de retransmission dépendent des mises en œuvre et sortent du domaine d'application du présent document. Cependant, ils doivent être assez raisonnables pour ne pas causer d'excessives retransmissions des messages SCMP tout en conservant la robustesse du protocole. Les algorithmes sur ce sujet sont décrits dans [WoHD95], [Jaco88], [KaPa87].

La plupart des algorithmes existants se fondent sur une estimation du délai d'aller retour (RTT) entre deux hôtes. Avec SCMP, si un agent ST souhaite avoir une estimation du RTT de et vers un voisin, il devrait générer un message STATUS avec un SID qui a tous ses champs mis à zéro. Un agent ST qui reçoit un message STATUS avec un tel SID devrait répondre aussi rapidement que possible avec un message STATUS-RESPONSE contenant le même SID, et aucune autre information de flux. L'intervalle de temps entre les opérations d'envoi et de réception peut être utilisé comme estimation du RTT de et vers le voisin.

8.6 Découverte de la MTU de réseau

À l'établissement de la connexion, l'application à l'origine demande à l'agent ST local de créer des flux avec certaines exigences de QS. L'agent ST local met sa valeur de MTU de réseau dans le paramètre MaxMsgSize du message CONNECT et le transmet aux agents ST de prochain bond. Chaque agent ST sur le chemin vérifie si sa MTU de réseau est inférieure à celle spécifiée dans le message CONNECT et, si elle l'est, l'agent ST met à jour le champ MaxMsgSize dans le message CONNECT à sa MTU de réseau. Si l'application de la cible décide d'accepter le flux, l'agent ST de la cible copie la valeur de la MTU dans le message CONNECT sur le champ MaxMsgSize du message ACCEPT et le renvoie à l'application à l'origine. Le champ MaxMsgSize dans le message ACCEPT est la MTU minimum des réseaux intervenant jusqu'à cette cible. Si l'application a plusieurs cibles, la MTU minimum du flux est alors la plus petite MaxMsgSize reçue de tous les messages ACCEPT. Il est de la responsabilité de l'application de segmenter ses PDU conformément à la MaxMsgSize minimum du flux car aucune fragmentation de données n'est acceptée durant la phase de transfert des données. Si la MaxMsgSize d'une cible particulière est inacceptable pour une application, elle peut déconnecter la cible du flux et supposer que la cible ne peut pas être acceptée. Lors de l'évaluation de la MaxMsgSize d'une cible particulière, l'application ou son interface d'application devra prendre en compte la taille de l'en-tête de données ST.

8.7 Encapsulation IP de ST

Les paquets ST peuvent être encapsulés dans IP pour leur permettre de passer à travers les routeurs qui ne prennent pas en charge le protocole ST. Bien sûr, la gestion de ressource ST est empêchée sur de tels chemins, et la redondance de paquet est accrue par l'encapsulation, mais si les performances sont raisonnablement prévisibles, cela peut être mieux que pas de communication du tout. Les paquets ST encapsulés dans IP commencent par un en-tête IP normal. La plupart des champs de l'en-tête IP devraient être remplis conformément aux mêmes règles que celles qui s'appliquent à tout autre paquet IP. Trois champs sont particulièrement intéressants :

- Protocole est 5 (voir la [RFC1700]) pour indiquer qu'un paquet ST est inclus, et non TCP ou UDP, par exemple.

- Adresse de destination est celle de l'agent ST du prochain bond. Cela peut être ou non la cible du flux ST. Elle peut être celle d'un agent ST intermédiaire à qui le paquet devrait être acheminé pour tirer parti des garanties de service sur le chemin au delà de cet agent. Un tel agent intermédiaire ne serait pas sur un réseau directement connecté (car alors l'encapsulation IP ne serait pas nécessaire), et ne serait donc pas sur les listes des tableaux d'acheminement normal. Des mécanismes d'acheminement supplémentaires, non définis ici, seront nécessaires pour en savoir plus sur de tels agents.

- Type-de-Service peut être réglé à une valeur appropriée pour le service demandé (voir la [RFC1700]). Cette caractéristique n'est pas mise en œuvre de façon uniforme dans l'Internet, de sorte que son utilisation ne peut être définie ici avec précision.

L'encapsulation ne présente que peu de difficultés pour l'agent ST qui reçoit le paquet. Cependant, lorsque l'encapsulation IP est effectuée, elle doit être faite dans les deux directions. Pour traiter le message encapsulé dans IP, les agents ST retirent simplement l'en-tête IP et traitent l'en-tête ST comme d'habitude.

La partie la plus difficile est durant l'établissement, lorsque l'agent ST doit décider d'encapsuler ou non. Si l'agent ST du prochain bond est sur un réseau distant et si le chemin vers ce réseau passe à travers un routeur qui prend en charge IP mais pas ST, l'encapsulation est alors nécessaire. La fonction d'acheminement donne aux agents ST les informations de chemin et de capacités nécessaires pour prendre l'encapsulation en charge.

À la transmission, l'en-tête IP (presque toujours constant) doit être inséré et la somme de contrôle IP mise à jour de façon appropriée.

Les applications sont informées du nombre de bonds IP traversés sur le chemin de chaque cible. Le champ Bonds IP du

message CONNECT (voir au paragraphe 10.4.4) porte le nombre de bonds IP traversés jusqu'à l'application cible. Le champ est incrémenté par chaque agent ST lorsque l'encapsulation IP est utilisée pour atteindre l'agent ST du prochain bond. Le nombre de bonds IP traversés est retourné à l'origine dans le champ Bonds IP du message ACCEPT (paragraphe 10.4.1).

Lorsque on utilise l'encapsulation IP, le champ MaxMsgSize ne va pas refléter la MTU des segments encapsulés IP. Cela signifie que la fragmentation et le réassemblage IP peuvent être nécessaires dans le nuage IP pour prendre en charge un message de taille MaxMsgSize. La fragmentation IP ne peut survenir que quand la MTU du nuage IP, moins la longueur de l'en-tête IP, est la plus petite MTU dans le chemin de réseau d'un flux.

8.8 Diffusion groupée IP

Si un agent ST doit utiliser l'encapsulation IP pour atteindre plusieurs prochains bonds vers différentes cibles, le paquet doit être dupliqué pour transmission à chaque prochain bond, ou la diffusion groupée IP peut être utilisée si elle est mise en œuvre chez les agents ST de prochain bond et dans les routeurs IP qui interviennent.

Lorsque le flux est établi, la collection des agents ST de prochain bond doit être constituée comme groupe de diffusion groupée IP. L'agent ST doit allouer une adresse de diffusion groupée IP appropriée (voir au paragraphe 10.3.3) et remplir le champ IPMulticastAddress avec cette adresse dans le message CONNECT. L'adresse de diffusion groupée IP dans le message CONNECT est utilisée pour informer les agents ST du prochain bond qu'ils devraient se joindre au groupe de diffusion groupée pour recevoir les PDU suivantes. Évidemment, le message CONNECT lui-même doit être envoyé en utilisant l'envoi individuel. Les agents ST du prochain bond doivent être capables de recevoir à l'adresse de diffusion groupée spécifiée afin d'accepter la connexion.

Si l'agent ST du prochain bond ne peut pas recevoir à l'adresse de diffusion groupée spécifiée, il envoie un message REFUSE avec le code de cause (BadMcastAddress). À réception du REFUSE, l'agent amont peut choisir de réessayer avec une adresse de diffusion groupée différente. Autrement, il peut choisir de perdre l'efficacité de la diffusion groupée et d'utiliser la livraison en envoi individuel.

Les adresses permanentes de diffusion groupée IP ont été allouées à ST :

224.0.0.7 Tous les routeurs ST (agents intermédiaires)
224.0.0.8 Tous les hôtes ST (agents)

De plus, un bloc d'adresses de diffusion groupée IP transitoires, 224.1.0.0 à 224.1.255.255, a été alloué aux groupes de diffusion groupée ST. Par exemple, les trois fonctions suivantes pourraient être disponibles :

- AllocateMcastAddr() -> résultat, McastAddr
- ListenMcastAddr(McastAddr) -> résultat
- ReleaseMcastAddr(McastAddr) -> résultat

9. Spécification du flux ST2+

La présente section définit la spécification du flux ST2+. La spécification du flux contient les exigences d'application d'utilisateur en termes de qualité de service. Son contenu dépend du LRM et est transparent pour le protocole d'établissement de ST2. ST2 porte la spécification de flux au titre du paramètre FlowSpec, qui est décrit au paragraphe 10.3.1. La spécification de flux ST2+ exigée n'est incluse dans le protocole que pour les besoins d'interopérabilité. ST2+ définit aussi une spécification de flux "nul" à n'utiliser que pour la prise en charge des essais.

ST2 ne dépend pas d'un format de spécification de flux particulier et il est prévu que d'autres versions de la spécification de flux seront nécessaires à l'avenir. Les différents formats de spécification de flux sont distingués par la valeur du champ Version du paramètre FlowSpec (voir au paragraphe 10.3.1). Un seul flux est toujours associé à un seul format de spécification de flux, c'est-à-dire que le champ Version est cohérent tout au long du flux entier. Les valeurs du champ Version sont définies :

0 – FlowSpec nul /* doit être pris en charge */
1 – ST version 1
2 – ST version 1.5
3 – FlowSpec de la RFC 1190
4 – FlowSpec HeiTS
5 – FlowSpec BerKom
6 – FlowSpec de la RFC 1363
7 – FlowSpec ST2+ /* doit être pris en charge */

Les FlowSpec des versions n° 0 et n° 7 doivent être prises en charge par les mises en œuvre de ST2+. Les numéros de version

dans la gamme 1 à 6 indiquent que les spécifications de flux sont actuellement utilisées dans les mises en œuvre existantes de ST2. Les valeurs dans la gamme 128 à 255 sont réservée pour les utilisations privées et expérimentales.

En général, une spécification de flux peut prendre en charge des descriptions de flux sophistiquées. Par exemple, une spécification de flux peut représenter des sous-flux d'un flux particulier. Cela pourrait alors être utilisé par une application coopérative et un LRM pour transmettre des paquets précis à des cibles spécifiques sur la base des différents sous-flux. Les bits réservés dans la PDU de données ST2 (voir au paragraphe 10.1) peuvent être utilisés avec une telle spécification de flux pour désigner les paquets associés aux différents sous-flux. La FlowSpec ST2+ n'est pas aussi sophistiquée, et elle est destinée à être utilisée avec des applications qui génèrent du trafic à un seul débit pour une livraison uniforme à toutes les cibles.

9.1 FlowSpec version n° 0 - (FlowSpec nul)

La spécification de flux identifiée par une valeur n° 0 du champ Version est appelée la FlowSpec nulle. Cette spécification de flux ne cause l'allocation d'aucune ressource. Elle est ignorée par les LRM. Son contenu n'est jamais mis à jour. L'établissement du flux se fait de la façon usuelle ce qui conduit à un établissement réussi du flux, mais aucune ressource n'est en fait réservée.

L'objet de la FlowSpec nulle est de faciliter les essais d'interopérabilité en permettant de construire des flux sans réellement allouer la quantité de ressources correspondante. La FlowSpec nulle peut aussi être utilisée à des fins d'essais et de débogage.

La FlowSpec nulle comporte seulement le paramètre FlowSpec de quatre octets (voir au paragraphe 10.3.1). Le troisième octet (champ Version) doit être mis à 0.

9.2 FlowSpec version n° 7 - FlowSpec ST2+

La spécification de flux identifiée par la valeur n° 7 du champ Version est la FlowSpec ST2+, à utiliser par toutes les mises en œuvre de ST2+. Elle permet aux applications d'utilisateur d'exprimer leurs exigences de temps réel sous la forme d'une classe de QS, de préséance, et des trois paramètres de QS de base :

- o taille de message,
- o débit de message,
- o délai de bout en bout.

La classe de QS indique quelles sortes de garanties de QS sont attendues par l'application, par exemple, des garanties strictes ou des prévisions (voir au paragraphe 9.2.1). les paramètres de QS sont exprimés via un ensemble de valeurs :

- o les valeurs "désirées" indiquent la QS désirée par l'application. Ces valeurs sont allouées par l'application et jamais modifiées par le LRM.
- o les valeurs "limite" indiquent la plus basse QS que l'application veut accepter. Ces valeurs sont aussi allouées par l'application et jamais modifiées par le LRM.
- o les valeurs "réelles" indiquent la QS que le système est capable de fournir. Elles sont mises à jour par le LRM à chaque nœud. Les valeurs "réelles" sont toujours bordées par les valeurs "limite" et "désirées".

9.2.1 Classes de qualité de service

Deux classes de QS sont définies :

```
        1 - QOS_PREDICTIVE      /* valeur du champ QoSClass = 0x01, doit être prise en charge */
2 - QOS_GUARANTEED              /* valeur du champ QoSClass = 0x10, facultative */
```

- o La classe QOS_PREDICTIVE implique que la QS négociée peut être outrepassée pour de courts intervalles de temps durant le transfert des données. Une application doit fournir des valeurs qui tiennent compte du cas "normal", par exemple, le débit de message "désiré" est le débit alloué pour la transmission. Les réservations sont faites pour le cas "normal" par opposition au cas de pointe exigé par la classe de service QOS_GUARANTEED. Cette classe de QS doit être prise en charge par toutes les mises en œuvre.

- o La classe QOS_GUARANTEED implique que la QS négociée pour le flux n'est jamais outrepassée durant le transfert des données. Une application doit fournir des valeurs qui tiennent compte du pire cas possible, par exemple, le débit de message "désiré" est le débit de pointe pour la transmission. Il en résulte que des ressources suffisantes pour traiter le débit de pointe sont réservées. Cette stratégie peut conduire à une surconsommation de ressources, mais donne des garanties strictes de temps réel. La prise en charge de cette classe de QS est facultative.

Si un LRM qui ne prend pas en charge la classe QOS_GUARANTEED reçoit une FlowSpec contenant la classe

QOS_GUARANTEED, il informe l'agent ST local. L'agent ST peut essayer des chemins différents ou supprimer la portion correspondante du flux comme décrit au paragraphe 5.5.3, c'est-à-dire avec le code de cause (FlowSpecError).

9.2.2 Préséance

Préséance est l'importance de la connexion à établir. Zéro représente la plus faible préséance. Le plus bas niveau est supposé être utilisé par défaut. En général, la distinction entre préséance et priorité est que la préséance spécifie les flux qui ont la permission de prendre des ressources précédemment engagées pour un autre flux, alors que priorité identifie les PDU qu'un flux sera le plus enclin à abandonner.

9.2.3 Taille maximum de données

Ce paramètre est exprimé en octets. Il représente la quantité maximum de données, à l'exclusion des en-têtes ST et autres, qu'il est permis d'envoyer dans les messages au titre du flux. Le LRM vérifie d'abord si il est possible d'obtenir la valeur désirée par l'application (DesMaxSize). Sinon, il met à jour la valeur réelle (ActMaxSize) avec la taille disponible sauf si cette valeur est inférieure au minimum permis par l'application (LimitMaxSize), auquel cas il informe l'agent ST local qu'il n'est pas possible de construire le flux le long de ce chemin.

9.2.4 Débit de message

Ce paramètre est exprimé en messages par seconde. Il représente le débit de transmission pour le flux. Le LRM vérifie d'abord si il est possible d'obtenir la valeur désirée par l'application (DesRate). Sinon, il met à jour la valeur réelle (ActRate) avec le débit disponible sauf si cette valeur est inférieure au minimum permis par l'application (LimitRate), auquel cas il informe l'agent ST local qu'il n'est pas possible de construire le flux le long de ce chemin.

9.2.5 Délai et gigue de délai

Le paramètre Délai s'exprime en millisecondes. Il représente le délai maximum de bout en bout pour le flux. Le LRM vérifie d'abord si il est possible d'obtenir la valeur désirée par l'application (DesMaxDelay). Sinon, il met à jour la valeur réelle (ActMaxDelay) avec le délai disponible sauf si cette valeur est supérieure au délai maximum admis par l'application (LimitMaxDelay) auquel cas il informe l'agent ST local qu'il n'est pas possible de construire le flux le long de ce chemin.

Le LRM met aussi à jour à chaque nœud le champ MinDelay en l'incrémentant du délai minimum possible vers le prochain bond. Les informations sur le délai minimum possible permettent de calculer la gamme maximale de délais de bout en bout, c'est-à-dire l'intervalle de temps dans lequel un paquet de données peut être reçu. Cet intervalle ne devrait pas excéder la valeur de DesMaxDelayRange indiquée par l'application. La gamme maximale de délais de bout en bout est une limite supérieure de la gigue de délai.

9.2.6 Format de FlowSpec ST2+

La FlowSpec ST2+ a le format suivant :

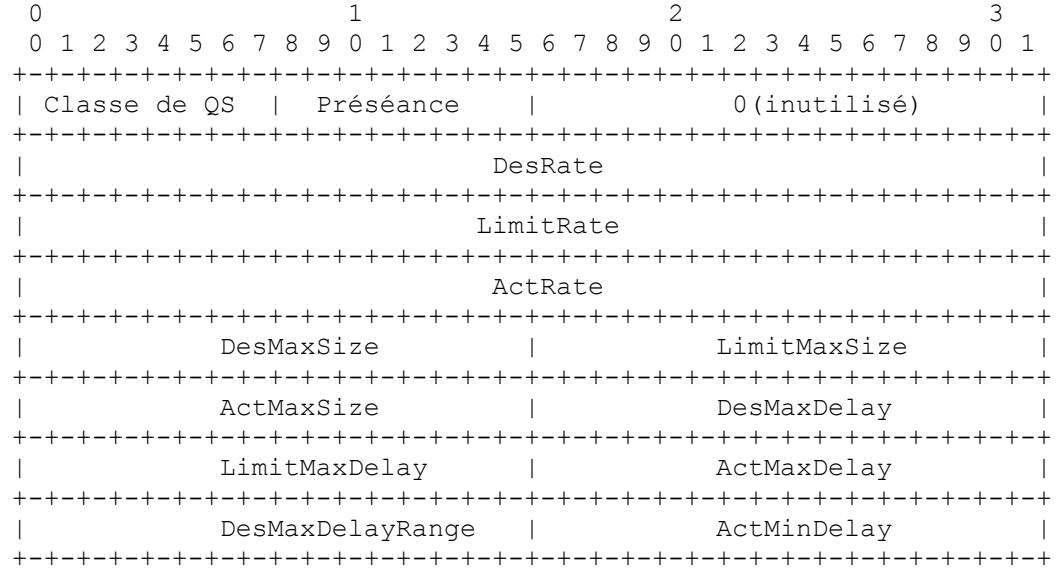


Figure 9 : FlowSpec ST2+

Le LRM ne modifie que les champs "réels", c'est-à-dire, ceux qui commencent par "Act". L'application d'utilisateur alloue les valeurs à tous les autres champs.

- Classe de QS indique laquelle des deux classes de service définies s'applique. Les deux classes sont : QOS_PREDICTIVE (Classe de QS = 1) et QOS_GUARANTEED (Classe de QS = 2).
- Préséance indique la préséance du flux. Zéro représente la plus faible préséance, et devrait être la valeur par défaut.
- DesRate est le taux de transmission désiré pour le flux en messages/seconde. Ce champ est réglé par l'origine et n'est pas modifié par les agents intermédiaires.
- LimitRate est le minimum acceptable du taux de transmission en messages/seconde. Ce champ est réglé par l'origine et n'est pas modifié par les agents intermédiaires.
- ActRate est le taux de transmission réel alloué pour le flux en messages/seconde. Chaque agent met à jour ce champ avec le taux disponible sauf si cette valeur est inférieure à LimitRate, auquel cas un REFUSE est généré.
- DesMaxSize est la taille maximum de données désirée en octets qui seront envoyés dans un message au sein du flux. Ce champ est réglé par l'origine.
- LimitMaxSize est la taille maximum acceptable des données en octets. Ce champ est réglé par l'origine.
- ActMaxSize est la taille réelle maximum des données qui peuvent être envoyées dans un message au sein du flux. Ce champ est mis à jour par chaque agent sur la base de la MTU et des ressources disponibles. Si la taille maximum disponible est inférieure à LimitMaxSize, la connexion doit être refusée avec le code de cause (CantGetResrc).
- DesMaxDelay est le délai maximum de bout en bout désiré pour le flux en millisecondes. Ce champ est réglé par l'origine.
- LimitMaxDelay est la limite supérieure du délai de bout en bout acceptable pour le flux en millisecondes. Ce champ est réglé par l'origine.
- ActMaxDelay est le délai maximum de bout en bout qui sera vu par les données dans le flux. Chaque agent ST ajoute à ce champ le délai maximum qui sera introduit par l'agent, y compris le temps de transmission à l'agent ST du prochain bond. Si le maximum réel excède LimitMaxDelay, la connexion est alors refusée avec le code de cause (CantGetResrc).
- DesMaxDelayRange est la gamme de délai désiré maximum qui peut être rencontrée de bout en bout par les données du flux en millisecondes. Cette valeur est réglée par l'application à l'origine.
- ActMinDelay est le délai minimum réel de bout en bout qui sera rencontré par les données du flux en millisecondes. Chaque agent ST ajoute à ce champ le délai minimum qui sera introduit par l'agent, y compris le temps de transmission vers l'agent ST du prochain bond. Chaque agent doit ajouter au moins 1 milliseconde. La gamme de délai pour le flux peut être calculée à partir des champs Délai réel maximum et minimum. On suppose que cette gamme sera importante pour certaines applications.

10. Spécification des unités de données de protocole ST2

10.1 PDU de données

Les paquets IP et ST peuvent être distingués par le champ Numéro de version IP, c'est-à-dire, les quatre (4) premiers bits du paquet ; la valeur 5 a été allouée à ST (voir la [RFC1700]). Il n'y a pas d'exigence pour la compatibilité entre les en-têtes de paquet IP et ST au delà de ces quatre premiers bits. (IP utilise la valeur 4.)

Les PDU de ST envoyées entre les agents ST consistent en un en-tête ST encapsulant soit une PDU de couche supérieure soit un message de contrôle ST. Les paquets de données sont distingués des messages de contrôle via le bit D (bit 8) dans l'en-tête ST.

L'en-tête ST comporte aussi un numéro de version ST, un champ de longueur totale, une somme de contrôle d'en-tête, un identifiant unique, et l'adresse IP de l'origine du flux de 32 bits. L'identifiant unique et l'adresse IP de 32 bits de l'origine du flux forment l'identifiant du flux (SID). Ceci est montré à la Figure 10. Prière de se référer au paragraphe 10.6 pour une explication de la notation.

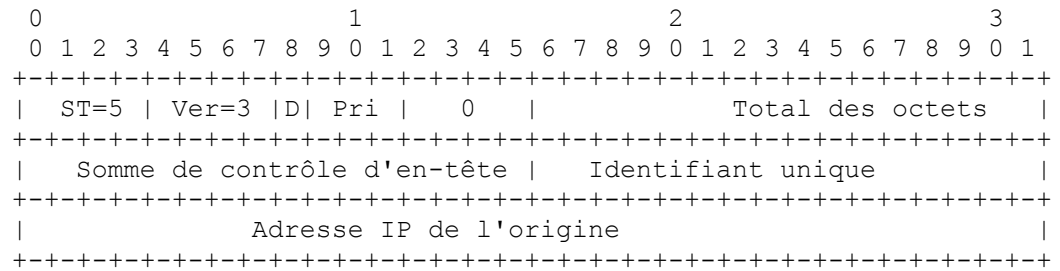


Figure 10 : En-tête ST

- ST est le numéro de version IP alloué pour identifier les paquets ST. La valeur pour ST est 5.
- Ver est le numéro de version ST. La valeur pour la version ST2+ actuelle est 3.
- D (bit 8) est mis à 1 dans tous les paquets de données ST et à 0 dans tous les messages de contrôle SCMP.
- Pri (bits 9-11) est le champ de priorité d'abandon de paquet avec zéro (0) pour la plus faible priorité et sept pour la plus forte. Le champ est à utiliser comme décrit au paragraphe 3.2.2.
- Total des octets est la longueur en octets, du paquet ST entier, il inclut l'en-tête ST mais n'inclut aucun en-tête ou en-queue local. En général, tous les champs de longueur dans le protocole ST sont en unités d'octets.
- Somme de contrôle d'en-tête ne couvre que l'en-tête ST (12 octets). Le protocole ST utilise des sommes de contrôle de 16 bits dans l'en-tête ST et dans chaque message de contrôle. Pour le calcul de la somme de contrôle, voir au paragraphe 8.3.
- Identifiant unique est le premier élément de l'identifiant de flux (SID). Il est localement unique à l'origine du flux (voir au paragraphe 8.1).
- Adresse IP de l'origine est le second élément du SID. Ce sont les 32 bits de l'adresse IP l'origine du flux (voir au paragraphe 8.1).

Les bits 12 à 15 doivent être mis à zéro (0) lors de l'utilisation des spécifications de flux définies dans ce document (voir la Section 9). Ils peuvent être réglés conformément à d'autres spécifications de flux si elles sont utilisées, par exemple, comme décrit dans [WoHD95].

10.1.1 Paquets de données ST

Les paquets ST dont le bit D n'est pas à zéro sont des paquets de données. Leur interprétation est l'affaire des protocoles de couche supérieure et par conséquent n'est pas spécifiée ici. Les paquets de données ne sont pas protégés par une somme de contrôle ST et seront livrés au protocole de couche supérieure même avec des erreurs. Les agents ST ne passeront pas de paquets de données sur un nouveau bond si leur établissement n'est pas achevé.

10.2 PDU de contrôle

Les messages de contrôle SCMP sont échangés entre des agents ST voisins en utilisant un bit D de zéro (0). Le protocole de contrôle suit un modèle de demande-réponse où toutes les demandes appellent une réponse. La retransmission après expiration d'un temporisateur (voir au paragraphe 4.3) est utilisée pour récupérer les messages perdus ou ignorés. Les messages de contrôle n'étendent pas les limites du paquet ; si un message de contrôle est trop long pour la MTU d'un bond, ses informations sont partagées et un message de contrôle par partition est envoyé (voir au paragraphe 5.1.2). Tous les messages de contrôle ont le format suivant :

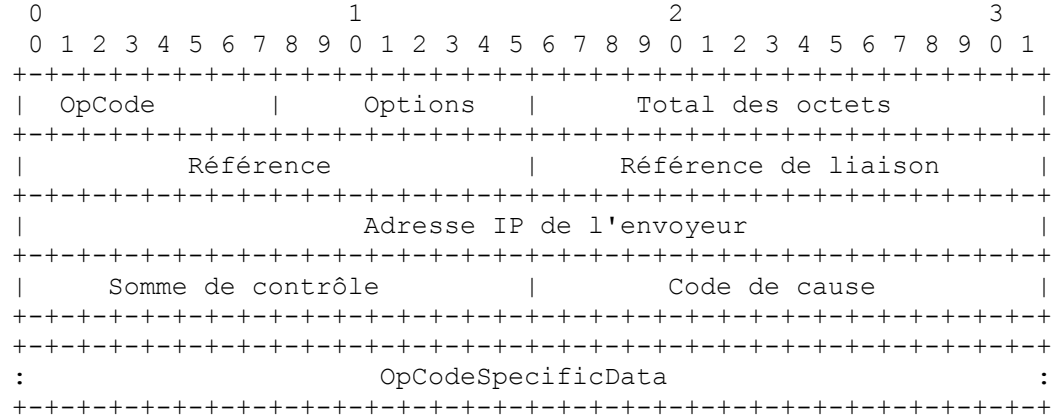


Figure 11 : Format de message de contrôle ST

- OpCode identifie le type de message de contrôle.
- Options est utilisé pour porter des variations spécifiques de l'OpCode pour un message de contrôle.
- Total des octets est la longueur du message de contrôle, en octets, y compris tous les champs spécifiques de l'OpCode et les paramètres facultatifs. La valeur est toujours divisible par quatre (4).
- Référence est un numéro de transaction. Chaque envoyeur d'un message de contrôle d'une demande alloue un numéro de référence au message qui est unique par rapport au flux. Le numéro de référence est utilisé par le receveur pour détecter et éliminer les duplications. Chaque accusé de réception porte le numéro de référence de la demande dont il est accusé réception. La référence zéro (0) n'est jamais utilisée, et les numéros de référence sont supposés suivre un accroissement monotone qui revient à zéro de sorte que les relations plus ancien que et plus récent que sont bien définies.
- Référence de liaison contient le champ Référence du message de contrôle de la demande qui a causé la création de ce

message de contrôle de demande. Il est utilisé dans des situations où une seule demande conduit à plusieurs réponses du même agent ST. Par exemple des messages CONNECT et CHANGE qui reçoivent d'abord un accusé de réception bond par bond ce qui conduit ensuite à une réponse ACCEPT ou REFUSE de la part de chaque cible.

- o Adresse IP de l'envoyeur est l'adresse IP de 32 bits de l'interface réseau que l'agent ST a utilisée pour envoyer le message de contrôle. Cette valeur change chaque fois que le paquet est transmis par un agent ST (bond par bond).
- o Somme de contrôle est la somme de contrôle du message de contrôle. Comme les messages de contrôle sont envoyés dans des paquets qui peuvent être livrés avec des bits erronés, chaque message de contrôle doit être vérifié afin qu'il soit sans erreur avant d'être traité.
- o Code de cause est mis à zéro (0 = Pas d'erreur) dans la plupart des messages SCMP. Autrement, il peut être réglé à une valeur appropriée pour indiquer une situation d'erreur comme défini au paragraphe 10.5.3.
- o OpCodeSpecificData contient toutes informations supplémentaires qui sont associées au message de contrôle. Il dépend du message de contrôle spécifique et est expliqué plus en détails ci-dessous. Dans certains messages de contrôle de réponse, les champs de zéro (0) sont inclus pour permettre que le format corresponde à celui du message de demande correspondant. Le champ OpCodeSpecificData peut aussi contenir des paramètres facultatifs. Les spécificités de OpCodeSpecificData sont définies au paragraphe 10.3.

10.3 Éléments SCMP communs

Plusieurs champs et paramètres (qu'on appellera de façon générique des éléments) sont communs à deux PDU ou plus. Ils sont décrits en détail ici au lieu de répéter leur description plusieurs fois. Dans de nombreux cas, la présence d'un paramètre est facultative. Pour permettre de définir et analyser facilement les paramètres, chacun est identifié par un octet PCode suivi par un octet PBytes qui indiquent la longueur du paramètre en octets (y compris le PCode, le PByte, et tous octets de bourrage). Si la longueur des informations n'est pas un multiple de quatre (4) octets, le paramètre est bourré avec de un à trois octets de zéros (0). PBytes est donc toujours un multiple de quatre (4). Les paramètres peuvent être présents dans n'importe quel ordre.

10.3.1 FlowSpec

Le paramètre FlowSpec (PCode = 1) est utilisé dans plusieurs messages SCMP pour porter la spécification de flux ST2. Le paramètre FlowSpec a le format suivant :

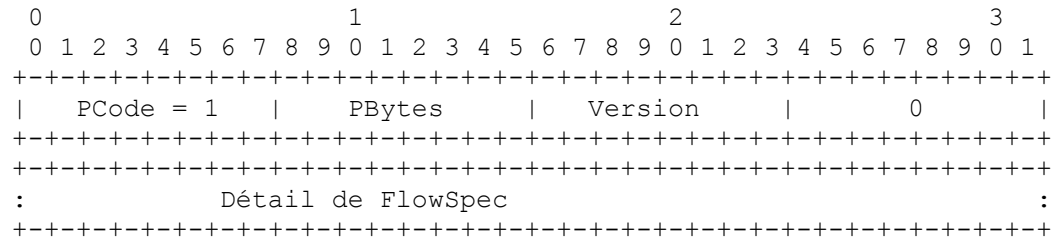


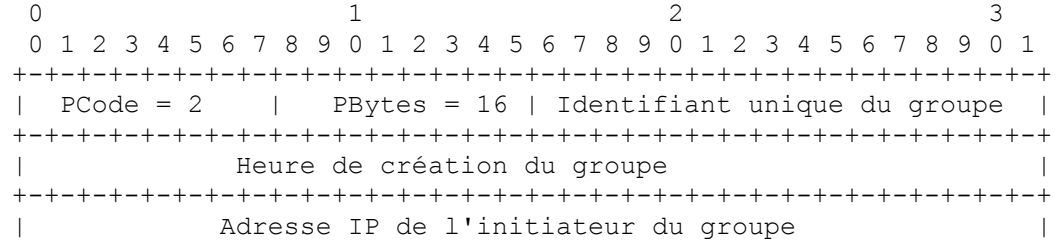
Figure 12 : Paramètre de FlowSpec

- o Le champ Version contient la version de la FlowSpec.
- o Le champ Détail de FlowSpec contient la spécification de flux et elle est transparente pour l'agent ST. C'est la structure des données à passer au LRM. Il doit être aligné sur quatre octets.

La FlowSpec nulle (voir au paragraphe 9.1) n'a pas de champ Détail de FlowSpec. Le PBytes est mis à quatre (4), et Version est mis à zéro (0). La FlowSpec ST2+ (voir au paragraphe 9.2) est une structure de données de 32 octets. PBytes est mis à 36, et Version est mis à sept (7).

10.3.2 Groupe

Le paramètre Groupe (PCode = 2) est un argument facultatif utilisé pour indiquer que le flux est membre du groupe spécifié.



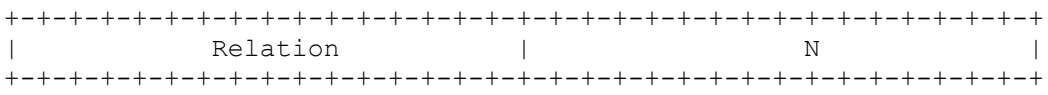


Figure 13 : Paramètre de groupe

- Identifiant unique du groupe, Adresse IP de l'initiateur du groupe et Heure de création du groupe forment ensemble le champ Nom de groupe. Ils sont alloués par la fonction de générateur de nom de groupe (voir au paragraphe 8.2). Identifiant unique du groupe et Heure de création du groupe sont spécifiques de la mise en œuvre et n'ont de définitions que locales.
- Relation a le format suivant :

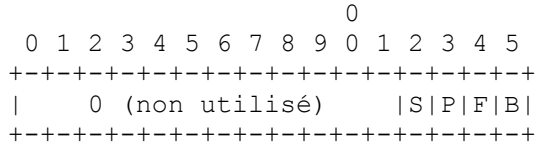


Figure 14 : Champ Relations

Les bits S, P, F et B correspondent respectivement à partage de ressources de sous-réseau, chemin *(Path)*, sort *(Fate)*, et Bande passante (voir la Section 7). Une valeur de 1 indique que la relation existe pour ce groupe. Toutes les combinaisons des quatre bits sont permises. Les bits 0 à 11 du champ Relation sont réservés pour une utilisation future et doivent être mis à 0.

- N ne contient une valeur légale que si le bit B est établi *(à 1)*. C'est la valeur du paramètre N à utiliser comme expliqué au paragraphe 7.1.1.

10.3.3 Adresse de diffusion groupée

Le paramètre Adresse de diffusion groupée (PCode = 3) est un paramètre facultatif qui est utilisé avec l'encapsulation IP et l'établissement d'un groupe de diffusion groupée IP. Ce paramètre est utilisé pour communiquer l'adresse de diffusion groupée IP désirée aux agents ST de prochain bond qui devraient devenir membres du groupe (voir au paragraphe 8.8).

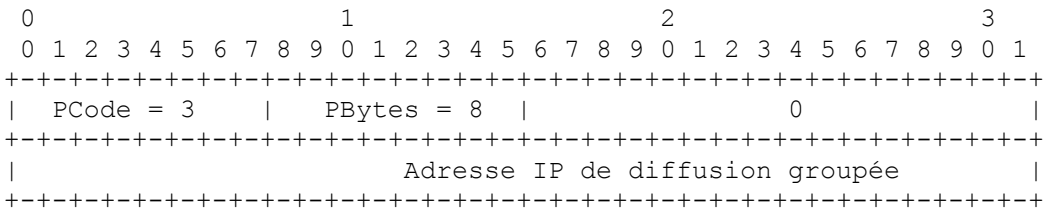


Figure 15 : Adresse de diffusion groupée

- Adresse IP de diffusion groupée est l'adresse de diffusion groupée IP de 32 bits à utiliser pour recevoir les paquets de données pour le flux.

10.3.4 Origine

Le paramètre Origine (PCode = 4) est utilisé pour identifier le prochain protocole de couche supérieure, et le SAP utilisé en conjonction avec ce protocole.

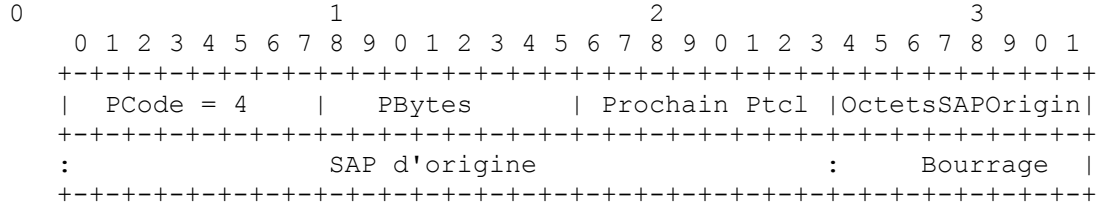


Figure 16 : Origine

- Prochain Ptcl est un champ de 8 bits utilisé dans les opérations de démultiplexage pour identifier le protocole à utiliser au

dessus de ST. Les valeurs de Prochain Ptcl sont dans le même espace de nombres que le champ Protocole de l'en-tête IP et sont par conséquent définies dans la RFC "Numéros alloués" [RFC1700].

- o OctetsSAPOrigin spécifie la longueur du SAP d'origine, à l'exclusion de tout bourrage nécessaire pour conserver l'alignement sur 32 bits.
- o SAP d'origine identifie le SAP de l'origine qui est associé au prochain protocole.

Noter que les 32 bits de l'adresse IP de l'origine du flux ne sont pas inclus dans ce paramètre parce que ils sont toujours disponibles au titre de l'en-tête ST.

10.3.5 RecordRoute

Le paramètre RecordRoute (PCode = 5) est utilisé pour demander que le chemin entre l'origine et la cible soit enregistré et livré à l'application d'utilisateur. L'agent ST à l'origine (ou à la cible) incluant ce paramètre, doit déterminer la longueur du paramètre, indiquée par le champ PBytes. Les agents ST qui traitent les messages contenant ce paramètre ajoutent leur adresse IP de réception dans la position indiquée dans l'espace permis par le champ Décalage libre. Si aucun espace n'est disponible, le paramètre est passé inchangé. Lorsque il est inclus par l'origine, tous les agents entre l'origine et la cible ajoutent leurs adresses IP et cette information est fournie à l'application de la cible. Lorsque elle est incluse par la cible, tous les agents entre la cible et l'origine, incluse, ajoutent leurs adresses IP et cette information est fournie à l'application à l'origine.

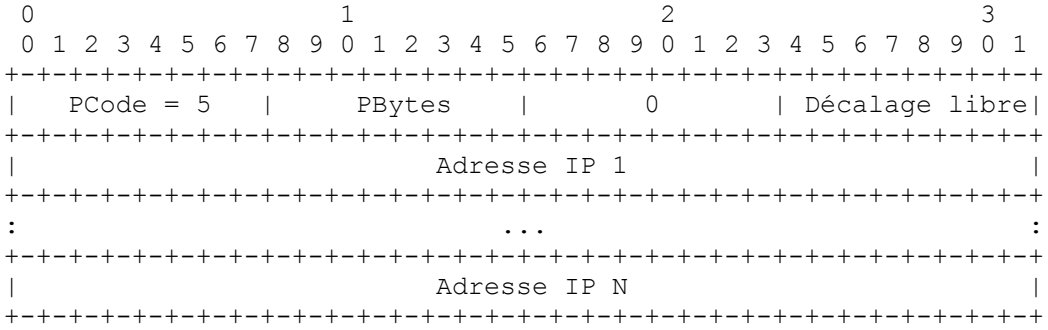


Figure 17 : RecordRoute

- o PBytes est la longueur du paramètre en octets. La longueur est déterminée par l'agent (cible ou origine) qui introduit le premier le paramètre. Une fois établie, la longueur du paramètre reste inchangée.
- o Décalage libre indique le décalage par rapport au début du paramètre, pour la prochaine adresse IP à enregistrer. Lorsque le Décalage libre est supérieur ou égal à PBytes, le paramètre RecordRoute est plein.
- o Adresse IP est rempli, dans l'espace permis, par chaque agent ST qui traite ce paramètre.

10.3.6 Cible et Liste de cibles

Plusieurs messages de contrôle utilisent un paramètre appelé Liste des cibles (PCode = 6), qui contient des informations sur les cibles auxquelles le message appartient. Pour chaque cible de la Liste des cibles, les informations comportent les 32 bits de l'adresse IP de la cible, le SAP applicable au prochain protocole de couche supérieure, et la longueur du SAP (Octets SAP). Par conséquent, la structure d'une cible peut être de longueur variable. Chaque entrée a le format indiqué à la Figure 18.

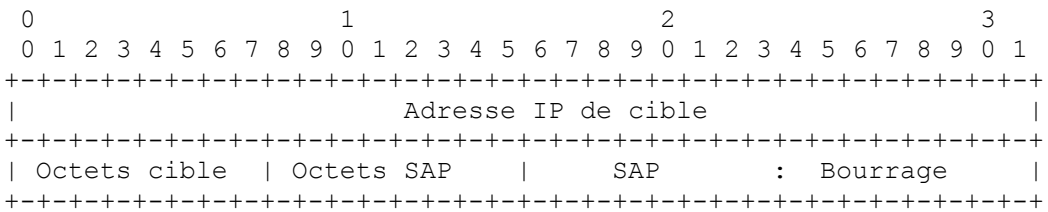


Figure 18 : Cible

- o Adresse IP de cible est les 32 bits de l'adresse IP de la cible.
- o Octets cible est la longueur de la structure cible, commençant par l'Adresse IP de cible.
- o Octets SAP est la longueur du SAP, à l'exclusion de tout bourrage requis pour conserver l'alignement à 32 bits.
- o SAP peut être plus long que 2 octets et il comporte un bourrage si nécessaire. Il n'y aura pas de bourrage pour les SAP qui font des longueurs de 2, 6, 10, etc., octets.

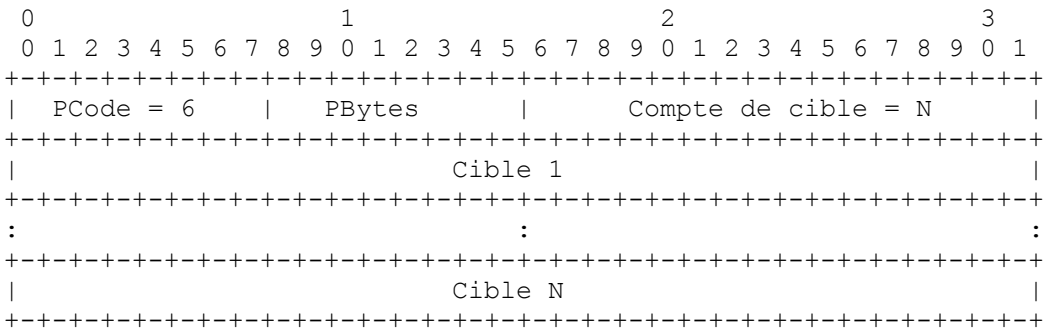


Figure 19 : Liste de cibles

10.3.7 Données d'utilisateur

Le paramètre Données d'utilisateur (PCode = 7) est un paramètre facultatif qui peut être utilisé par le protocole de couche supérieure ou une application pour porter des informations arbitraires à ses homologues. Ce paramètre est propagé dans des messages de contrôle et son contenu n'a pas de signification pour les agents ST. Noter que comme la taille des messages de contrôle est limitée par la plus petite MTU sur le chemin vers les cibles, la taille maximum de ce paramètre ne peut pas être spécifiée à priori. Si la taille de ce paramètre est cause qu'un message excède la MTU du réseau, un agent ST se comporte comme décrit au paragraphe 5.1.2. Le paramètre doit être bourré à un multiple de 32 bits.

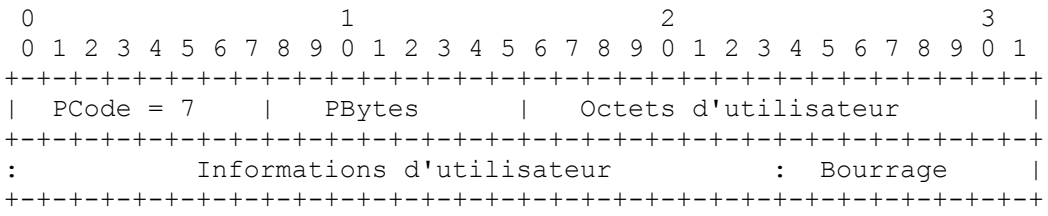


Figure 20 : Données d'utilisateur

- o Octets d'utilisateur spécifie le nombre d'octets valides d'informations d'utilisateur.
- o Informations d'utilisateur sont des données arbitraires significatives pour le protocole ou application de la prochaine couche supérieure.

10.3.8 Traitement des paramètres indéfinis

Un agent ST doit être capable de traiter tous les paramètres dont la liste figure ci-dessus. Pour prendre en charge les possibles utilisations futures, les paramètres avec des PCodes inconnus doivent aussi être pris en charge. Si un agent reçoit un message qui contient un paramètre avec une valeur de Pcode inconnue, l'agent devrait traiter le paramètre comme si il était un paramètre Données d'utilisateur. C'est-à-dire que le contenu du paramètre devrait être ignoré, et le message devrait être propagé, comme approprié, avec le message de contrôle qui s'y rapporte.

10.4 PDU de message de contrôle ST

Les messages de contrôle ST sont décrits dans les paragraphes suivants. Prière de se reporter au paragraphe 10.6 pour une explication de la notation.

10.4.1 ACCEPT

ACCEPT (OpCode = 1) est produit par une cible comme réponse positive à un message CONNECT. Il implique que la cible est prête à accepter des données provenant de l'origine avec le flux qui a été établi par le CONNECT. ACCEPT est aussi produit comme réponse positive à un message CHANGE. Il implique que la cible accepte la modification de flux proposée.

ACCEPT est relayé par les agents ST de la cible à l'origine le long du chemin établi par CONNECT (ou CHANGE) mais dans la direction inverse. ACCEPT doit recevoir un accusé de réception avec ACK à chaque bond.

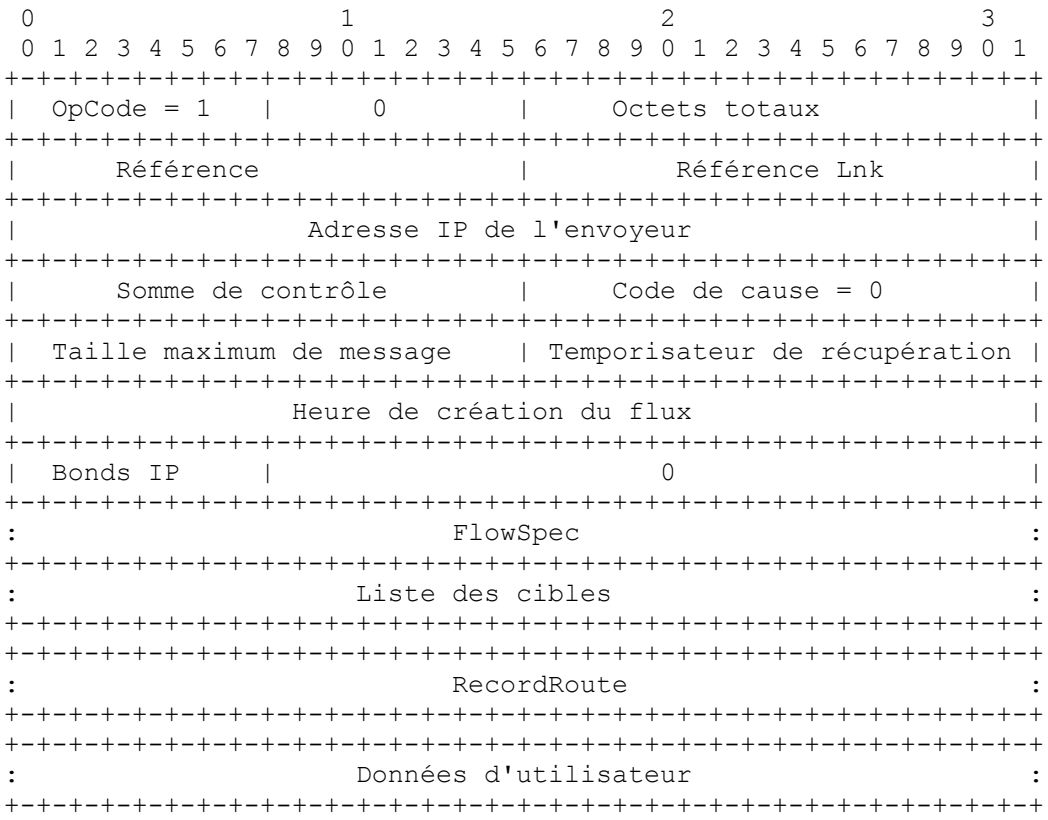


Figure 21 : Message de contrôle ACCEPT

- Référence contient un numéro alloué par l'agent ST qui envoie ACCEPT pour l'utiliser dans le ACK d'accusé de réception
- Référence Lnk est le numéro de Référence provenant du CONNECT (ou CHANGE) correspondant.
- Taille maximum de message indique la plus petite MTU le long du chemin traversé par le flux. Ce champ n'est établi que lors d'une réponse à une demande CONNECT.
- Temporisateur de récupération reflète le nombre nominal de millisecondes pendant lequel l'application accepte d'attendre qu'un composant système défaillant soit détecté et qu'une action corrective soit prise. Ce champ représente ce qui peut réellement être accepté par chaque agent participant, et n'est établi qu'en réponse à une demande CONNECT.
- Heure de création du flux est l'horodatage de 32 bits dépendant du système qui est copié de la demande CONNECT correspondante.
- Bonds IP est le nombre de bonds IP encapsulés traversés par le flux. Ce champ est mis à zéro par l'origine, et il est incrémenté par chaque agent IP encapsulant.

10.4.2 ACK

ACK (OpCode = 2) est utilisé pour accuser réception d'une demande. Le message ACK n'est pas propagé au delà du bond précédent ou de l'agent ST du prochain bond.

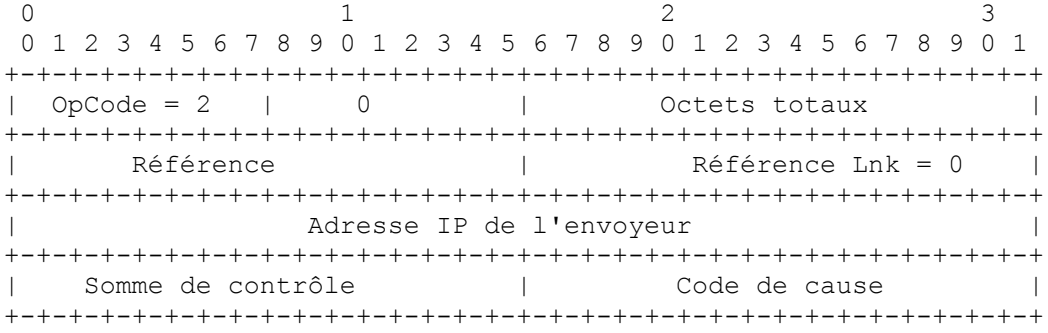


Figure 22 : Message de contrôle ACK

- Référence est le numéro de référence du message de contrôle dont il est accusé réception.
- Code de cause est normalement Pas d'erreur, mais d'autres possibilités existent, par exemple, DuplicateIgn.

10.4.3 CHANGE

CHANGE (OpCode = 3) est utilisé pour changer la FlowSpec du flux établi. Le message CHANGE est traité de la même façon que CONNECT, excepté qu'il voyage le long du chemin d'un flux établi. CHANGE doit être propagé jusqu'à ce qu'il atteigne les cibles du flux auquel il se rapporte. CHANGE doit recevoir un accusé de réception ACK à chaque bond.

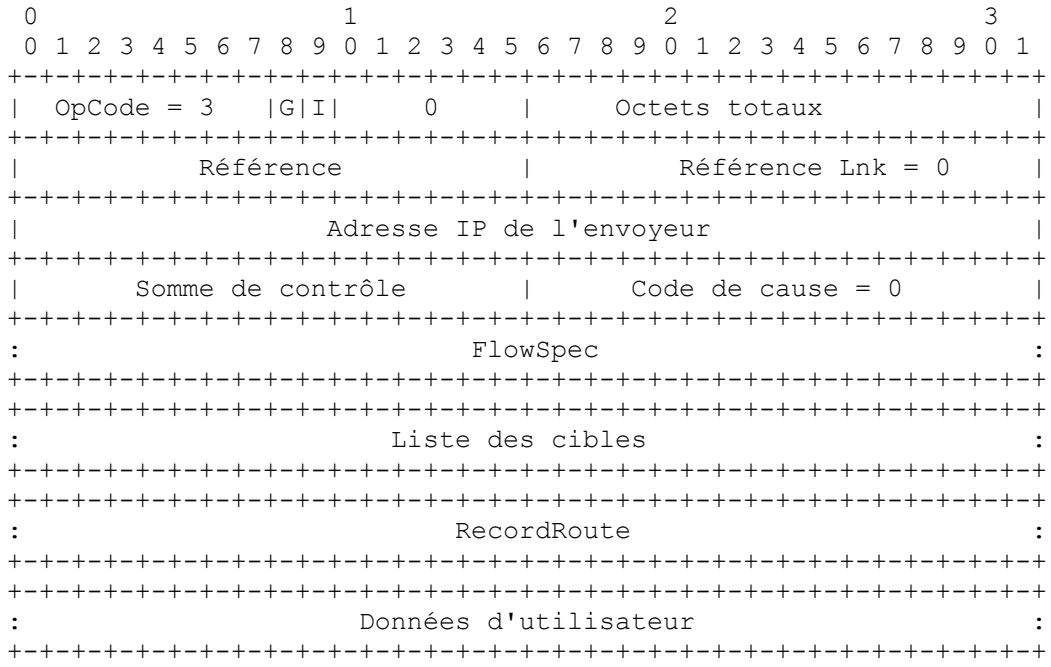


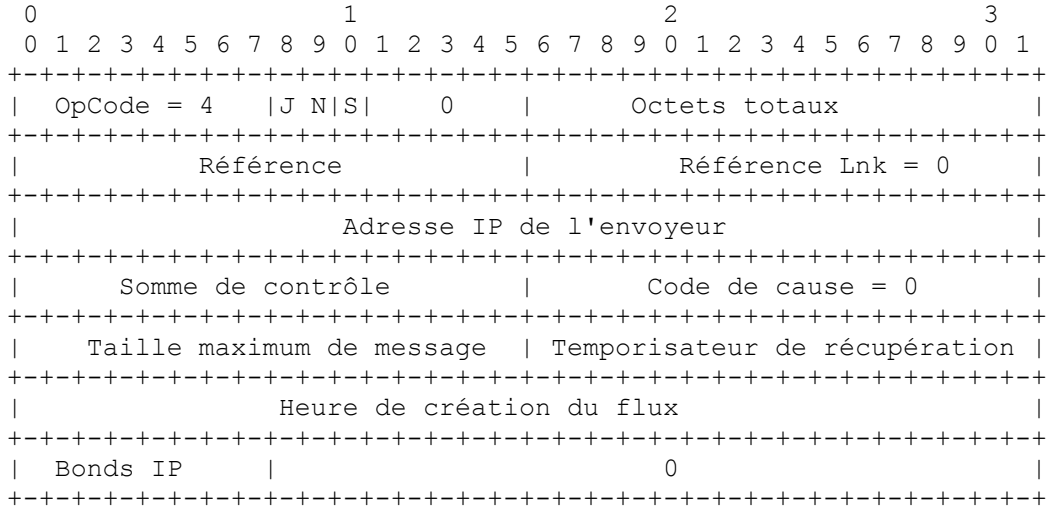
Figure 23 : Message de contrôle CHANGE

- o G (bit 8) est utilisé pour demander un changement global, au niveau du flux ; le paramètre Liste des cibles devrait être omis lorsque le bit G est spécifié.
- o I (bit 7) est utilisé pour indiquer qu'il est permis au LRM d'interrompre, et si nécessaire de casser le flux pour essayer de satisfaire au changement demandé.
- o Référence contient un numéro alloué par l'agent ST qui envoie CHANGE à utiliser dans le ACK d'accusé de réception.

10.4.4 CONNECT

CONNECT (OpCode = 4) demande l'établissement d'un nouveau flux ou un ajout ou une récupération d'un flux existant. Seule l'origine peut émettre l'ensemble initial des CONNECT pour établir un flux, et le premier CONNECT à chaque prochain bond est utilisé pour porter le SID.

Le prochain bond répond initialement avec un ACK, ce qui implique que le CONNECT était valide et est en cours de traitement. Le prochain bond va ensuite servir de relais vers l'amont pour le ACCEPT ou REFUSE provenant de chaque cible. Un agent ST intermédiaire qui reçoit un CONNECT se comporte comme expliqué au paragraphe 4.5.



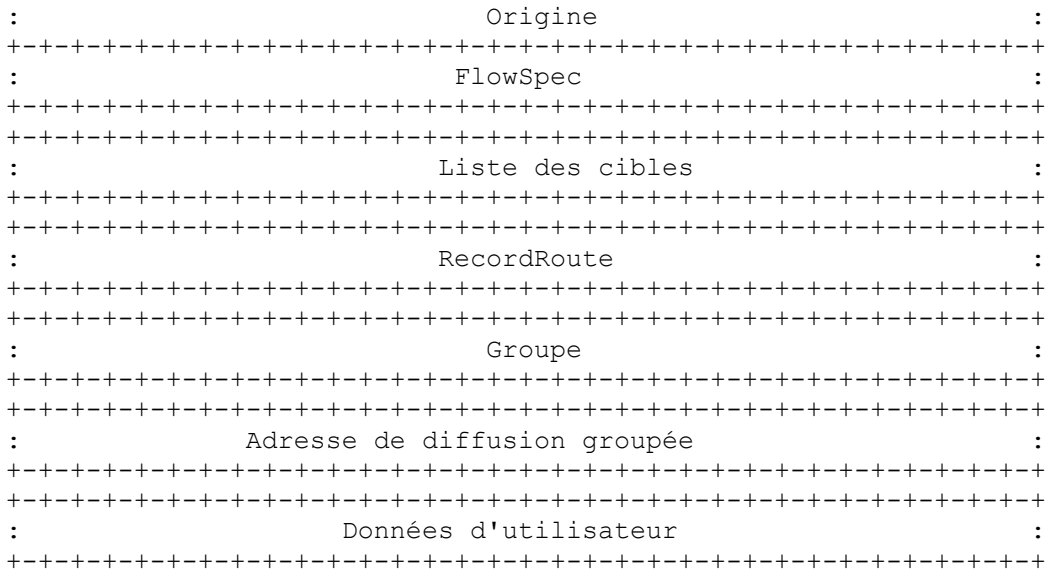


Figure 24 : Message de contrôle CONNECT

- JN (bits 8 et 9) indique le niveau d'autorisation d'adhésion pour le flux (voir au paragraphe 4.4.2).
- S (bit 10) indique l'option NoRecovery (§ 4.4.1). Si le bit S est mis (à 1), l'option NoRecovery est spécifiée pour le flux.
- Référence contient un numéro alloué par l'agent ST qui envoie CONNECT à utiliser dans le ACK d'accusé de réception.
- Taille maximum de message indique la plus petite MTU le long du chemin traversé par le flux. Ce champ est mis initialement à la MTU du réseau de l'agent qui produit le CONNECT.
- Temporisateur de récupération est le nombre nominal de millisecondes que l'application accepte d'attendre pour qu'un composant de système défaillant soit détecté et que toute action corrective soit entreprise.
- Heure de création du flux est l'horodatage de 32 bits qui dépend du système généré par l'agent ST qui émet le CONNECT.
- Bonds IP est le nombre de bonds encapsulés dans IP traversés par le flux. Ce champ est mis à zéro par l'origine, et est incrémenté à chaque agent d'encapsulation IP.

10.4.5 DISCONNECT

DISCONNECT (OpCode = 5) est utilisé par une origine pour éliminer un flux établi ou une partie d'un flux, ou par un agent ST intermédiaire qui détecte une défaillance entre lui-même et son bond précédent, comme indiqué par le code de cause. Le message DISCONNECT spécifie la liste des cibles qui sont à déconnecter. Un ACK est exigé en réponse à un message DISCONNECT. Le message DISCONNECT est propagé tout le long du chemin vers les cibles spécifiées. Les cibles sont supposées terminer leur participation au flux.

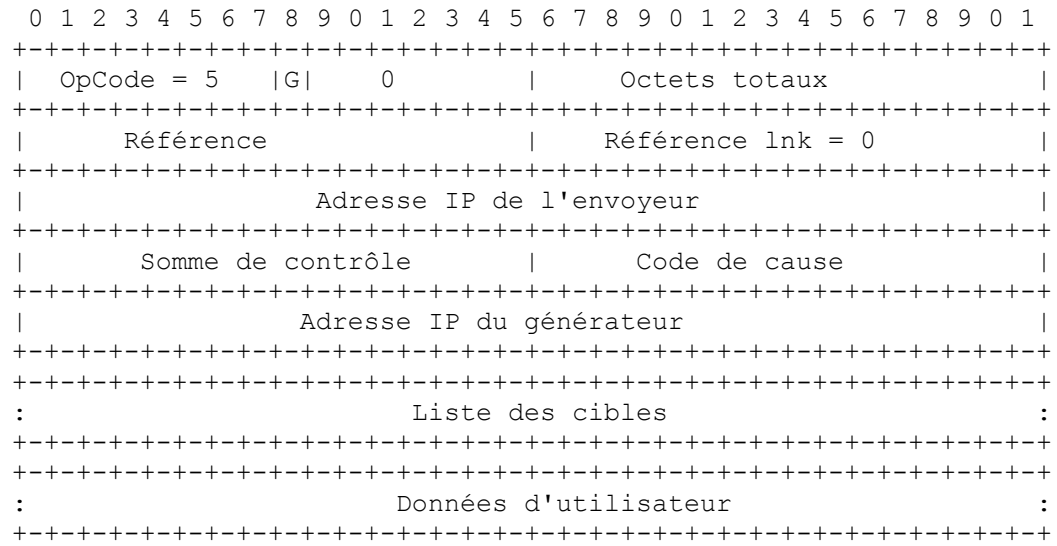


Figure 25 : Message de contrôle DISCONNECT

- G (bit 8) est utilisé pour demander un DISCONNECT à toutes les cibles du flux. La Liste des cibles devrait être omise lorsque le bit G est mis *(à 1)*. Si la Liste des cibles est présente, il est ignoré.
- Référence contient un numéro alloué par l'agent ST qui envoie le DISCONNECT à utiliser dans l'accusé de réception.

- Code de cause reflète l'événement qui est à l'origine du message.
- Adresse IP du générateur est l'adresse IP de 32 bits de l'hôte qui a le premier généré le message DISCONNECT.

10.4.6 ERROR

ERROR (OpCode = 6) est envoyé pour accuser réception d'une demande dans laquelle une erreur est détectée. Aucune action n'est entreprise sur la demande erronée. Aucun ACK n'est attendu. Le message ERROR n'est pas propagé au delà du bond précédent ou de l'agent ST du prochain bond. Un ERROR n'est jamais envoyé en réponse à un autre ERROR. Le receveur d'un ERROR est invité à réessayer sans attendre une fin de temporisation de retransmission.

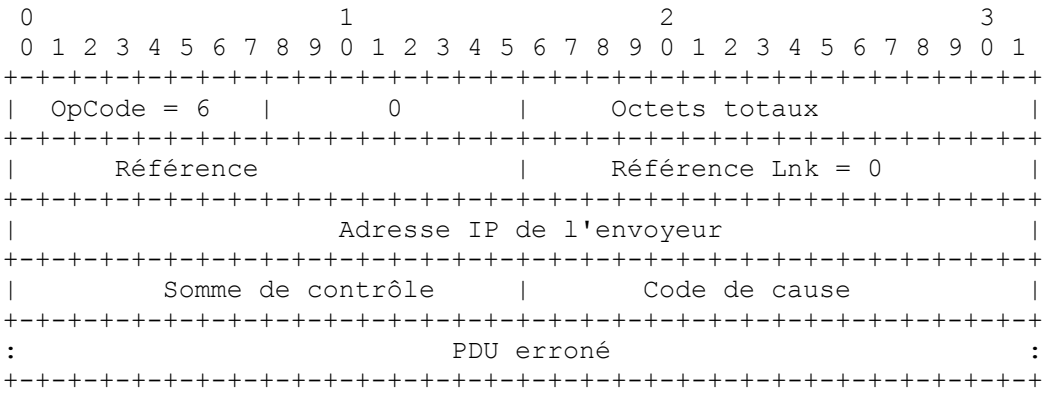


Figure 26 : Message de contrôle ERROR

- Référence est le numéro de référence de la demande erronée.
- Code de cause indique l'erreur qui a déclanché le message.
- PDU erroné est la PDU en erreur, qui commence avec l'en-tête ST. Ce paramètre est facultatif. Sa longueur est limitée par la MTU du réseau, et il peut être tronqué lorsque il est trop long.

10.4.7 HELLO

HELLO (OpCode = 7) est utilisé au titre du mécanisme de détection de défaillance ST (voir au paragraphe 6.1).

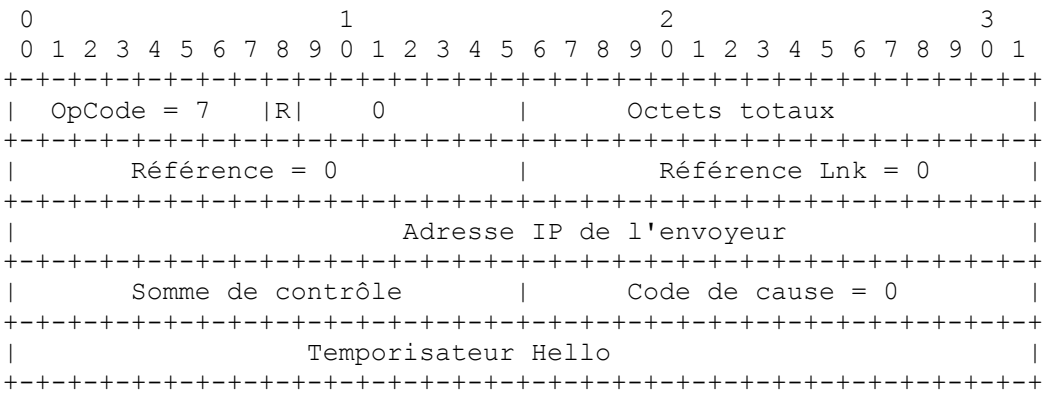


Figure 27 : Message de contrôle HELLO

- R (bit 8) est utilisé pour le bit de redémarrage.
- Temporisateur Hello représente la durée en millisecondes depuis que l'agent a été redémarré, modulo la précision du champ. Il est utilisé pour détecter les messages HELLO dupliqués ou retardés.

10.4.8 JOIN

JOIN (OpCode = 8) est utilisé au titre du mécanisme d'adhésion au flux ST (voir au paragraphe 4.6.3).

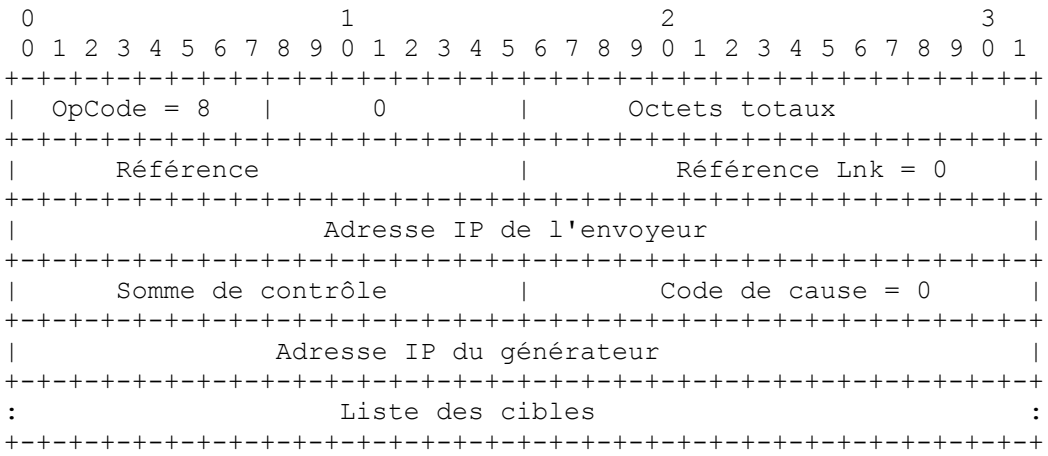


Figure 28 : Message de contrôle JOIN

- Référence contient un numéro alloué par l'agent ST qui envoie JOIN pour être utilisé dans le ACK d'accusé de réception.
- Adresse IP du générateur est l'adresse IP de 32 bits de l'hôte qui a généré le message JOIN.
- Liste des cibles sont les informations associées à la cible à ajouter au flux.

10.4.9 JOIN-REJECT

JOIN-REJECT (OpCode = 9) est utilisé au titre du mécanisme d'adhésion au flux ST (voir au paragraphe 4.6.3).

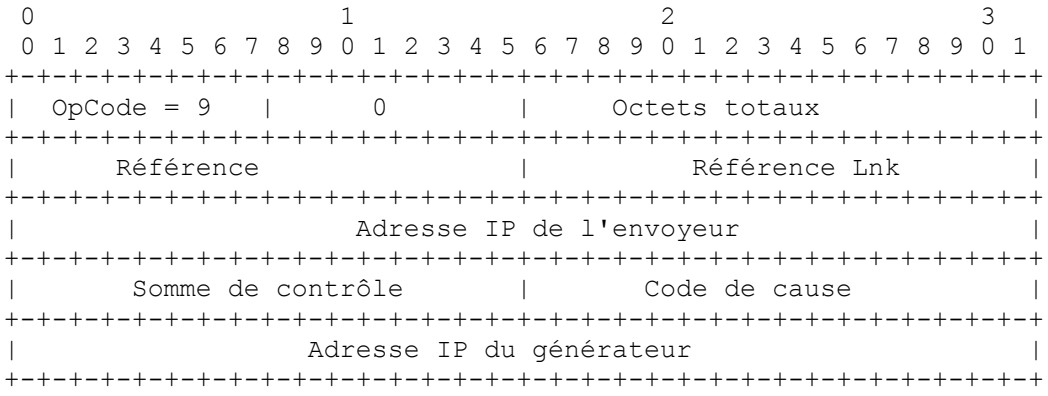
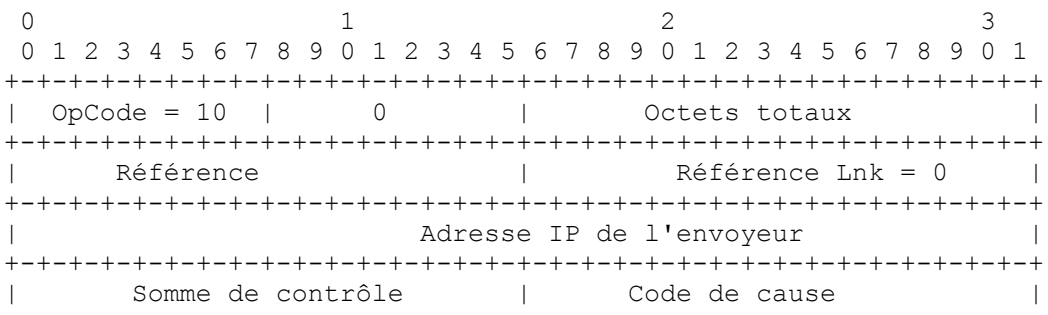


Figure 29 : Message de contrôle JOIN-REJECT

- Référence contient un numéro alloué par l'agent ST qui envoie le REFUSE à utiliser dans le ACK d'accusé de réception.
- Référence Lnk est le numéro de référence tiré du message JOIN correspondant.
- Code de cause reflète la raison pour laquelle la demande JOIN a été rejetée.
- Adresse IP du générateur est l'adresse IP de 32 bits de l'hôte qui a le premier généré le message JOIN-REJECT.

10.4.10 NOTIFY

NOTIFY (OpCode = 10) est émis par un agent ST pour informer les autres agents ST des événements qui peuvent être significatifs. NOTIFY peut être propagé au delà du bond précédent ou de l'agent ST du prochain bond selon le code de cause (voir au paragraphe 10.5.3) ; NOTIFY doit recevoir un ACK d'accusé de réception.



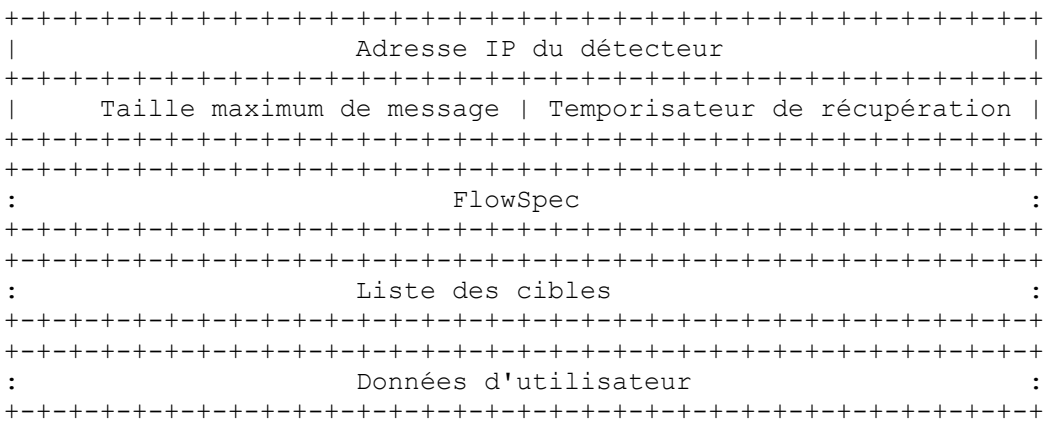


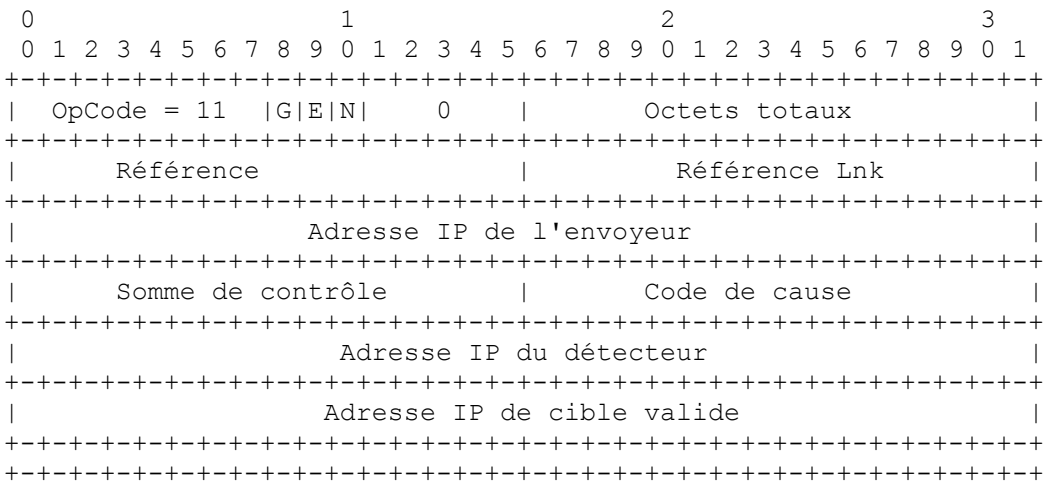
Figure 30 : Message de contrôle NOTIFY

- Référence contient un numéro alloué par l'agent ST qui envoie le NOTIFY pour l'utiliser dans le ACK d'accusé de réception.
- Code de cause identifie la raison de la notification.
- Adresse IP du détecteur est l'adresse IP de 32 bits de l'agent ST qui a détecté l'événement.
- Taille maximum de message est établi lorsque la MTU des cibles de la liste a changée (par exemple, du fait d'une récupération), ou lorsque la notification est générée après un JOIN réussi. Autrement il est réglé à zéro (0).
- Temporisateur de récupération est mis lorsque la notification est générée après un JOIN réussi. Autrement, il est mis à zéro (0).
- FlowSpec est présent lorsque la notification est générée après un JOIN réussi.
- Liste des cibles est présent lorsque la notification se rapporte à une ou plusieurs cibles, ou lorsque Taille maximum de message est établi.
- Données d'utilisateur est présent si la notification est générée après un JOIN réussi et si le paramètre Données d'utilisateur était mis dans le message ACCEPT.

10.4.11 REFUSE

REFUSE (OpCode = 11) est émis par une cible qui ne souhaite pas accepter un message CONNECT ou qui souhaite se retirer d'un flux établi. Il peut aussi être émis par un agent ST intermédiaire en réponse à un CONNECT ou CHANGE pour mettre fin à une boucle d'acheminement, ou lorsque ne peut être trouvé un prochain bond satisfaisant vers une cible. Il peut aussi être une commande distincte lorsque un flux existant a été préempté par un flux de préséance supérieure ou lorsque un agent ST détecte la défaillance d'un bond précédent, d'un prochain bond, ou du réseau entre eux. Dans tous les cas, la liste des cibles spécifie les cibles qui sont affectées par la condition. Chaque REFUSE doit recevoir un ACK en accusé de réception.

Le REFUSE est relayé vers l'origine par les agents ST (ou l'agent ST intermédiaire qui a créé le CONNECT ou CHANGE) tout le long du chemin tracé par le CONNECT. L'agent ST qui reçoit le REFUSE va le traiter différemment selon la condition qui en est la cause, telle que spécifiée dans le champ Code de cause. Aucun effort particulier n'est fait pour combiner plusieurs messages REFUSE car il est considéré comme très peu vraisemblable que des REFUSE séparés surviennent dans les deux sens d'un agent ST au même moment et soient facilement combinés, par exemple, en ayant des codes de cause et des paramètres identiques.



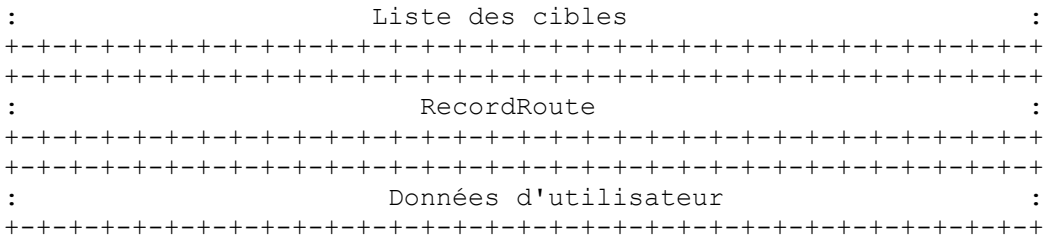


Figure 31 : Message de contrôle REFUSE

- G (bit 8) est utilisé pour indiquer que toutes les cibles vers l'aval à partir de l'envoyeur refusent. On suppose qu'il sera établi le plus couramment lors d'une défaillance réseau. Le paramètre Liste des cibles est ignoré ou absent lorsque ce bit est mis *(à 1)* et il doit être inclus s'il n'est pas établi *(mis à 0)*.
- E (bit 9) est mis par un agent ST pour indiquer que la demande a échoué et que les attributs du flux antérieurs au changement, y compris les ressources, et le flux lui-même, existent toujours.
- N (bit 10) est utilisé pour indiquer qu'aucune autre tentative de récupération du flux ne devrait être faite. Ce bit doit être établi lorsque la récupération du flux ne devrait pas être tentée, même dans le cas où l'application cible s'est clôturée normalement (ApplDisconnect).
- Référence contient un numéro alloué par l'agent ST qui envoie le REFUSE pour utilisation dans le ACK d'accusé de réception.
- Référence Lnk est le numéro de référence du CONNECT ou CHANGE correspondant, si il est le résultat d'un tel message, ou zéro lorsque le REFUSE a été généré comme une commande distincte.
- Adresse IP du détecteur est l'adresse IP de 32 bits de l'hôte qui le premier a généré le message REFUSE.
- Adresse IP de cible valide est l'adresse IP de 32 bits d'un hôte correctement connecté au titre du flux. Ce paramètre n'est utilisé que lors de la récupération suite à une convergence de flux, autrement, il est mis à zéro (0).

10.4.12 STATUS

STATUS (OpCode = 12) est utilisé pour enquêter sur l'existence d'un flux particulier identifié par le SID. L'utilisation de STATUS est destinée à la collecte d'informations à partir d'un agent ST voisin, y compris des informations générales et spécifiques d'un flux, et l'estimation de la durée de l'aller-retour. L'utilisation de ce type de message est décrite au paragraphe 8.4.

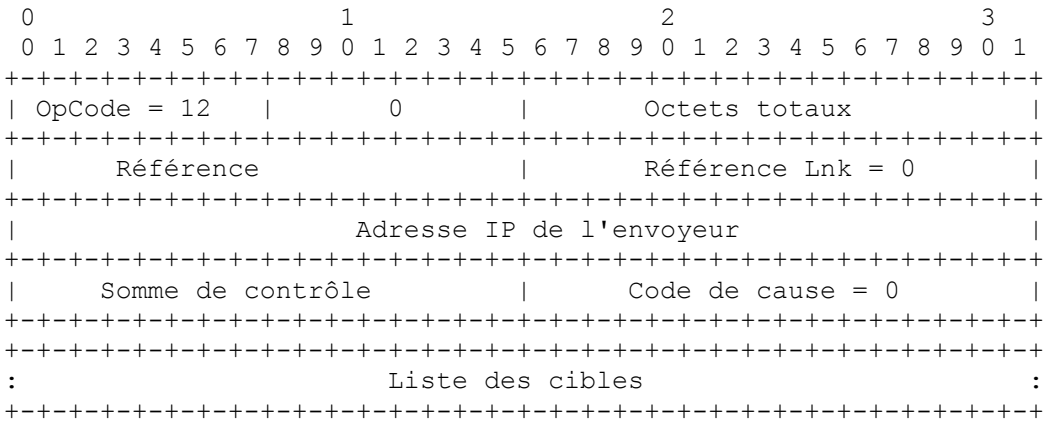


Figure 32 : Message de contrôle STATUS

- Référence contient un numéro alloué par l'agent ST qui envoie le STATUS pour utilisation dans le STATUS-RESPONSE de la réplique.
- Liste des cibles est un paramètre facultatif qui lorsque il est présent indique que seules les informations se rapportant aux cibles spécifiées devraient être relayées dans la STATUS-RESPONSE.

10.4.13 STATUS-RESPONSE

STATUS-RESPONSE (OpCode = 13) est la réplique à un message STATUS. Si le flux spécifié dans le message STATUS est inconnu, la STATUS-RESPONSE va contenir le SID spécifié mais pas d'autre paramètre. Autrement, elle contiendra le SID actuel, la FlowSpec, la Liste des cibles, et éventuellement les Groupes du flux. Si toute la liste des cibles ne peut pas tenir dans un seul message, seule les cibles qui peuvent être incluses dans un message seront incluses. Comme mentionné au

paragraphe 10.4.12, il est possible de demander des informations sur une cible spécifique.

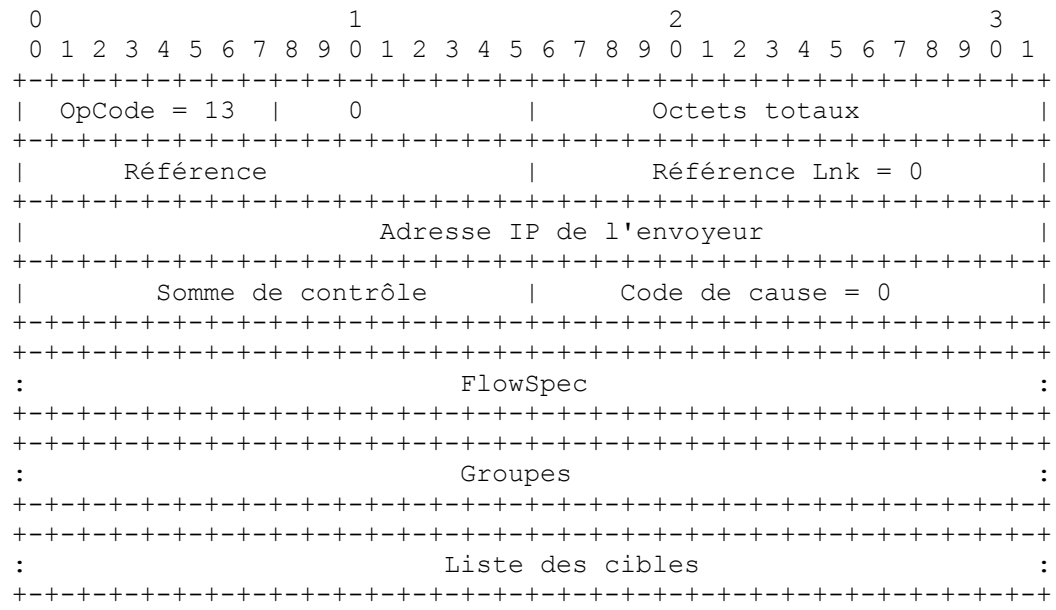


Figure 33 : Message de contrôle STATUS-RESPONSE

o Référence contient un numéro alloué par l'agent ST qui envoie le STATUS.

10.5 Constantes de protocole suggérées

Le protocole ST utilise plusieurs champs qui doivent avoir des valeurs spécifiques pour que le protocole fonctionne, et aussi plusieurs valeurs qu'une mise en œuvre doit choisir. Ce paragraphe spécifie les valeurs requises et suggère des valeurs initiales pour les autres. Il est recommandé que ces dernières soient mises en œuvre comme variables de façon à ce qu'elles soient facilement changées lorsque l'expérience indiquera de meilleures valeurs. Ensuite, elles devraient être gérées via les facilités normales de la gestion de réseau.

ST utilise le numéro de version IP 5.

Lorsque il est encapsulé dans IP, ST utilise le numéro de protocole IP 5.

10.5.1 Messages SCMP

1) ACCEPT
2) ACK
3) CHANGE
4) CONNECT
5) DISCONNECT
6) ERROR
7) HELLO
8) JOIN
9) JOIN-REJECT
10) NOTIFY
11) REFUSE
12) STATUS
13) STATUS-RESPONSE

10.5.2 Paramètres SCMP

1) FlowSpec
2) Groupe
3) Adresse de diffusion groupée
4) Origine

5) RecordRoute
6) Liste des cibles
7) Données d'utilisateur

10.5.3 Code de cause

Diverses erreurs peuvent survenir durant le traitement du protocole. Tous les codes d'erreur ST sont tirés d'un seul espace numérique. Les valeurs actuellement définies et leur signification sont présentées dans la liste ci-dessous. Noter que de nouveaux codes d'erreur peuvent être définis de temps en temps. Toutes les mises en œuvre sont supposées traiter les nouveaux codes avec douceur. Si un Code de cause inconnu est rencontré, il devrait être supposé fatal. Le code de cause est un champ de 8 bits. Les valeurs suivantes sont définies :

1	NoError	Aucune erreur n'est survenue.
2	ErrorUnknown	Une erreur non contenue dans cette liste a été détectée.
3	AccessDenied	Accès refusé.
4	AckUnexpected	Un ACK inattendu a été reçu.
5	ApplAbort	L'application a interrompu le flux de façon anormale.
6	ApplDisconnect	L'application a clos le flux normalement.
7	ApplRefused	Les applications ont refusé la connexion ou changement demandé.
8	AuthentFailed	La fonction d'authentification a échoué.
9	BadMcastAddress IP	L'adresse de diffusion groupée est inacceptable dans CONNECT
10	CantGetResrc	Incapable d'acquérir des ressources (supplémentaires).
11	CantRelResrc	Incapable de libérer les ressources en excès.
12	CantRecover	Incapable de récupérer le flux défaillant.
13	CksumBadCtl	La PDU de contrôle a une mauvaise somme de contrôle de message.
14	CksumBadST	La PDU a une mauvaise somme de contrôle d'en-tête ST.
15	DuplicateIgn	La PDU de contrôle est une réplique dont l'accusé de réception est en cours.
16	DuplicateTarget Control	La PDU contient une cible dupliquée, ou une tentative d'ajout d'une cible existante.
17	FlowSpecMismatch	La FlowSpec dans la demande ne correspond pas à la FlowSpec existante.
18	FlowSpecError	Une erreur est survenue durant le traitement de la FlowSpec
19	FlowVerUnknown	La PDU de contrôle a un numéro de version de FlowSpec qui n'est pas pris en charge.
20	GroupUnknown	La PDU de contrôle contient un Nom de groupe inconnu.
21	InconsistGroup	Une incohérence a été détectée avec les flux qui forment un groupe.
22	IntfcFailure	Une défaillance d'interface réseau a été détectée.
23	InvalidSender	La PDU de contrôle a un champ Adresse IP de l'envoyeur invalide.
24	InvalidTotByt	La PDU de contrôle a un champ Octets totaux invalide.
25	JoinAuthFailure	Échec de l'adhésion due au niveau d'autorisation d'adhésion du flux.
26	LnkRefUnknown	La PDU de contrôle contient une Référence Lnk inconnue.
27	NetworkFailure	Une défaillance réseau a été détectée.
28	NoRouteToAgent	Impossible de trouver un chemin vers un agent ST.
29	NoRouteToHost	Impossible de trouver un chemin vers un hôte.
30	NoRouteToNet	Impossible de trouver un chemin vers un réseau.
31	OpCodeUnknown	La PDU de contrôle a un champ OpCode invalide.
32	PCodeUnknown	La PDU de contrôle a un paramètre qui a un PCode invalide.
33	ParmValueBad	La PDU de contrôle contient une valeur de paramètre invalide.
34	PathConvergence	Deux branches du flux se joignent durant l'établissement de CONNECT.
35	ProtocolUnknown	La PDU de contrôle contient un identifiant inconnu de prochain protocole de couche supérieure.
36	RecordRouteSize	RecordRoute est trop long pour permettre au message de tenir dans la MTU du réseau.
37	RefUnknown	La PDU de contrôle contient une référence inconnue.
38	ResponseTimeout	Le message de contrôle a reçu un accusé de réception mais il n'y a pas été répondu par un message de contrôle approprié.
39	RestartLocal	L'agent ST local a récemment redémarré.
40	RestartRemote	L'agent ST distant a récemment redémarré.
41	RetransTimeout	Après plusieurs retransmissions aucun accusé de réception n'a été reçu.
42	RouteBack	Le chemin pour le prochain bond à travers la même interface que le bond précédent n'est pas le bond précédent.
43	RouteInconsist	Une incohérence d'acheminement a été détectée.
44	RouteLoop	Une boucle d'acheminement a été détectée.
45	SAPUnknown	La PDU de contrôle contient un SAP (accès) de prochaine couche supérieure inconnu.
46	SIDUnknown	La PDU de contrôle contient un SID inconnu.

47	STAgentFailure	Une défaillance d'agent ST a été détectée.
48	STVer3Bad	Une PDU reçus n'est pas ST version 3.
49	StreamExists	Un flux avec ce SID existe déjà.
50	StreamPreempted	Le flux a été préempté par un flux de préséance supérieure.
51	TargetExists	Un CONNECT qui spécifie une cible existante a été reçu.
52	TargetUnknown	Une cible n'est pas membre du flux spécifié.
53	TargetMissing	Un paramètre Cible était attendu et n'est pas inclus, ou est vide.
54	TruncatedCtl	La PDU de contrôle est plus courte qu'attendu.
55	TruncatedPDU	La PDU ST reçue est plus courte que ce que l'en-tête ST indique.
56	UserDataSize	Le paramètre Données d'utilisateur est trop grand pour permettre à un message de tenir dans la MTU du réseau.

10.5.4 Temporisations et autres constantes

SCMP utilise la retransmission pour assurer la fiabilité et a donc plusieurs "temporisateurs de retransmission". Chaque "temporisateur" est modélisé par un intervalle de temps initial (ToXxx), qui peut être mis à jour de façon dynamique par la mesure du trafic de contrôle, et le nombre de fois (NXxx) à retransmettre un message avant de déclarer une défaillance. Tous les intervalles de temps sont en millisecondes. Noter que les variables ne sont décrites qu'aux fins de référence, les différentes mises en œuvre pouvant ne pas inclure des variables identiques.

Valeur	Nom du temporisateur	Signification
00	ToAccept	Temporisation initiale bond par bond de l'accusé de réception de ACCEPT
3	NAccept	Réessais de ACCEPT avant déclaration de défaillance
500	ToChange	Temporisation initiale bond par bond de l'accusé de réception de CHANGE
3	NChange	Réessais de CHANGE avant déclaration de défaillance
5000	ToChangeResp	Temporisation de CHANGE de bout en bout pour la réception de ACCEPT ou REFUSE
500	ToConnect	Temporisation initiale bond par bond de l'accusé de réception de CONNECT
5	NConnect	Réessais de CONNECT avant déclaration de défaillance
5000	ToConnectResp	Temporisation de CONNECT de bout en bout pour la réception d'un ACCEPT ou REFUSE par l'origine de la part des cibles
500	ToDisconnect	Temporisation initiale bond par bond de l'accusé de réception de DISCONNECT
3	NDisconnect	Réessais de DISCONNECT avant déclaration de défaillance
500	ToJoin	Temporisation initiale bond par bond de l'accusé de réception de JOIN
3	NJoin	Réessais de JOIN avant déclaration de défaillance
500	ToJoinReject	Temporisation initiale bond par bond de l'accusé de réception de JOIN-REJECT
3	NJoinReject	Réessais de JOIN-REJECT avant déclaration de défaillance
5000	ToJoinResp	Temporisation pour la réception de CONNECT ou JOIN-REJECT de la par de l'origine ou d'un bond intermédiaire
500	ToNotify	Temporisation initiale bond par bond de l'accusé de réception de NOTIFY
3	NNotify	Réessais de NOTIFY avant déclaration de défaillance
500	ToRefuse	Temporisation initiale bond par bond de l'accusé de réception de REFUSE
3	NRefuse	Réessais de REFUSE avant déclaration de défaillance
500	ToRetryRoute	Temporisation pour la réception de ACCEPT ou REFUSE de la part des cibles durant une récupération de défaillance
5	NRetryRoute	Réessais de CONNECT avant déclaration de défaillance
1000	ToStatusResp	Temporisation pour la réception de STATUS-RESPONSE
3	NStatus	Réessais de STATUS avant déclaration de défaillance
10000	HelloTimerHoldDown	Intervalle auquel le bit Redémarré doit être réglé après un redémarrage ST
5	HelloLossFactor	Nombre de messages HELLO consécutifs manqués avant de déclarer la liaison défaillante
2000	DefaultRecoveryTimeout	Intervalle entre les Hello successifs de/vers les voisins actifs

10.6 Notations des données

La convention dans la documentation du protocole Internet est d'exprimer les nombres en décimal et de représenter les données avec l'octet de plus fort poids à gauche et l'octet de moindre poids à droite.

L'ordre de transmission de l'en-tête et des données décrites dans le présent document est résolu au niveau de l'octet. Chaque fois qu'un diagramme montre un groupe d'octets, l'ordre de transmission de ces octets est l'ordre normal dans lequel ils sont lus en français. Par exemple, dans le diagramme qui suit, les octets sont transmis dans l'ordre de leur numérotation.

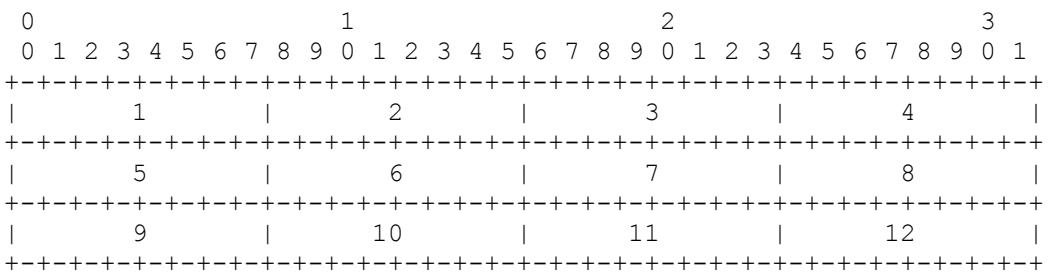


Figure 34 : Ordre de transmission des octets

Chaque fois qu'un octet représente une quantité numérique, le bit le plus à gauche est le bit de poids fort ou bit de plus fort poids. C'est à dire que le bit marqué 0 est le bit de plus fort poids. Par exemple, le diagramme suivant représente la valeur 170 (en décimal).

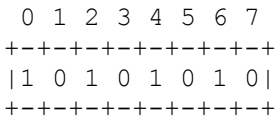


Figure 35 : Signification des bits

De même, chaque fois qu'un champ multi octet représente une quantité numérique, le bit le plus à gauche de tout le champ est le bit de plus fort poids. Lorsque une quantité multi octet est transmise, l'octet de poids fort est transmis en premier.

Les champs dont la longueur est fixe et complètement illustrée sont représentés avec une barre verticale (|) à la fin ; les champs fixes dont le contenu est abrégé sont représentés avec un point d'exclamation (!) ; les champs variables sont représentés avec des points-virgules (:). Les paramètres facultatifs sont séparés des messages de contrôle par une ligne blanche. L'ordre des paramètres n'est pas significatif.

11. Références

[RFC1071] R. Braden, D. Borman et C. Partridge, "Calcul de la somme de contrôle Internet", septembre 1988.

[RFC1112] S. Deering, "Extensions d'hôte pour diffusion groupée sur IP", STD 5, août 1989.

[WoHD95] L. Wolf, R. G. Herrtwich, L. Delgrossi, "Filtering Multimedia Data in Reservation-based Networks", Kommunikation in Verteilten Systemen 1995 (KiVS), Chemnitz-Zwickau, Germany, février 1995.

[RFC1122] R. Braden, "Exigences pour les hôtes Internet – couches de communication", STD 3, octobre 1989.

[Jaco88] Jacobson, V., "Congestion Avoidance and Control", ACM SIGCOMM-88, août 1988.

[KaPa87] Karn, P. and C. Partridge, "Round Trip Time Estimation", ACM SIGCOMM-87, août 1987.

[RFC1141] T. Mallory et A. Kullberg, "Mise à jour incrémentaire de la somme de contrôle Internet", janvier 1990.

[RFC1363] C. Partridge, "Proposition de spécification de flux", septembre 1992. *(Info.)*

[RFC0791] J. Postel, éd., "Protocole Internet - Spécification du protocole du programme Internet", STD 5, septembre 1981.

[RFC1700] J. Reynolds et J. Postel, "Numéros alloués", STD 2, octobre 1994. *(Historique)*

[RFC1190] C. Topolcic, éditeur, "Protocole expérimental de flux Internet, version 2 (ST-II)", octobre 1990. *(obsolète, voir la RFC 1819)*

[RFC1633] R. Braden, D. Clark et S. Shenker, "Intégration de services dans l'architecture de l'Internet : généralités", juin 1994. *(Info.)*

[VoHN93] C. Vogt, R. G. Herrtwich, R. Nagarajan, "HeiRAT: the Heidelberg Resource Administration Technique - Design Philosophy and Goals", Kommunikation In Verteilten Systemen, Munich, Informatik Aktuell, Springer-Verlag, Heidelberg, 1993.

[Cohe81] D. Cohen, "A Network Voice Protocol NVP-II", University of Southern California, Los Angeles, 1981.

[Cole81] R. Cole, "PVP - A Packet Video Protocol", University of Southern California, Los Angeles, 1981.

[DeAl92] L. Delgrossi, éd., "The BERKOM-II Multimedia Transport System, Version 1", Document de travail BERKOM, octobre 1992.

[DHHS92] L. Delgrossi, C. Halstrick, R. G. Herrtwich, H. Stuettgen, "HeiTP: a Transport Protocol for ST-II", GLOBECOM'92, Orlando (Florida), décembre 1992.

[RFC1889] H. Schulzrinne, S. Casner, R. Frederick et V. Jacobson, "RTP : protocole de transport pour applications en temps réel", janvier 1996. *(Obsolète, voir RFC3550 STD64)*

12. Considérations pour la sécurité

Les questions de sécurité ne sont pas abordées dans le présent mémoire.

13. Remerciements et adresse des auteurs

De nombreuses personnes ont contribué au travail décrit dans le présent mémoire. Nous remercions les participants au groupe de travail ST pour leurs apports, leur relecture et leurs commentaires constructifs, le centre C3I de l'Université George Mason pour avoir accueilli une réunion intermédiaire, Murali Rajagopal pour ses efforts sur les automates à état de ST2+. Des remerciements tout particuliers à Steve DeJarnett, qui a été co-président du groupe de travail jusqu'à l'été 1993.

Nous tenons aussi à remercier les auteurs de la [RFC1190]. Tous les auteurs de la [RFC1190] doivent être considérés comme les auteurs du présent document car il contient la plus grande partie de leur texte et de leurs idées.

Louis Berger
BBN Systems and Technologies
1300 North 17th Street, Suite 1200
Arlington, VA 22209
téléphone : 703-284-4651
mél : lberger@bbn.com

Luca Delgrossi
Andersen Consulting Technology Park
449, Route des Cretes
06902 Sophia Antipolis, France
téléphone : +33.92.94.80.92
mél : luca@andersen.fr

Dat Duong
BBN Systems and Technologies
1300 North 17th Street, Suite 1200
Arlington, VA 22209
téléphone : 703-284-4760
mél : dat@bbn.com

Steve Jackowski
Syzygy Communications Incorporated
269 Mt. Hermon Road
Scotts Valley, CA 95066
USA
téléphone : 408-439-6834
mél : stevej@syzygycomm.com

Sibylle Schaller
IBM ENC
Broadband Multimedia Communications
Vangerowstr. 18
D69020 Heidelberg, Germany
téléphone : +49-6221-5944553
mél : schaller@heidelbg.ibm.com