Groupe de travail Réseau
Request for Comments : 3290
Catégorie : Information

Traduction Claude Brière de L'Isle

Y. Bernet, Microsoft
S. Blake, Ericsson
D. Grossman, Motorola
A. Smith, Harbour Networks
mai 2002

Modèle informel de gestion pour les routeurs Diffserv

Statut du présent mémoire
Le présent mémoire apporte des informations pour la communauté de l'Internet. Il ne spécifie aucune sorte de norme de l'Internet. La distribution du présent mémoire n'est soumise à aucune restriction.

Résumé
Le présent document propose un modèle informel de gestion des routeurs de services différenciés (Diffserv, *Differentiated Services*) à utiliser dans leur gestion et leur configuration. Ce modèle définit des éléments fonctionnels du chemin des données (par exemple, classeurs, mesureurs, actions, marquage, abandon absolu, comptage, multiplexage) des abandonneurs algorithmiques, des files d'attente et des programmateurs. Il décrit les paramètres de configuration possibles pour ces éléments et la façon dont ils peuvent être interconnectés pour réaliser la gamme des fonctionnalités de conditionnements de trafic et de comportement par bond (PHB, *per-hop behavior*) décrites dans l'architecture Diffserv.

Table des Matières

1. Introduction

Les services différenciés (Diffserv) [RFC2475] sont un ensemble de technologies qui permet aux fournisseurs de services réseau d'offrir des services avec différentes sortes d'objectifs de qualité de service (QS) aux différents consommateurs et à leurs flux de trafic. Le présent document utilise la terminologie définie dans les [RFC2475] et [RFC3260] (certaines de ces définitions sont incluses dans la Section 2 pour en faciliter la consultation).

La première condition des réseaux Diffserv est que les routeurs dans le cœur du réseau traitent les paquets dans les différents flux de trafic en les transmettant à l'aide de comportements par bond (PHB, *per-hop behavior*) différents. Le PHB à appliquer est indiqué par un codet Diffserv (DSCP, *Diffserv codepoint*) dans l'en-tête IP de chaque paquet [RFC2474]. Les marquages DSCP sont appliqués soit par un nœud amont de confiance, par exemple, un consommateur, soit pas les routeurs bordure à l'entrée du réseau Diffserv.

L'avantage d'un tel schéma est que de nombreux flux de trafic peuvent être agrégés en un des quelques agrégats de comportement (BA, *behavior aggregate*) dont chacun est transmis en utilisant le même PHB au routeur, simplifiant par là le traitement et la mémorisation associés. De plus, il n'y a pas d'autre signalisation que ce qui est porté dans le DSCP de chaque paquet, et aucun autre traitement que celui qui est requis dans le cœur du réseau Diffserv car la QS est invoquée paquet par paquet.

L'architecture Diffserv permet le déploiement de divers services possibles dans un réseau. Ces services sont indiqués au consommateur sur les bords du réseau Diffserv sous la forme de spécification de niveau de service (SLS, *Service Level Specification*) [RFC3260]. Bien que une discussion plus approfondie de tels services sorte du domaine d'application du présent document (voir la [RFC3086]) la capacité à fournir ces services dépend de la disponibilité d'outils de gestion et de configuration qui puissent être utilisés pour provisionner et surveiller un ensemble de routeurs Diffserv d'une manière coordonnée. Pour faciliter le développement de tels outils de configuration et de gestion, il sera utile de définir un modèle conceptuel de routeur Diffserv qui fasse abstraction des détails de mise en œuvre des routeurs Diffserv particuliers pour s'attacher aux paramètres intéressant la configuration et la gestion. L'objet du présent document est la définition d'un tel modèle.

La fonction de transmission de base d'un routeur Diffserv est définie dans d'autres spécifications, par exemple, [RFC2474], [RFC2475], [RFC2597], [RFC3246].

Le présent document n'est destiné en aucune façon à contraindre ou imposer une autre manière de mettre en œuvre les routeurs Diffserv. On s'attend à ce que les mises en œuvre de routeur fassent preuve d'une grande variabilité. Dans la mesure où les mises en œuvre sont capables de se modéliser en utilisant les abstractions décrites dans le présent document, les outils de configuration et de gestion seront plus directement capables de configurer et gérer les réseaux qui incorporent des routeurs Diffserv d'origines compatibles.

Le présent modèle est destiné à être abstrait et capable de représenter les paramètres de configuration importants pour la fonction Diffserv pour diverses mises en œuvre spécifiques de routeurs. Il n'est pas destiné à être un guide de mise en œuvre des systèmes ni une description formelle de modélisation. Le présent modèle explique les raisons de la conception d'une MIB SNMP [RFC3289] et des autres interfaces de configuration (par exemple, des autres protocoles de gestion de politique) et éventuellement, de plus de modèles formels détaillés (par exemple, [RFC3644]) : ceux-ci devraient être cohérents avec le présent modèle.

- La Section 3 commence par décrire les blocs de base de haut niveau d'un routeur Diffserv. Elle explique les concepts utilisés dans le modèle, incluant le modèle de gestion hiérarchique pour ces blocs qui utilisent des éléments fonctionnels du chemin des données de bas niveau tels que des classeurs, des actions, des files d'attente.
- La Section 4 décrit les éléments de classeurs.
- La Section 5 expose les éléments de mesureur.
- La Section 6 discute des éléments d'action.
- La Section 7 expose les éléments de base de la mise en file d'attente par les abandonneurs algorithmiques, les files d'attente, et les programmateurs, et leurs comportements fonctionnels (par exemple, le formatage du trafic).
- La Section 8 montre comment les éléments de niveau supérieur peuvent être combinés pour construire des modules appelés des blocs de conditionnement de trafic (TCB, *Traffic Conditioning Block*) qui sont utiles pour les besoins de la gestion.
- La Section 9 expose les problèmes de sécurité.
- L'Appendice A contient un bref exposé sur le baquet de jetons et les algorithmes de baquet à fuites utilisés dans le présent modèle et certains des effets pratiques de l'utilisation de baquets de jetons dans l'architecture Diffserv.

2. Glossaire

Le présent document utilise une terminologie qui est définie dans la [RFC2475]. Il y a encore des travaux en cours sur cette terminologie à l'IETF et certaines des définitions fournies sont tirées de ces travaux. Certains des termes de ces autres références sont définis à nouveau ici afin de mieux les préciser, ainsi que quelques nouveaux termes spécifiques du présent document.

Abandonneur absolu : élément fonctionnel du chemin des données qui élimine simplement tous les paquets qui arrivent à son entrée.

Abandonneur algorithmique : élément fonctionnel du chemin des données qui élimine de façon sélective les paquets qui arrivent à son entrée, sur la base d'un algorithme d'élimination. Il a une entrée et une sortie de données.

Classeur : élément fonctionnel du chemin des données qui consiste en filtres qui choisissent les paquets conformes et non conformes. Sur la base de ce choix, les paquets sont transmis sur le chemin de données approprié au sein du routeur. Un classeur partage donc un seul flux de trafic entrant en plusieurs flux sortants.

Compteur : élément fonctionnel du chemin des données qui à met jour un compteur de paquets et aussi un compteur d'octets pour chaque paquet qui passe à travers lui.

Chemin de données *(datapath)* : chemin conceptuel pris par les paquets à travers un routeur Diffserv avec des caractéristiques particulières. Les décisions sur le chemin à prendre par un paquet sont prises par les éléments fonctionnels du chemin des données comme les classeurs et les mesureurs.

Filtre : ensemble de conditions de caractères génériques, de préfixes, de gabarit, de gamme et/ou de conditions de correspondance exacte sur le contenu des en-têtes d'un paquet ou d'autres données, et/ou d'attributs implicite ou déduits associés au paquet. Un filtre est dit correspondre seulement si chaque condition est satisfaite.

Élément fonctionnel du chemin des données : bloc de construction de base du routeur conceptuel. Les éléments typiques sont les classeurs, les mesureurs, les actions, les abandonneurs algorithmiques, les files d'attente et les programmateurs.

Multiplexeur (Mux) : un multiplexeur est un élément fonctionnel du chemin des données qui fusionne plusieurs flux de trafic (chemins de données) en un seul flux de trafic (chemin de données).

Ne conservant pas le travail : propriété d'un algorithme de programmation telle qu'il ne sert pas les paquets avant une heure de début programmée, même si cela signifie de laisser des paquets en file d'attente alors que la sortie (par exemple, une liaison réseau ou une connexion au prochain élément) est inactive.

Régulation *(policing)* : processus de comparaison de l'arrivée des paquets de données à un profil temporel et de leur transmission, retard ou abandon de façon à ce que le flux de sortie soit conforme au profil.

Bloc de mise en file d'attente : combinaison d'éléments fonctionnels du chemin des données qui module la transmission des paquets appartenant à un flux de trafic et détermine leur ordre, les mémorisant éventuellement temporairement ou les éliminant.

Algorithme de programmation : algorithme qui détermine quelle file d'attente d'un ensemble de files d'attente servir ensuite. Ce peut être fondé sur la priorité relative des files d'attente, sur une politique de partage équitable de la bande passante, ou sur quelque autre politique. Un tel algorithme peut être à conservation du travail ou sans conservation du travail.

Spécification de niveau de service (SLS) : ensemble de paramètres et de leurs valeurs qui définit le traitement offert à un flux de trafic par un domaine Diffserv.

Formatage : processus de retard des paquets au sein d'un flux de trafic pour le faire se conformer à un profil temporel défini. Le formatage peut être mis en œuvre en utilisant une file d'attente desservie par un algorithme de programmation sans conservation du travail.

Bloc de conditionnement du trafic (TCB) : entité logique du chemin des données qui consiste en un certain nombre d'éléments fonctionnels du chemin des données interconnectés d'une façon telle qu'elles effectuent un ensemble spécifique de fonctions de conditionnement de trafic sur un flux de trafic entrant. Un TCB peut être vu comme une entité avec une entrée et une ou plusieurs sorties et un ensemble de paramètres de contrôle.

Spécification de conditionnement de trafic (TCS) : ensemble de paramètres et de leurs valeurs qui ensemble spécifient un ensemble de règles de classement et un profil de trafic. Une TCS est une partie intégrante d'une SLS.

Conservation du travail : propriété d'un algorithme de programmation tel qu'il dessert un paquet, s'il en est un disponible, à chaque opportunité de transmission.

3. Modèle conceptuel

La présente section introduit un diagramme de routeur Diffserv et décrit les divers composants illustrés à la Figure 1. Noter qu'un routeur de cœur Diffserv va probablement n'avoir besoin que d'un sous ensemble de ces composants : le modèle présenté ici est destiné à couvrir le cas aussi bien des routeurs Diffserv de bordure que de cœur.

3.1 Composants d'un routeur Diffserv

Le modèle conceptuel inclut des définitions abstraites pour :
- o les éléments de classement du trafic,
- o les fonctions de mesure,
- o les actions de marquage, d'élimination absolue, de comptage, et de multiplexage,
- o les éléments de mise en file d'attente, incluant les capacités d'abandon et de programmation algorithmique,
- o certaines combinaisons des éléments fonctionnels du chemin des données ci-dessus en blocs de niveau supérieur appelés des blocs de conditionnement du trafic (TCB, *Traffic Conditioning Block*).

Les composants et combinaisons de composants décrits dans le présent document forment les blocs de construction qui doivent être gérables par les outils Diffserv de configuration et de gestion. Un des objectifs du présent document est de montrer comment un modèle d'appareil Diffserv peut être construit en utilisant ces blocs composants. Le présent modèle est sous la forme d'un graphe acyclique directement connecté (DAG, *directed acyclic graph*) d'éléments fonctionnels du chemin des données qui décrivent les comportements de conditionnement et de mise en file d'attente du trafic que va rencontrer tout paquet transmis au routeur Diffserv. La Figure 1 illustre les blocs fonctionnels majeurs d'un routeur Diffserv.

3.1.1 Chemin des données

Une interface d'entrée, un cœur d'acheminement, et une interface de sortie sont illustrés au centre du diagramme. Dans les mises en œuvre réelles de routeur, il peut y avoir un nombre arbitraire d'interfaces d'entrée et de sortie interconnectées par le cœur d'acheminement. L'élément cœur d'acheminement sert d'abstraction de la fonction normale d'acheminement et de commutation d'un routeur. Le cœur d'acheminement déplace les paquets entre les interfaces selon des politiques qui sortent du domaine d'application de Diffserv (noter qu'il est possible que de telles politiques pour le choix de l'interface de sortie impliquent l'utilisation de champs du paquet comme le DSCP, mais cela sort du domaine de notre modèle). Le délai réel de mise en file d'attente et le comportement de perte de paquet de la fabrique/arrière plan de commutation spécifique du routeur ne sont pas modélisés par le cœur d'acheminement ; ils devraient être modélisés en utilisant les éléments fonctionnels du chemin des données décrits plus loin. Le cœur d'acheminement de ce modèle peut être vu comme une bande passante infinie, interconnectée sans retard entre des interfaces – des propriétés comme le comportement du cœur lorsque il est en surcharge doivent être répercutés dans les éléments de mise en file d'attente qui sont modélisés autour de lui (par exemple, lorsque trop de trafic est dirigé à travers le cœur à une interface de sortie) l'excédant doit être éliminé ou mis en file d'attente quelque part : les éléments qui effectuent ces fonctions doivent être modélisés sur une des interfaces impliquées.

Les composants intéressants à l'entrée et la sortie des interfaces sont les éléments fonctionnels du chemin des données (par exemple, classeurs, éléments de mise en file d'attente) qui prennent en charge le conditionnement de trafic et les comportements par bond Diffserv [RFC2475]. Ce sont les composants fondamentaux d'un routeur Diffserv et le point de focalisation du présent modèle.

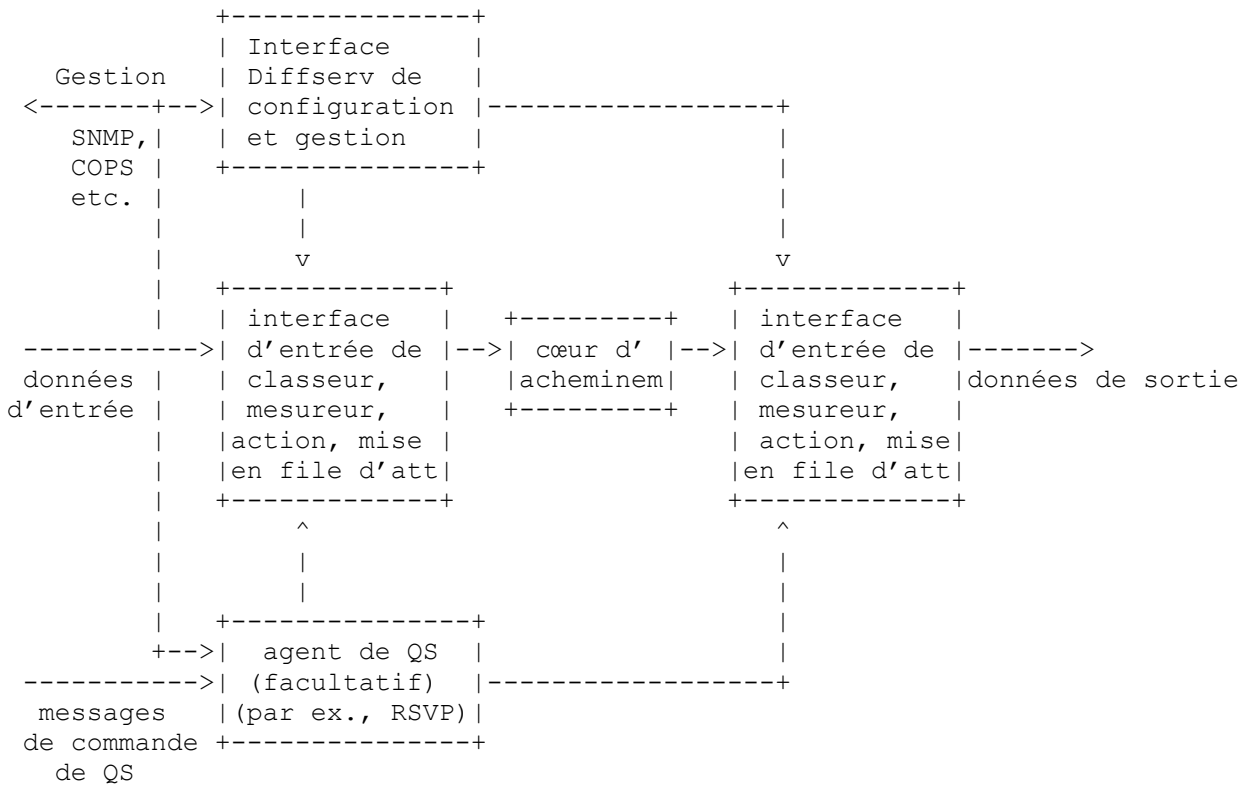

Figure 1 : Blocs fonctionnels majeurs d'un routeur Diffserv

3.1.2 Interface de configuration et de gestion

Les paramètres de fonctionnement de Diffserv sont surveillés et approvisionnés par cette interface. Les paramètres surveillés incluent des statistiques concernant le trafic porté aux divers niveaux de service de Diffserv. Ces statistiques peuvent être importantes pour des besoins de comptage et/ou pour retracer la conformité aux spécifications de conditionnement du trafic (TCS) négociées avec les consommateurs. Les paramètres provisionnés sont principalement les paramètres de TCS pour les classeurs et mesureurs et les paramètres de configuration de PHB associés pour les éléments d'actions et de mise en file d'attente. L'administrateur de réseau interagit avec l'interface de configuration et gestion Diffserv via un ou plusieurs protocoles de gestion, comme SNMP ou COPS, ou par d'autres outils de configuration de routeurs comme un terminal de série ou des consoles telnet.

Les règles et objectifs spécifiques de politique qui gouvernent le comportement Diffserv d'un routeur sont présumées être installées par des mécanismes de gestion de politique. Cependant, les routeurs Diffserv sont toujours soumis aux limites de mise en œuvre qui définissent la portée des politiques qui peuvent être mises en œuvre avec succès par le routeur. Le rapport externe de telles capacités de mise en œuvre est considéré comme sortant du domaine d'application du présent document.

3.1.3 Module facultatif d'agent de qualité de service

Les routeurs Diffserv peuvent surveiller ou participer à la signalisation par microflux ou par agrégat de flux des exigences de QS [RFC2998] (par exemple, en utilisant le protocole RSVP). La surveillance des messages RSVP peut être utilisée, par exemple, pour apprendre comment classer le trafic sans réellement participer comme un homologue du protocole RSVP. Les routeurs Diffserv peuvent rejeter ou admettre des demandes de réservation RSVP pour fournir un moyen de contrôle d'admission à des services fondés sur Diffserv, ou ils peuvent utiliser ces demandes pour déclencher des changements de provisionnement pour une agrégation de flux dans le réseau Diffserv. Une agrégation de flux dans ce contexte peut être équivalente à un BA Diffserv ou elle peut être d'une granularité plus fine, en s'appuyant sur un classeur multi champs (MF, *multi-field*) [RFC2475]. Noter que le modèle conceptuel d'un tel routeur met en œuvre le modèle de service intégré décrit dans la [RFC1633], appliquant les contrôles du plan de contrôle aux données classées et conditionnées dans le plan des données, comme décrit dans la [RFC2998].

Noter qu'un composant d'agent de QS d'un routeur Diffserv, si il est présent, ne peut être actif que dans le plan de contrôle et non dans le plan des données. Dans ce scénario, RSVP pourrait être utilisé simplement pour signaler l'état de réservation sans installer aucune réservation réelle dans le plan des données du routeur Diffserv : le plan des données pourrait encore agir sur les DSCP Diffserv et fournir des PHB pour traiter le trafic de données sans le traitement normal par microflux attendu pour

prendre en charge certains services Intserv.

3.2 Fonctions Diffserv à l'entrée et à la sortie

Le présent document se focalise sur les composants spécifiques de Diffserv du routeur. La Figure 2 donne une vision à haut niveau des interfaces d'entrée et de sortie d'un routeur. Le diagramme illustre deux interfaces de routeur Diffserv, ayant chacune un ensemble d'éléments d'entrée et de sortie. Il montre les fonctions de classement, de mesure, d'action et de mise en file d'attente qui peuvent être instanciées à l'entrée et la sortie de chaque interface.

Le simple diagramme de la Figure 2 suppose que l'ensemble de fonctions Diffserv à exécuter sur le trafic sur une certaine interface est indépendant des fonctions sur toutes les autres interfaces. Dans certaines architectures, les fonctions Diffserv peuvent être partagées entre plusieurs interfaces (par exemple, les ressources de processeur et de mise en mémoire tampon qui traitent plusieurs interfaces sur la même carte de ligne avant la transmission à travers un cœur d'acheminement). Le modèle présenté dans ce document peut être facilement étendu pour traiter de tels cas ; cependant, ce sujet n'est pas traité plus avant ici car il conduit à une excessive complexité dans l'explication des concepts.

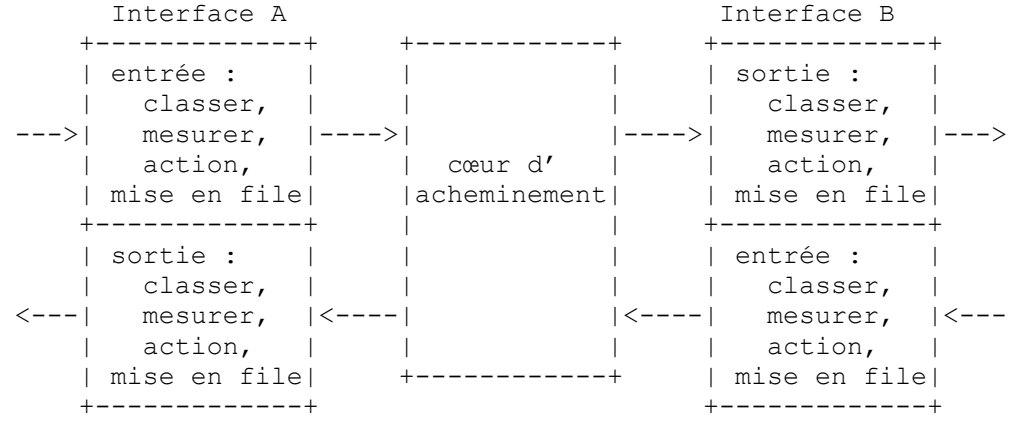

Figure 2 : Éléments de conditionnement et de mise en file d'attente du trafic

En principe, si on veut construire un réseau entièrement à partir de routeurs à deux accès (connectés par des LAN ou des supports similaires) il sera alors nécessaire que chaque routeur effectue quatre fonctions de contrôle de QS dans le chemin des données sur le trafic dans chaque direction :

- Classer chaque message selon un ensemble de règles, éventuellement juste une règle "correspondre à tout".
- Si nécessaire, déterminer si le flux de données dont fait partie le message est dans ou hors de son débit en mesurant le flux.
- Effectuer un ensemble d'actions en résultant, y compris en appliquant une politique d'abandon appropriée au classement et file d'attente en question et peut-être un marquage supplémentaire du trafic avec un codet de service différencié (DSCP) [RFC2474].
- Mettre le trafic en file d'attente pour sortie dans la file d'attente appropriée. La programmation de la sortie de cette file d'attente peut conduite à formater le trafic ou peut simplement causer sa transmission avec une assurance de débit minimum ou de latence maximum.

Si le réseau est maintenant construit à partir de routeurs à N accès, le comportement attendu du réseau devrait être identique. Donc, ce modèle doit fournir essentiellement le même jeu de fonctions à l'entrée comme à la sortie des interfaces d'un routeur. Le seul point de différence entre l'entrée et la sortie du modèle est que tout le trafic en sortie d'une interface est mis en file d'attente, tandis que le trafic en entrée de l'interface ne va probablement être mis en file d'attente que pour des besoins de formatage, s'il en est. Donc, des éléments fonctionnels équivalents du chemin des données peuvent être modélisés à la fois à l'entrée et à la sortie d'une interface.

Noter qu'il n'est pas obligatoire que chacun de ces éléments fonctionnels du chemin des données soit mis en œuvre à la fois à l'entrée et à la sortie ; également, le modèle permet que plusieurs ensembles de ces éléments soient placés en série et/ou en parallèle à l'entrée ou la sortie. L'arrangement des éléments dépend des exigences de service sur une interface particulière sur un certain routeur. En modélisant ces éléments à la fois à l'entrée et la sortie, on n'implique pas qu'ils doivent être mis en œuvre de cette façon dans un routeur spécifique. Par exemple, un routeur peut mettre en œuvre tout le formatage et la mise en file d'attente de PHB à l'interface de sortie, ou peut à la place ne le mettre en œuvre qu'à la sortie. De plus, le classement nécessaire pour transposer un paquet dans une file d'attente de sortie (s'il est présent) n'a pas besoin d'être mis en œuvre à la sortie mais peut plutôt être mis en œuvre à l'entrée, avec le paquet passé à travers le cœur d'acheminement avec des informations de contrôle dans la bande pour permettre le choix de la file d'attente de sortie.

Précisément, certaines interfaces vont être au "bord" externe et certaines vont être vers le "cœur" du domaine Diffserv. On doit s'attendre (d'après les principes généraux qui guident la motivation de Diffserv) que les interfaces de "bordure", ou au moins les routeurs qui les contiennent, mettent en œuvre plus de complexité et exigent plus de configuration que celles du cœur bien que ce ne soit évidemment pas une exigence.

3.3 Formatage et régulation

Les nœuds Diffserv peuvent appliquer le formatage, la régulation et/ou le marquage aux flux de trafic qui excèdent les limites de leur TCS afin d'empêcher qu'un flux de trafic prenne plus que sa part des ressources d'un réseau Diffserv. Dans ce modèle, le formatage, parfois considéré comme action de TC, est traité comme une fonction des éléments de mise en file d'attente – voir la Section 7. Les technique d'abandon algorithmique (par exemple, RED) sont traitées de façon similaire car elles ont souvent étroitement associées aux files d'attente. La régulation est modélisée soit comme un enchaînement d'un mesureur avec un abandonneur absolu ou un enchaînement d'un abandonneur algorithmique avec un programmateur. Ces éléments vont éliminer les paquets qui excèdent la TCS.

3.4 Vision hiérarchique du modèle

Du point de vue de la gestion de configuration au niveau appareil, il existe la hiérarchie suivante :

Au plus bas niveau considéré ici, il y a des éléments fonctionnels individuels du chemin des données, dont chacun a ses propres paramètres de configuration et compteurs de gestion et fanions.

Au niveau suivant, l'administrateur de réseau gère les groupements de ces éléments fonctionnels du chemin des données interconnectés dans un DAG. Ces éléments fonctionnels du chemin des données sont organisés en TCB auto contenus qui sont utilisés pour mettre en œuvre une politique de réseau désirée (voir la Section 8). Un ou plusieurs TCB peuvent être instanciés à l'entrée ou sortie de chaque interface ; ils peuvent être connectés en série et/ou en parallèle sur les multiples sorties d'un TCB précédant. Un TCB peut être vu comme une "boîte noire" avec une entrée et une ou plusieurs sorties (dans le chemin des données). Chaque interface peut avoir une configuration de TCB différente et chaque direction (entrée ou sortie) le peut aussi.

Au plus haut niveau considéré ici, l'administrateur réseau gère les interfaces. Chaque interface a des fonctions d'entrée et de sortie, dont chacune est exprimée par un ou plusieurs TCB. C'est ce niveau de la hiérarchie qui est illustré à la Figure 2.

D'autres niveaux peuvent être bâtis par dessus cette hiérarchie, en particulier pour aider aux tâches répétitives de configuration pour les routeurs avec beaucoup d'interfaces : de tels outils de "gabarit" pour les routeurs Diffserv sortent du domaine d'application du présent modèle mais sont à l'étude dans d'autres groupe de travail de l'IETF.

4. Classeurs

4.1 Définition

Le classement est effectué par un élément classeur. Les classeurs sont des appareils 1:N (ventilation) : ils prennent un seul flux de trafic en entrée et génèrent N flux de trafic logiquement séparés en sortie. Les classeurs sont paramétrés par des filtres et des flux de sortie. Divers types de classeurs utilisant différents filtres sont décrits dans les paragraphes qui suivent. Les paquets provenant du flux d'entrée sont triés en divers flux de sortie par des filtres qui se confrontent au contenu du paquet ou éventuellement à d'autres attributs associés au paquet. La Figure 3 illustre un classeur, et les sorties se connectent aux éléments fonctionnels successifs du chemin des données.

L'élément Classeur le plus simple possible est celui qui correspond pour tous les paquets qui sont appliqués à son entrée. Dans ce cas, l'élément Classeur est juste une non opération et peut être omis.

Noter qu'on permet un multiplexeur (voir au paragraphe 6.5) avant le classeur pour admettre des entrées provenant de plusieurs flux de trafic. Par exemple, si un flux de trafic originaire de multiples interface d'entrées s'alimente à travers un seul classeur alors le nombre d'interfaces pourrait être une des clé de classement de paquet utilisées par le classeur. Cette optimisation peut être importante pour l'adaptabilité dans le plan de gestion. Les classeurs peuvent aussi être mis à la suite (en cascade) pour effectuer des opérations plus complexes de recherche tout en conservant une telle adaptabilité.

Un autre exemple d'attribut de paquet pourrait être un entier représentant la chaîne de la communauté BGP associée avec le chemin de la meilleure correspondance du paquet. D'autre information de contexte peuvent aussi être utilisées par un classeur (par exemple, savoir qu'une certaine interface est devant un domaine Diffserv ou un domaine de type IP traditionnel

[RFC2475] pourrait être utilisé pour déterminer si un DSCP est présent ou non).

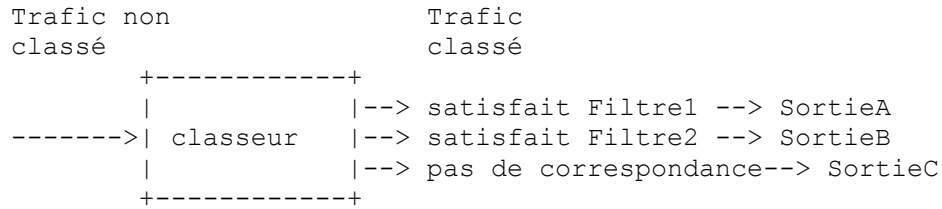

Figure 3 : Exemple de classeur

Le classeur suivant BA sépare le trafic en un des trois flux de sortie sur la base de la satisfaction des filtres :

Filtre satisfait	Flux de sortie
Filtre1	A
Filtre2	B
pas de corr.	C

Où les filtres sont définis comme étant les filtres BA suivants ([RFC2475], paragraphe 4.2.1):

Filtre	DSCP
Filtre1	101010
Filtre2	111111
Filtre3	****** (caractère générique)

4.1.1 Filtres

Un filtre consiste en un ensemble de conditions portant sur les valeurs composantes de la clé de classement d'un paquet (les valeurs d'en-tête, le contenu, et les attributs pertinents pour la classification). Dans l'exemple du classeur BA ci-dessus, la clé de classement consiste en un champ d'en-tête de paquet, le DSCP, et Filtre1 et Filtre2 spécifient tous deux des conditions de correspondance exacte sur la valeur du DSCP. Filtre3 est un filtre générique par défaut qui correspond exactement à chaque paquet, mais qui n'est sélectionné que dans le cas où aucun filtre plus spécifique ne correspond.

En général il y a un ensemble de conditions composantes possibles qui incluent les correspondances exactes de préfixe, de gamme, de gabarit et de caractère générique. Noter que les gammes peuvent être représentées (avec moins d'efficacité) comme un ensemble de préfixes et que les correspondances de préfixes sont juste un cas particulier des deux correspondances de gabarit et de gamme.

Dans le cas d'un classeur MF, la clé de classement consiste en un certain nombre de champs d'en-têtes de paquet. Le filtre peut spécifier une condition différente pour chaque composante de clé, comme illustré dans l'exemple ci-dessous pour un classeur IPv4/TCP :

Filtre	**Adresse IPv4 de source**	**Adresse IPv4 de destination**	**Accès TCP de source**	**Accès TCP de destination**
Filtre4	172.31.8.1/32	172.31.3.X/24	X	5003

Dans cet exemple, le quatrième octet de l'adresse IPv4 de destination et l'accès TCP de source sont avec un caractère générique ou "ne pas tenir compte".

Le classement MF des paquets IP fragmentés est impossible si le filtre utilise des numéros d'accès de couche transport (par exemple, des numéros d'accès TCP). La découverte de la MTU est donc un prérequis pour un bon fonctionnement d'un réseau Diffserv qui utilise de tels classeurs.

4.1.2 Chevauchement de filtres

Noter qu'il est facile de définir des ensembles de filtres qui se chevauchent dans un classeur. Par exemple :

Filtre	**Adresse IPv4 de source**	**Adresse IPv4 de destination**
Filtre5	172.31.8.X/24	X/0
Filtre6	X/0	172.30.10.1/32

Un paquet contenant {IP Dest Addr 172.31.8.1, IP Src Addr 172.30.10.1} ne peut pas être classé de façon univoque par cette

paire de filtres et donc une préséance doit être établie entre Filtre5 et Filtre6 afin de faire le départage. Cette préséance doit être établie soit (a) par un gestionnaire qui sait que le routeur peut accomplir ce rangement particulier (par exemple, au moyen d'un rapport de capacités), soit (b) par le routeur avec un mécanisme pour faire rapport à un gestionnaire de la préséance qui est utilisée. De tels mécanismes de préséance doivent être pris en charge dans toute traduction de ce modèle dans une syntaxe spécifique des protocoles de configuration et de gestion.

Comme autre exemple, on peut vouloir d'abord interdire à certaines applications d'utiliser du tout le réseau, ou bien classer certains flux de trafic individuels qui ne sont pas marqués Diffserv. Le trafic qui n'est pas classé par ces tests peut alors être inspecté à la recherche d'un DSCP. Le mot "alors" implique une séquence et cela doit être spécifié au moyen d'une préséance.

Un classeur non ambigu exige que toute clé de classement possible corresponde au moins à un filtre (éventuellement le générique par défaut) et que toute ambiguïté entre chevauchement de filtres soit résolue par une préséance. Donc, les classeurs sur toute interface doivent être "complets" et vont souvent inclure un filtre "pour tout le reste" comme élément de plus faible préséance afin que le résultat de la classification soit déterministe. Noter que cette complétude n'est requise que du premier classeur que va rencontrer le trafic entrant dans une interface - les classeurs suivants sur une interface ont seulement besoin de traiter le trafic dont on sait qu'ils vont le recevoir.

Le présent modèle de fonctionnement de classeur prend pour hypothèse que tous les filtres de même préséance sont appliqués simultanément. Bien que pratique du point de vue de la modélisation, cela peut n'être pas la façon dont le classeur est en fait mis en œuvre – cette hypothèse n'est pas destinée à imposer une façon de mettre cela en œuvre mais simplement à définir avec clarté le résultat final exigé.

4.2 Exemples

4.2.1 Classeur d'agrégat de comportement

Le plus simple classeur Diffserv est un classeur d'agrégat de comportement (BA) [RFC2475]. Un classeur BA utilise seulement le codet Diffserv (DSCP) dans un en-tête de paquet IP pour déterminer le flux de sortie logique sur lequel le paquet devrait être dirigé. On permet seulement une condition de correspondance exacte sur ce champ parce que les valeurs de DSCP allouées n'ont pas de structure, et donc qu'aucun sous ensemble de bits de DSCP n'est significatif.

Ce qui suit définit un filtre BA possible :
Filtre8 :
Type : BA
Valeur : 111000

4.2.2 Classeur multi champs

Un autre type de classeur est un classeur multi-champs (MF) [RFC2475]. Il classe les paquets sur la base d'un ou plusieurs champs dans le paquet (en incluant éventuellement le DSCP). Un type courant de classeur MF est un classeur en sextuplet qui classe sur la base de six champs des en-têtes IP et TCP ou UDP (adresse de destination, adresse de source, protocole IP, accès de source, accès de destination, et DSCP). Les classeurs MF peuvent classer sur d'autres champs comme des adresses MAC, des étiquettes de VLAN, des champs de classe de trafic de couche de liaison, ou d'autres champs de protocole de couche supérieure.

Ce qui suit définit un filtre MF possible :

Filtre9 :
Type : IPv4-6-tuple
Ipv4DestAddrValue : 0.0.0.0
Ipv4DestAddrMask : 0.0.0.0
Ipv4SrcAddrValue : 172.31.8.0
Ipv4SrcAddrMask : 255.255.255.0
Ipv4DSCP : 28
Ipv4Protocol : 6
Ipv4DestL4PortMin : 0
Ipv4DestL4PortMax : 65535
Ipv4SrcL4PortMin : 20
Ipv4SrcL4PortMax : 20

Un type de classeur similaire peut être défini pour IPv6.

4.2.3 Classeur de forme libre

Un classeur de forme libre est constitué d'un ensemble de filtres de définition arbitraire constitués de {taille de champ binaire, décalage (par rapport à la tête du paquet), gabarit} :

```
Classeur2 :
Filtre12 :   SortieA
Filtre13 :   SortieB
Défaut :     SortieC

Filtre12 :
Type :        FormeLibre
TailleBits :  3 (bits)
Décalage :    16 (octets)
Valeur :      100 (binaire)
Gabarit :     101 (binaire)

Filtre13 :
Type :        FormeLibre
TailleBits :  12 (bits)
Décalage :    16 (octets)
Valeur :      100100000000 (binaire)
Gabarit :     111111111111 (binaire)
```

Les filtres de forme libre peuvent être combinés en groupes de filtres pour former des filtres très puissants.

4.2.4 Autres classeurs possibles

Le classement peut aussi être effectué sur la base des informations à la couche de liaison des données en dessous d'IP (par exemple, VLAN ou priorité de couche de liaison des données) ou peut-être à l'entrée ou sortie IP, un identifiant d'interface physique ou logique (par exemple, le numéro de canal entrant sur une interface à canaux). Un classeur qui filtre sur la base de la priorité IEEE 802.1p et sur l'identifiant de VLAN 802.1Q pourrait être représenté comme :

```
Classeur3 :
Filtre14 ET Filtre15 :  SortieA
Défaut :                SortieB

Filtre14 :                  -- priorité 4 ou 5
Type :       Ieee8021pPriorité
Valeur :     100 (binaire)
Gabarit :    110 (binaire)

Filtre15 :                  -- VLAN 2304
Type :       Ieee8021QVlan
Valeur :     100100000000 (binaire)
Gabarit :    111111111111 (binaire)
```

De tels classeurs peuvent être sujets à d'autres standard ou être la propriété d'un fabricant de routeur mais on n'en parlera pas plus ici.

5. Mesureurs

Le mesurage est défini dans la [RFC2475]. Les fournisseurs de réseau Diffserv peuvent choisir d'offrir des services aux consommateurs sur la base d'un profil temporel (c'est-à-dire, un débit) au sein duquel le consommateur soumet du trafic pour le service. Dans cette éventualité, un mesureur peut être utilisé pour déclencher des actions de conditionnement de trafic en temps réel (par exemple, de marquage) en acheminant un paquet non conforme au moyen d'un élément d'action approprié de prochaine étape. Autrement, par un comptage du trafic conforme et/ou non conforme en utilisant un élément Compteur en aval du mesureur, cela peut aussi être utilisé pour aider à collecter des données pour des fonctions de gestion hors bande telles que des applications de facturation.

Les mesureurs sont logiquement des appareils 1:N (ventilation) (bien qu'un multiplexeur puisse être utilisé devant un mesureur). Les mesureurs sont paramétrés par un profil temporel et par des niveaux de conformité, dont chacun est associé à

une sortie de mesureur. Chaque sortie peut être connectée à un autre élément fonctionnel.

Noter que ce modèle de mesureur diffère légèrement de celui décrit dans la [RFC2475]. Dans cette description, le mesureur n'est pas un élément du chemin des données, mais est plutôt utilisé pour surveiller le flux de trafic et envoyer des signaux de contrôle aux éléments d'action pour moduler dynamiquement leur comportement sur la base de la conformité du paquet. Cette différence dans la description ne change pas la fonction d'un mesureur. La Figure 4 illustre un mesureur avec 3 niveaux de conformité.

Dans certains exemples Diffserv (par exemple, [RFC2597]), trois niveaux de conformité sont discutés en termes de couleurs, le vert représentant la conformité, le jaune représentant la conformité partielle et le rouge représentant la non conformité. Ces différents niveaux de conformité peuvent être utilisés pour déclencher différents traitements de mise en file d'attente, de marquage ou d'abandon plus tard dans le traitement. D'autres exemples de mesureurs utilisent une notion binaire de conformité ; dans le cas général, N niveaux de conformité peuvent être pris en charge. En général il n'y a pas de contrainte de type d'élément fonctionnel du chemin des données qui suive une sortie de mesureur, mais il faut faire attention à ne pas configurer par inadvertance un chemin des données qui résulterait en un réarrangement des paquets qui ne serait pas cohérent avec les exigences de la spécification pertinente de PHB.

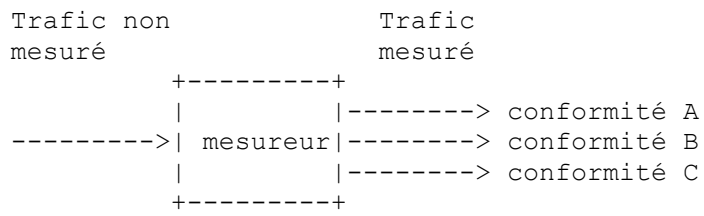

Figure 4 : Mesureur générique

Un mesureur, selon ce modèle, mesure le débit auquel les paquets constituant un flux de trafic passent, comparé au débit d'un ensemble de seuils, et produit un certain nombre de résultats potentiels (deux ou plus) : un certain paquet est dit être "conforme" à un niveau du mesureur si, au moment de l'examen du paquet, le flux paraît être dans les limites de débit pour le profil associé à ce niveau. Une discussion plus complète de la conformité aux profils de mesureurs (et des exigences associées qui pèsent sur les programmateurs en amont) figure en Appendice A.

5.1 Exemples

Voici des exemples de mesureurs possibles.

5.1.1 Mesureur de débit moyen

Un exemple de mesureur très simple est un mesureur de débit moyen. Ce type de mesureur mesure le débit moyen auquel les paquets lui sont soumis pendant un délai spécifié.

Un profil de débit moyen peut prendre la forme suivante :

```
Mesureur1 :
Type :                DébitMoyen
Profil :              Profil1
SortieConforme :      Queue1
SortieNonConforme : Compteur1

Profil1 :
Type :          DébitMoyen
DébitMoyen :    120 kbit/s
Delta :         100 ms
```

Un mesureur qui mesure par rapport à ce profil va tenir un compte en continu qui indique le nombre total et/ou le compte cumulatif d'octets des paquets qui arrivent entre le temps T (maintenant) et le temps T - 100 ms. Tant qu'un paquet arrivant ne pousse pas le compte au delà de 12 kbits dans les dernières 100 ms, le paquet va être réputé conforme. Tout paquet qui pousse le compte au delà de 12 kbits va être réputé non conforme. Donc, ce mesureur suppose que les paquets correspondent à un des deux niveaux de conformité : conforme ou non conforme, et les envoie sur le traitement ultérieur approprié.

5.1.2 Mesureur de moyenne mobile à pondération exponentielle

La forme moyenne mobile pondérée exponentielle (EWMA, *Exponential Weighted Moving Average*) d'un mesureur est facile à mettre en œuvre dans le matériel et peut être paramétrée comme suit :

débit_moyen(t) = (1 - Gain) * débit_moyen(t') + Gain * débit(t)
t = t' + Delta

Pour un paquet qui arrive à l'instant t :

```
si (débit_moyen(t) > DébitMoyen)
  non conforme
autrement
  conforme
```

"Gain" contrôle la constante de temps (par exemple, réponse en fréquence) de ce qui est essentiellement un simple filtre IIR passe bas. "Débit(t)" mesure le nombre d'octets entrants dans un petit intervalle d'échantillonnage fixé, Delta. Tout paquet qui arrive et pousse le débit moyen au dessus d'un débit prédéfini DébitMoyen est réputé non conforme. Un profil de mesureur EWMA pourrait ressembler à quelque chose comme :

```
Mesureur2 :
Type :                MoyMobPondExp
Profil :              Profil 2
SortieConforme :      Queue1
SortieNonConforme : AbandonneurAbsolu1

Profil2 :
Type :          MoyMobPondExp
DébitMoyen :    25 kbit/s
Delta :         10 µs
Gain :          1/16
```

5.1.3 Mesureur de baquet de jetons à deux paramètres

Un mesureur plus sophistiqué peut mesurer la conformité à un profil de baquet de jetons (TB, *token-bucket*). Un profil TB a en général deux paramètres, un débit de jetons moyen, R, et une taille de salve, B. Les mesureurs TB comparent le débit d'arrivée des paquets au débit moyen spécifié par le profil TB. Logiquement, les jetons s'accumulent dans un baquet au taux moyen, R, jusqu'à un crédit maximum qui est la taille de salve, B. Lorsque un paquet de longueur L arrive, un essai de conformité est appliqué. Deux de ces essais sont largement utilisés :

Conformité stricte
Les paquets d'une longueur de L octets ne sont considérés conformes que si il y a suffisamment de jetons disponibles dans le baquet au moment de l'arrivée du paquet pour le paquet complet (c'est-à-dire que la profondeur actuelle est supérieure ou égale à L) : aucun jeton ne peut être emprunté aux futures allocations de jetons. Pour des exemples de cette approche, voir la [RFC2697] et la [RFC2698].

Conformité lâche
Les paquets d'une longueur de L octets sont considérés comme conformes si un nombre quelconque de jetons est disponible dans le baquet au moment de l'arrivée du paquet : jusqu'à L octets peuvent alors être empruntés aux allocations futures de jetons.

Il est permis aux paquets de dépasser le débit moyen dans des salves allant jusqu'à la taille de salve. L'Appendice A contient une discussion plus poussée des profils de conformité stricte et lâche de baquet de jetons, ainsi que des questions de système et de mise en œuvre.

Un mesureur TB à deux paramètres a exactement deux niveaux de conformité possibles (conforme, non conforme). Un tel mesureur peut apparaître comme suit :

Mesureur3 :
Type : SimpleBaquetDeJetons
Profil : Profil3
TypeConformité : lâche
SortieConforme : Queue1
SortieNonConforme : AbandonneurAbsolu1

Profil3 :
Type : SimpleBaquetDeJetons
DébitMoyen : 200 kbit/s
Taille Salve : 100 koctets

5.1.4 Mesureur de baquet de jetons multi étapes

Des mesureurs TB plus compliqués peuvent définir plusieurs tailles de salve et plus de niveaux de conformité. Les paquets qui excèdent la plus grande taille de salve sont réputés non conformes. Les paquets qui excèdent la plus petite taille de salve sont réputés partiellement conformes. Les paquets qui n'excèdent ni l'une ni l'autre sont réputés conformes. Certains mesureurs de baquet de jetons conçus pour les réseaux Diffserv sont décrits plus en détails dans la [RFC2697] et la [RFC2698] ; dans certaines de ces références, trois niveaux de conformité sont discutés en termes de couleurs avec le vert qui représente la conformité, le jaune qui représente la conformité partielle, et le rouge qui représente la non conformité. Noter que ces mesureurs à plusieurs niveaux de conformité peuvent parfois être mis en œuvre en utilisant une séquence appropriée de plusieurs mesureurs TB à deux paramètres.

Un profil pour mesureur TB multi étapes avec trois niveaux de conformité pourrait ressembler à ceci :

Mesureur4 :
Type : BaquetJetonsDeuxDébits
ProfilA : Profil4
ConformitéTypeA : stricte
SortieConformeA : Queue1

ProfilB : Profil5
ConformitéTypeB : stricte
SortieConformeB : Marqueur1
SortieNonConforme : AbandonneurAbsolu1

Profil4 :
Type : SimpleBaquetJetons
DébitMoyen : 100 kbit/s
TailleSalve : 20 koctets

Profil5 :
Type : SimpleBaquetJetons
DébitMoyen : 100 kbit/s
TailleSalve : 100 koctets

5.1.5 Mesureur nul

Un mesureur nul a seulement une sortie : toujours conforme, et pas de profil temporel associé. Il est parfois utile de définir un tel mesureur lorsque l'interface de configuration ou de gestion n'a pas la souplesse d'omettre un mesureur dans un segment du chemin des données.

Mesureur5 :
Type : MesureurNul
Sortie : Queue1

6. Éléments d'action

Les classeurs et mesureurs décrits jusqu'ici sont des éléments de ventilation qui sont généralement utilisés pour déterminer l'action appropriée à appliquer à un paquet. L'ensemble des actions qui peuvent alors être appliquées inclut :

- le marquage
- l'abandon absolu
- le multiplexage
- le comptage
- l'action nulle – ne rien faire

Les éléments d'action correspondants sont décrits dans les paragraphes qui suivent.

6.1 Marqueur DSCP

Les marqueurs DSCP sont des éléments 1:1 qui établissent un codet (par exemple, le DSCP dans un en-tête IP). Les marqueurs DSCP peuvent aussi agir sur les paquets non marqués (par exemple, ceux soumis avec un DSCP de zéro) ou peuvent marquer à nouveau des paquets marqués précédemment. En particulier, le modèle prend en charge l'application d'un marquage fondé sur une correspondance précédente de classeur. La marque établie dans un paquet va déterminer son traitement de PHB ultérieur dans les nœuds aval d'un réseau et éventuellement dans les étapes suivantes du traitement au sein de ce routeur.

Les marqueurs DSCP pour Diffserv sont normalement paramétrés par un seul paramètre : le DSCP de six bits à marquer dans l'en-tête du paquet.

Marqueur1 :
Type : MarqueurDSCP
Marque : 010010

6.2 Abandonneur absolu

Les abandonneurs absolus éliminent simplement les paquets. Il n'y a pas de paramètre pour ces abandonneurs. Parce que cet abandonneur absolu est un point de terminaison du chemin des données et n'a pas de sortie, il est probablement souhaitable de transmettre d'abord le paquet à travers une action de compteur pour les besoins de l'instrumentation.

AbandonneurAbsolu1 :
Type : AbandonneurAbsolu

Les abandonneurs absolus ne sont pas les seuls éléments qui peuvent causer l'élimination d'un paquet : un autre élément est un élément d'abandonneur algorithmique (voir le paragraphe 7.1.3). Cependant, comme le comportement de cet élément est étroitement lié à l'état d'une ou plusieurs files d’attente, on a choisi de le distinguer comme élément séparé du chemin fonctionnel des données.

6.3 Multiplexeur

Il est occasionnellement nécessaire de multiplexer le flux de trafic en un élément fonctionnel de chemin de données avec une seule entrée. Un multiplexeur M:1 (concentrateur) est un simple appareil logique pour fusionner des flux de trafic. Il est paramétré par son nombre d'accès entrants.

Mux1 :
Type : Multiplexeur
Sortie : Queue2

6.4 Compteur

Une action passive est de tenir compte du fait qu'un paquet de données a été traité. Les statistiques qui en résultent peuvent être utilisées ultérieurement pour la facturation des usagers, la vérification des services ou pour des besoins d'ingénierie du réseau. Les compteurs sont des éléments fonctionnels du chemin des données 1:1 qui mettent à jour un compteur de L et un compteur de paquets de 1 chaque fois qu'un paquet de taille L octets passe à travers eux. Les compteurs peuvent être utilisés pour compter les paquets sur le point d'être éliminés par un abandonneur absolu ou pour compter les paquets qui arrivent ou partent de quelque autre élément fonctionnel.

Compteur1 :
Type : Compteur
Sortie : Queue1

6.5 Action nulle

Une action nulle a une entrée et une sortie. L'élément n'effectue aucune action sur le paquet. La définition d'un tel élément est utile dans l'éventualité d'une interface de configuration ou de gestion qui n'a pas la souplesse d'omettre un élément d'action dans un segment du chemin des données.

Nul1 :
Type : Nul
Sortie : Queue1

7. Éléments de mise en file d'attente

Les éléments de mise en file d'attente modulent la transmission de paquets appartenant aux différents flux de trafic et déterminent leur ordre, éventuellement en les mémorisant temporairement ou en les éliminant. Les paquets sont usuellement mémorisés soit parce qu'il y a des contraintes de ressources (par exemple, la bande passante disponible) qui empêchent la transmission immédiate, soit parce que le bloc de mise en file d'attente est utilisé pour altérer les propriétés temporelles d'un flux de trafic (c'est-à-dire, le formatage). Les paquets sont éliminés pour une des raisons suivantes :
- à cause de limitations de mémoire tampon ;
- à cause du dépassement d'un seuil de mémoire tampon (incluant quand le formatage est effectué) ;
- comme signal de contrôle de retour pour des protocoles de contrôle réactifs comme TCP ;
- à cause du dépassement d'un profil configuré par un mesureur (c'est-à-dire, une régulation).

Les éléments de mise en file d'attente dans ce modèle représentent une abstraction logique d'un système de mise en file d'attente qui est utilisé pour configurer des paramètres en rapport avec le PHB. Le modèle peut être utilisé pour représenter une grande variété de mises en œuvre possibles. Cependant, il ne se transpose pas nécessairement de façon biunivoque avec les systèmes de mise en file d'attente physique dans une mise en œuvre spécifique de routeur. Les mises en œuvre devraient transposer les paramètres configurables des systèmes de mise en file d'attente des mises en œuvre en les paramètres d'éléments de mise en file d'attente qui sont appropriés pour réaliser des comportements équivalents.

7.1 Modèle de mise en file d'attente

La mise en file d'attente est une fonction qui la prête à l'innovation. Elle doit être modélisée pour permettre de représenter une large gamme de possibles mises en œuvre en utilisant des structures et paramètres communs. Le présent modèle utilise la décomposition fonctionnelle comme outil pour permettre la latitude nécessaire.

Les systèmes de mise en file d'attente effectuent trois fonctions distinctes mais en relations entre elles : elles mémorisent les paquets, elles modulent le départ des paquets qui appartiennent aux divers flux de trafic et elles éliminent les paquets de façon sélective. Le présent modèle décompose la mise en file d'attente en les éléments composants qui chacun effectuent des fonctions : respectivement, files d'attente, programmateurs, et abandonneurs algorithmiques. Ces éléments peuvent être connectés ensemble au titre d'un TCB, comme décrit à la Section 8.

Le reste de cette section discute des files d'attente FIFO *(First In, First Out)* : normalement, l'élément File d'attente de ce modèle va être mis en œuvre comme une structure de données FIFO. Cependant, cela n'empêche pas les mises en œuvre qui ne sont pas strictement FIFO, en ce qu'elle prennent aussi en charge des opérations qui suppriment ou examinent les paquets (par exemple, pour l'usage des élimineurs) ailleurs qu'en tête ou en queue. Cependant, de telles opérations ne doivent pas avoir pour effet de réarranger des paquets qui appartiennent au même microflux.

Noter que le terme FIFO a plusieurs usages courants différents : il est parfois pris pour signifier, entre autres choses, une structure de données qui permet de ne retirer des éléments que dans l'ordre dans lequel ils ont été insérés, ou une discipline de service qui ne réarrange pas.

7.1.1 File d'attente FIFO

Dans le présent modèle, un élément de file d'attente FIFO est une structure de données qui à tout instant peut contenir zéro, un ou plusieurs paquets. Elle peut avoir un ou plusieurs seuils associés. Une file d'attente FIFO a une ou plusieurs entrées et exactement une sortie. Elle doit supporter une opération en queue pour ajouter un paquet à la queue de la file d'attente et une opération de sortie de file d'attente pour retirer un paquet de la tête de la file d'attente. Les paquets doivent être sortis de la file

d'attente dans l'ordre dans lequel ils ont été mis en file d'attente. Une file d'attente FIFO a une profondeur actuelle, qui indique le nombre de paquets et/ou octets qu'elle contient à un instant donné. Les files d'attente FIFO dans le présent modèle sont modélisés sans limites inhérentes à leur profondeur – cela ne reflète évidemment pas la réalité des mises en œuvre : les limites de taille des files d'attente FIFO sont modélisées ici par un abandonneur algorithmique associé, normalement à son entrée. Il est assez probable que toute file d'attente FIFO sera précédée par un abandonneur algorithmique. Une exception pourrait être lorsque le flux de paquets a déjà été régulé par un profil qui ne peut jamais excéder la bande passante disponible du programmateur à la sortie de la file d'attente FIFO – cela ne nécessiterait pas un abandonneur algorithmique à l'entrée de la file d'attente FIFO.

Cette représentation d'une file d'attente FIFO permet un type commun de limite de profondeur, qui résulte d'un réservoir limité de mémoires tampons fourni par une file d'attente FIFO, partagé entre plusieurs files d'attentes FIFO.

Dans une mise en œuvre, les paquets sont présumés être mémorisés dans une ou plusieurs mémoires tampons. Celles-ci sont allouées à partir d'un ou plusieurs réservoirs de mémoires tampons libres. Si il y a plusieurs instances de file d'attente FIFO, leurs mémoires tampons de paquets peuvent ou non être allouées à partir du même réservoir de mémoires tampons libres. Les réservoirs de mémoire tampon libres peuvent aussi avoir un ou plusieurs seuils associés, qui peuvent affecter l'élimination et/ou la programmation. À part cela, les mécanismes de mise en mémoire tampon sont spécifiques de la mise en œuvre et ne font pas partie du modèle.

Une file d'attente FIFO peut être représentée en utilisant les paramètres suivants :

Queue1 :
Type : FIFO
Sortie : Programmateur1

Noter qu'une file d'attente FIFO doit fournir des déclencheurs et/ou les informations d'état en cours aux autres éléments amont et avals : en particulier, il est probable que la profondeur actuelle devra être utilisée par les éléments d'abandonneur algorithmique placés avant ou après la file d'attente FIFO. Elle devra aussi probablement fournir un signal implicite "j'ai des paquets pour vous" aux éléments de programmeur vers l'aval.

7.1.2 Programmateur

Un programmateur est un élément qui provoque le départ de chaque paquet qui arrive à une de ses entrées, sur la base d'une discipline de service. Il a une ou plusieurs entrées et exactement une sortie. Chaque entrée a un élément amont auquel elle est connectée, et un ensemble de paramètres qui affectent la programmation des paquets reçus à cette entrée.

La discipline de service (aussi appelée algorithme de programmation) est un algorithme qui peut prendre toutes les entrées suivantes :

a) des paramètres statiques comme une priorité relative associée à chacune des entrées du programmateur ;
b) des paramètres de baquet de jetons absolu pour les débits maximum ou minimum associés à chacune des entrées du programmateur ;
c) des paramètres, comme une longueur de paquet ou un DSCP, associés au paquet actuellement présent à son entrée ;
d) l'heure absolue et/ou l'état local.

Les disciplines de service possibles entrent dans un certain nombre de catégories, incluant (sans s'y limiter) le premier entré premier servi (FCFS, *first come, first served*), la priorité stricte, le partage de bande passante pondéré équitable (par exemple, WFQ), la priorité stricte à débit limité, et fondé sur le débit. Les disciplines de services peuvent encore être distinguées par si elles sont conservatrices du travail ou non (voir le Glossaire). Les programmateurs non conservateurs du travail peuvent être utilisés pour formater les flux de trafic à correspondre à un certain profil en retardant les paquets qui pourraient être réputés non conformes par un nœud aval : un paquet est retardé jusqu'au moment où il va se conformer à un mesureur aval qui utilise le même profil.

La [RFC2475] définit des PHB sans spécifier les algorithmes de programmation requis. Cependant, des PHB comme les sélecteurs de classe [RFC2474], EF [RFC3246] et AF [RFC2597] ont des descriptions ou des paramètres de configuration qui suggèrent fortement la sorte de discipline de programmation nécessaire pour les mettre en œuvre. Le présent document discute un ensemble minimal de paramètres de file d'attente pour permettre la réalisation de ces PHB. Il ne tente pas de spécifier un ensemble de paramètres couvrant tous les modèles de mise en œuvre possibles. Un ensemble minimal inclut :

a) un profil de débit de service minimum qui permet des garanties de débit pour chaque flux de trafic comme requis par EF et AF sans spécifier les détails de comment est partagé entre ces flux de trafic l'excès de bande passante. Des paramètres supplémentaires pour contrôler ce comportement devraient être disponibles, mais dépendent de l'algorithme de programmation mis en œuvre.

b) une priorité de service, utilisée seulement après que les profils de débit minimum de toutes les entrées ont été satisfaites, pour décider comment allouer la bande passante restante.

c) un profil de débit de service maximum, à n'utiliser qu'avec une discipline de service non conservatrice du travail.

Chacun de ces profils est composé, pour les besoins du présent modèle, d'un débit (en unités convenables de bits, d'octets ou plus grands tronçons d'une certaine unité de temps) et d'une taille de salve, comme exposé plus en détails à l'Appendice A.

Par exemple, pour une mise en œuvre du PHB EF utilisant un algorithme de stricte priorité qui suppose que le débit EF agrégé a été bordé de façon appropriée par une régulation en amont pour éviter d'affamer les autres BA, les profils de débit de service ne sont pas utilisés : le profil de service minimum serait par défaut de zéro et le profil de service maximum serait effectivement le "débit de ligne". Une telle mise en œuvre, avec de multiples classes de priorité, pourrait aussi être utilisée pour les sélecteurs de classe Diffserv [RFC2474].

Autrement, régler les valeurs de priorité de service pour chaque entrée au programmateur à la même valeur permet au programmateur de satisfaire les débits de service minimum pour chaque entrée, tant que la somme de tous les débits de service minimum est inférieure ou égale au débit de ligne.

Par exemple, un programmateur non conservateur de travail, qui alloue également la bande passante restante entre toutes ses entrées, pourrait être représenté en utilisant les paramètres suivants :

Programmateur1 :
Type : Programmateur2Entrée
Entrée1 :
ProfilDébitMax : Profil1
ProfilDébitMin : Profil2
Priorité : aucune

Entrée2 :
ProfilDébitMax : Profil3
ProfilDébitMin : Profil4
Priorité : aucune

Un programmateur conservateur du travail pourrait être représenté en utilisant les paramètres suivants :

Programmateur2:
Type: Programmateur3Entrée
Entrée1 :
ProfilDébitMax : ConserveTravail
ProfilDébitMin : Profil5
Priorité : 1

Entrée2 :
ProfilDébitMax : ConserveTravail
ProfilDébitMin : Profil6
Priorité : 2

Entrée3 :
ProfilDébitMax : ConserveTravail
ProfilDébitMin : aucune
Priorité : 3

7.1.3 Abandonneur algorithmique

Un abandonneur algorithmique est un élément qui élimine de façon sélective les paquets qui arrivent à son entrée, sur la base d'un algorithme d'élimination. Il a une entrée de données et une sortie. Dans le présent modèle (mais pas nécessairement dans une mise en œuvre réelle) un paquet entre dans l'abandonneur comme entrée et soit sa mémoire tampon est retournée à un réservoir de mémoire tampon libre, soit le paquet sort de l'abandonneur par la sortie.

Autrement, un abandonneur algorithmique peut être vu comme invoquant des opérations sur une file d'attente FIFO qui retire sélectivement un paquet et retourne sa mémoire tampon au réservoir de mémoire tampon libre sur la base d'un algorithme d'élimination. Dans ce cas, l'opération pourrait être modélisée comme étant un effet collatéral du fonctionnement d'une file

d'attente FIFO, plutôt que comme ayant une entrée et une sortie discrètes. Ce traitement est équivalent et on choisit pour le modèle celui décrit au paragraphe précédent.

Une des principales caractéristiques d'un abandonneur algorithmique est le choix du paquet (s'il en est) à éliminer : pour les besoins du présent modèle, on restreint le choix du paquet à un de ceux qui suivent et on indique le choix par les positions relatives des éléments Abandonneur algorithmique et File d'attente FIFO dans le modèle :

a) Le choix d'un paquet qui va être ajouté à la fin d'une file d'attente (un "abandonneur de queue") : le résultat de l'élément d'abandonneur algorithmique est connecté à l'entrée de l'élément pertinent de file d'attente FIFO.

b) Un paquet qui est actuellement en tête d'une file d'attente (un "abandonneur de tête") : le résultat de l'élément de file d'attente FIFO est connecté à l'entrée de l'élément d'abandonneur algorithmique.

D'autres méthodes de choix de paquet pourraient être ajoutées à ce modèle sous la forme d'un type différent d'élément du chemin des données.

L'abandonneur algorithmique est modélisé comme ayant une seule entrée. Il est possible que des paquets qui ont été classés différemment par un classeur dans ce TCB vont finir par passer à travers le même abandonneur. L'algorithme d'abandonneur peut avoir besoin d'appliquer des calculs différents sur la base de caractéristiques du paquet entrant (par exemple, son DSCP). Il y a donc besoin, dans les mises en œuvre de ce modèle, d'être capable de mettre en relation les informations sur l'élément de classeur auquel a été confronté un paquet provenant d'un classeur avec un abandonneur algorithmique. Dans les rares cas où ceci est requis, le modèle choisi est d'insérer un autre élément Classeur à ce point du flux et de l'alimenter dans plusieurs éléments d'abandonneur algorithmique dont chacun met en œuvre un calcul d'abandon qui est indépendant de toute clé de classement du paquet : cela va probablement exiger la création d'un nouveau TCB pour contenir les éléments de classeur et d'abandonneur algorithmique.

Note : Il y a beaucoup d'autres formulations d'un modèle qui pourraient représenter cette liaison qui sont différentes de celle décrite ci-dessus : une formulation serait d'avoir un pointeur d'un des algorithmes de calcul de probabilité d'abandon à l'intérieur de l'abandonneur de l'élément Classeur d'origine qui choisit cet algorithme. Une autre façon serait d'avoir plusieurs "entrées" à l'élément d'abandonneur algorithmique alimentées à partir des éléments précédents, conduisant finalement à retourner aux éléments de classeur qui correspondaient au paquet. Encore une autre formulation serait que le classeur inclue (logiquement) une sorte de "identifiant de classification" qui suive le paquet le long de son chemin, pour être utilisé par tout élément suivant. Et encore une autre qui pourrait être d'inclure un classeur à l'intérieur de l'abandonneur, afin qu'il choisisse l'algorithme d'abandon à appliquer. Ces autres approches pourraient être utilisées par les mises en œuvre mais sont réputées moins claires que celle retenue ici.

Un abandonneur algorithmique, dont un exemple est illustré à la Figure 5, a un ou plusieurs déclencheurs qui lui font prendre une décision d'abandonner ou non un (ou éventuellement plus d'un) paquet. Un déclencheur peut être interne (à l'arrivée d'un paquet à l'entrée de l'abandonneur) ou il peut être externe (résultant d'un ou plusieurs changements d'état à un autre élément, comme la profondeur d'une file d'attente FIFO qui passe un seuil ou un événement de programmation). Il est probable qu'une profondeur instantanée de file d'attente FIFO devra être lissée sur un intervalle moyen avant d'être utilisée comme déclencheur utile. Certains algorithmes d'abandon peuvent exiger plusieurs entrées de déclenchement rapportant des retours d'événements ailleurs dans le système (par exemple, des fonctions de lissage de profondeur qui calculent des moyennes sur plus d'un intervalle de temps).

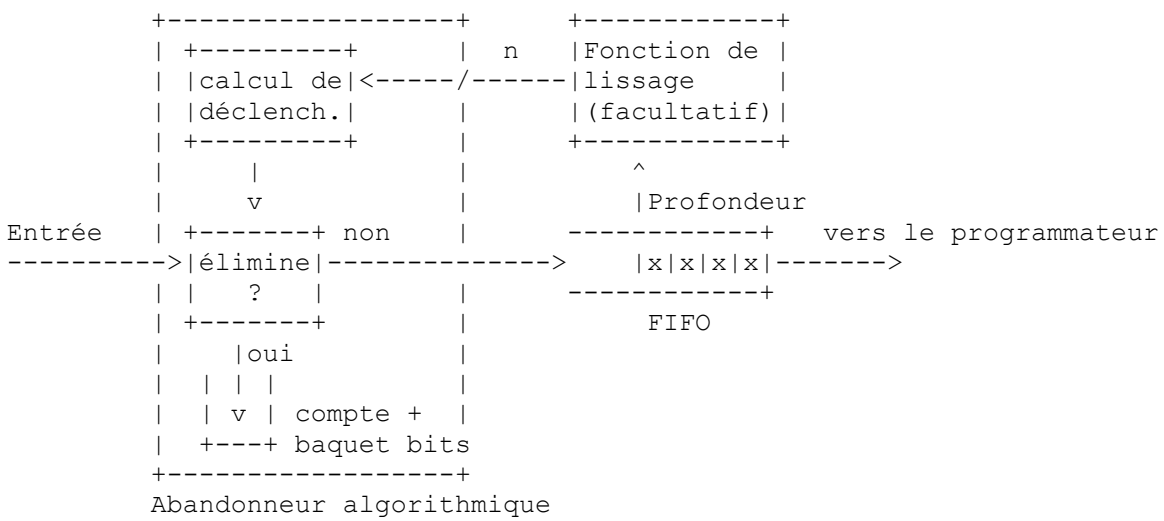

Figure 5 : Exemple d'abandonneur algorithmique de queue d'une file d'attente

Un déclencheur peut être une combinaison booléenne d'événements (par exemple, une profondeur de file d'attente FIFO qui excède un seuil OU une profondeur de réservoir de mémoire tampon qui tombe en dessous d'un seuil). Il prend en entrée un ensemble de paramètres dynamiques (par exemple, une profondeur de file d'attente FIFO lissée ou instantanée) et un ensemble de paramètres statiques (par exemple, des seuils) et éventuellement d'autres paramètres associés au paquet. Il peut aussi avoir un état interne (par exemple, l'historique de ses actions). Noter que, bien qu'un abandonneur algorithmique puisse exiger la connaissance des champs de données dans un paquet, comme le découvre un classeur dans le même TCB, il peut ne pas modifier le paquet (c'est-à-dire que ce n'est pas un marqueur).

Le résultat du calcul de déclencheur est que l'algorithme d'abandon prend une décision de transmettre ou d'éliminer un paquet. La fonction d'élimination va probablement tenir des compteurs concernant les paquets éliminés (il n'y a ici pas d'endroit approprié pour inclure un élément d'action de compteur).

L'exemple de la Figure 5 montre aussi un élément de file d'attente FIFO dont la queue va être le lieu de l'abandon et dont les caractéristiques de profondeur sont utilisées par cet abandonneur algorithmique. Il montre aussi où une fonction de lissage de profondeur peut être incluse : les fonctions de lissage sortent du domaine d'application du présent document et ne sont pas modélisées explicitement ici ; on indique simplement où elles peuvent être ajoutées.

RED, RED-on-In-and-Out (RIO) et Drop-on-threshold sont des exemples d'algorithmes d'abandon. L'abandon de tête et l'abandon de queue sont affectés par la localisation de l'élément d'abandonneur algorithmique relatif à l'élément de file d'attente FIFO. Par exemple, un abandonneur qui utilise un algorithme RIO pourrait être représenté en utilisant deux abandonneurs algorithmiques avec les paramètres suivants :

AbandonneurAlgorithmique1 : (pour du trafic dans le profil)
- Type : AbandonneurAlgorithmique
- Discipline : RED
- Déclencheur : Interne
- Sortie : Fifo1
- SeuilMin : Fifo1.Profondeur > 20 koctet
- SeuilMax : Fifo1.Profondeur > 30 koctet
- PoidsÉchantillon : 0,002
- ProbMaxAbandon : 1 %

AbandonneurAlgorithmique2 : (pour trafic hors profil)
- Type : AbandonneurAlgorithmique
- Discipline : RED
- Déclencheur : Interne
- Sortie : Fifo1
- SeuilMin : Fifo1.Profondeur > 10 koctet
- SeuilMax : Fifo1.Profondeur > 20 koctet
- PoidsÉchantillon : 0,002
- ProbMaxAbandon : 2 %

Une autre forme d'abandonneur algorithmique, un abandonneur de seuil, peut être représenté en utilisant les paramètres suivants :

AbandonneurAlgorithmique3 :
- Type : AbandonneurAlgorithmique
- Discipline : Abandon sur seuil
- Déclencheur : Fifo2.Profondeur > 20 koctet
- Sortie : Fifo1

7.2 Partage de charge entre flux de trafic utilisant la mise en file d'attente

Les files d'attente sont utilisées, dans les services différenciés, pour un certain nombre d'objets. Par essence, ce sont simplement des endroits pour mémoriser le trafic jusqu'à sa transmission. Cependant, lorsque plusieurs files d'attente sont utilisées ensemble dans un système de file d'attente, elles peuvent aussi avoir des effets au delà d'un certain flux de trafic. Elles peuvent être utilisées pour limiter les variations de délai ou imposer un débit maximum (formatage) pour permettre que plusieurs flux partagent une liaison d'une façon semi-prévisible (partage de charge) ou pour déplacer une variation de délai de certains flux à d'autres flux.

Le formatage de trafic est souvent utilisé pour conditionner le trafic, de telle sorte que les paquets qui arrivent en salve soient "lissés" et réputés conformes par les mesureurs suivants vers l'aval dans ce nœud ou d'autres. Dans la [RFC2475] un formateur est décrit comme un élément de mise en file d'attente contrôlé par un mesureur qui définit son profil temporel. Cependant, cette

représentation d'un formateur diffère substantiellement des mises en œuvre typiques de formateur.

Dans le modèle qu'on décrit ici, un formateur est réalisé en utilisant un programmateur sans conservation du travail. Certaines mises en œuvre peuvent choisir d'avoir des files d'attente dont le seul propos est le formatage, tandis que d'autres peuvent intégrer la fonction de formatage avec celles de mise en mémoire tampon, d'élimination, et de programmation associées à l'accès à une ressource. Les formateurs opèrent en retardant le départ des paquets qui auraient été réputés non conformes par un mesureur configuré au profil de débit de service maximum du formateur. Le paquet est programmé pour partir à un instant tel qu'il va devenir conforme.

7.2.1 Partage de charge

Le partage de charge est l'utilisation traditionnelle des files d'attente et il a été exploré théoriquement par Floyd & Jacobson [FJ95], bien qu'il ait été utilisé dans les systèmes de communications depuis les années 1970.

La [RFC2475] discute du partage de charge comme divisant un interface entre des classes de trafic de façon prévisible, ou en appliquant un débit minimum à chacune des classes de trafic d'un ensemble de classes, qui peut être mesuré comme une limite inférieure absolue du débit que réalise un flux de trafic ou une fraction du débit qu'offre une interface. Il est généralement mis en œuvre comme une forme d'algorithme de mise en file d'attente pondérée parmi un ensemble de files d'attente FIFO, c'est-à-dire, un schéma de WFQ. Cela a des effets collatéraux intéressants.

Un effet clé recherché est d'assurer que le débit moyen que subit le trafic d'un flux n'est jamais inférieur à un certain seuil quand il y a au moins cette quantité de trafic à envoyer. Quand il y a moins de trafic que ce seuil, la file d'attente tend à être privée de trafic, ce qui signifie que le système de mise en file d'attente ne va pas retarder beaucoup son trafic. Quand il y a significativement plus de trafic et que la file d'attente commence à se remplir, les paquets dans cette classe vont être retardés significativement plus que le trafic dans les autres classes qui sous utilisent leur capacité disponible. Cette forme de système de mise en file d'attente tend donc à déplacer le retard et la variation de retard des classes de trafic sous utilisées aux plus gros utilisateurs, ainsi qu'à gérer les débits des flux de trafic.

Un effet collatéral de la mise en œuvre WRR ou WFQ est qu'entre deux paquets d'une certaine classe de trafic, le programmateur peut émettre un ou plusieurs paquets de chacune des autres classes du système de mise en file d'attente. Dans les cas où on vise un comportement moyen, cela est parfaitement acceptable. Dans les cas où le trafic est très intolérant à la gigue et où il y a un certain nombre de classes en compétition, cela peut avoir des conséquences indésirables.

7.2.2 Priorité de trafic

La priorité de trafic est un cas particulier de partage de charge, dans lequel une certaine classe de trafic est réputée si intolérante à la gigue que si elle a du trafic présent, ce trafic doit être envoyé au plus tôt. Par extension, plusieurs priorités peuvent être définies, telles que le trafic dans chacune des diverses classes reçoive un service préférentiel sur tout trafic d'une classe inférieure. C'est la mise en œuvre évidente de la préséance IP décrite dans la [RFC0791], des classes de trafic 802.1p [802.1D], et autres technologies similaires.

La priorité est souvent bafouée dans les réseaux réels ; les gens tendent à penser que le trafic d'affaires à haute priorité mérite ce traitement et parlent plus des impératifs des affaires que des exigences d'application réelles. Ceci peut avoir des conséquences sérieuses ; des réseaux ont été configurés avec du trafic critique pour les affaires placé à une priorité plus élevée que le trafic de protocole d'acheminement, d'où résultent des écroulements des systèmes de gestion ou de contrôle de réseau. Cependant, elle peut avoir une utilisation légitime pour les services fondés sur un PHB de transmission expédiée (EF, *Expedited Forwarding*) où on est absolument sûr, grâce à la régulation à tous les points d'entrée possibles du trafic, qu'un flux de trafic n'abuse pas de son débit et que l'application est assez intolérante à la gigue pour mériter ce type de traitement. Noter que, même dans les cas de points d'entrées bien régulés, il y a quand même la possibilité de boucles de trafic inattendues au sein d'une partie de cœur non régulée du réseau qui causerait un tel effondrement.

8. Blocs de conditionnement de trafic

Les éléments fonctionnels du chemin des données de classeur, mesureur, action, abandonneur algorithmique, file d'attente et programmateur décrits ci-dessus peuvent être combinés dans des blocs de conditionnement de trafic (TCB, *Traffic Conditioning Block*). Un TCB est une abstraction d'un ensemble d'éléments fonctionnels du chemin des données qui peut être utilisé pour faciliter la définition d'une fonction de conditionnement de trafic spécifique (par exemple, il peut être relié à un gabarit qui peut être reproduit de nombreuses fois pour différents flux de trafic ou différents clients). Il n'a pas de représentation physique vraisemblable dans la mise en œuvre du chemin des données : il est purement inventé comme abstraction à l'usage des outils de gestion.

Le présent modèle décrit la configuration et la gestion d'une interface Diffserv en termes de TCB qui contient, par définition, zéro, un ou plusieurs éléments de classeur, mesureur, action, abandonneur algorithmique, file d'attente et programmateur. Ces éléments sont arrangés arbitrairement selon la politique exprimée, mais toujours dans l'ordre ci-dessus. Le trafic peut être

classé ; le trafic classé peut être mesuré ; chaque flux de trafic identifié par une combinaison de classeurs et mesureurs peut avoir un ensemble d'actions effectuées sur lui, suivi par des algorithmes d'abandon ; les paquets du flux de trafic peuvent finalement être mémorisés dans une file d'attente et ensuite programmés en sortie sur le prochain TCB ou interface physique. Il est permis d'omettre des éléments ou d'inclure des éléments nuls de tout type, ou d'enchaîner plusieurs éléments fonctionnels du chemin des données du même type.

Lorsque le traitement Diffserv pour un certain paquet doit répéter de tels blocs de construction, c'est effectué en mettant en cascade plusieurs TCB : une sortie d'un TCB peut piloter l'entrée du suivant. Par exemple, considérons le cas où le trafic d'un ensemble de classes est formaté en un ensemble de débits, mais où le débit de sortie total du groupe de classes doit aussi être limité à un débit. On peut imaginer un ensemble de nouvelles alimentations de réseau, chacune avec un certain débit maximum, et une politique disant que leur agrégation ne doit pas excéder une certaine valeur. Cela peut être réalisé simplement en mettant en cascade deux TCB. Le premier classe le trafic en deux alimentations séparées et met en files d'attente séparément chaque alimentation. Les alimentations (ou un sous ensemble de celles-ci) sont maintenant nourries dans un second TCB, qui place toutes les entrées (ces nouvelles alimentations) dans une seule file d'attente avec un certain débit maximum. Dans la mise en œuvre, on peut imaginer cela comme plusieurs files d'attente littérales, un système CBQ ou WFQ avec un schéma de pondération approprié (et complexe) ou un certain nombre d'autres approches. Mais elles auraient le même effet mesurable extérieurement sur le trafic que si elles avaient été littéralement mises en œuvre avec des TCB séparés.

8.1 TCB

Un TCB généralisé pourrait comporter les étapes suivantes :
- Étape de classement
- Étape de mesure
- Étape d'action (impliquant des marqueurs, des abandonneurs absolus, des compteurs, et des multiplexeurs)
- Étape de mise en file d'attente (impliquant des abandonneurs algorithmiques, des files d'attente, et des programmateurs)

où chaque étape peut consister en un ensemble de chemins de données parallèles consistant en éléments traités en parallèle.

Un classeur ou un mesureur est normalement un élément 1:N, une action, un abandonneur algorithmique, ou une file d'attente est normalement un élément 1:1 et un programmateur est un élément N:1. Un TCB complet devrait, cependant, résulter en un élément abstrait 1:1 ou 1:N. Noter que la ventilation ou la contraction d'un élément n'est pas une caractéristique de définition importante de cette taxonomie.

8.1.1 Blocs de construction de la mise en file d'attente

Certaines règles particulières sont appliquées à l'ordre des éléments au sein d'une étape de mise en file d'attente dans un TCB : les éléments du même type peuvent apparaître plus d'une fois, soit en parallèle, soit en série. Normalement, une étape de mise en file d'attente aura relativement beaucoup d'éléments en parallèle et peu en série. L'itération et la récurrence ne sont pas des constructions acceptées (les éléments sont arrangés dans un graphe acyclique). Les interconnexions d'éléments suivantes sont permises :
- L'entrée d'une file d'attente peut être l'entrée du bloc de mise en file d'attente , ou elle peut être connectée à la sortie d'un abandonneur algorithmique, ou à la sortie d'un programmateur.
- Chaque entrée d'un programmateur peut être connectée à la sortie 'une file d'attente, à la sortie d'un abandonneur algorithmique, ou à la sortie d'un autre programmateur.
- L'entrée d'un abandonneur algorithmique peut être le premier élément d'une étape de mise en file d'attente, le résultat d'un autre abandonneur algorithmique, ou elle peut être connectée à la sortie d'une file d'attente (pour indiquer l'abandon de tête).
- La sortie d'un bloc de mise en file d'attente peut être la sortie d'une file d'attente, un abandonneur algorithmique, ou un programmateur.

Noter, en particulier, que les programmateurs peuvent fonctionner en série de telle sorte qu'un paquet en tête d'une file d'attente qui nourrit les programmateurs enchaînés n'est servi qu'après que tous les critères de programmation sont satisfaits. Par exemple, une file d'attente qui porte un flux de trafic EF peut être servie en premier par un programmateur sans conservation du travail pour formater le flux à un débit maximum, puis par un programmateur conservant le travail pour mélanger le flux EF avec d'autres flux de trafic. Autrement, il peut y avoir une file d'attente et/ou un abandonneur entre les deux programmateurs.
Noter aussi que certains scénarios non significatifs (par exemple, une file d'attente précédant un abandonneur algorithmique, alimentant directement une autre file d'attente) sont interdits.

8.2 Exemple de TCB

Une SLS est présumée avoir été négociée entre le consommateur et le fournisseur qui spécifie le traitement du trafic du consommateur, comme défini par une TCS, par le réseau du fournisseur. L'accord peut être de la forme suivante :

DSCP	PHB	Profil	Traitement
1001	EF	Profil4	Éliminer quand non conforme
1100	AF11	Profil5	Formater au profil, couper la queue quand c'est plein
1101	AF21	Profil3	Re marquer non conforme à DSCP 001000, couper la queue quand c'est plein
autre	BE	aucun	Applique l'abandon de style RED.

Cette SLS spécifie que le consommateur peut soumettre des paquets marqués avec le DSCP 001001 qui va obtenir le traitement EF tant qu'ils restent conformes au Profil4, qui seront éliminés si ils excèdent ce profil. Les paquets éliminés sont comptés dans cet exemple, peut-être pour être utilisés par le service commercial du fournisseur pour convaincre le consommateur d'acheter une plus grande SLS. Les paquets marqués du DSCP 001100 seront formatés au Profil5 avant leur transmission. Les paquets marqués au DSCP 001101 seront mesurés au Profil3 avec les paquets non conformes "dégradés" en étant re-marqués avec un DSCP de 001000. Il est implicite dans cet accord que les paquets conformes reçoivent le PHB indiqué à l'origine par le champ DSCP des paquets.

Les Figures 6 et 7 illustrent un TCB qui peut être utilisé pour traiter cette SLS à une interface d'entrée à la limite consommateur/fournisseur.

L'étape classement de cet exemple consiste en un seul classeur BA. Le classeur BA est utilisé pour séparer le trafic sur la base du niveau de service Diffserv demandé par le consommateur (comme indiqué par le DSCP dans l'en-tête IP de chaque paquet soumis). On illustre trois valeurs de filtre DSCP : A, B, et C. Le 'X' dans le classeur BA est un filtre générique qui correspond à tout paquet qui ne correspond pas par ailleurs.

Le chemin pour le DSCP 001100 procède directement de l'abandonneur1 tandis que les chemins pour les DSCP 001001 et 001101 incluent une étape de mesure. Tous les autres trafics sont passés directement à l'abandonneur3. Il y a un mesureur séparé pour chaque ensemble de paquets correspondant aux résultats de classeur A et C. Chaque mesureur utilise un profil spécifique, comme spécifié dans la TCS, pour le niveau de service Diffserv correspondant. Les mesureurs dans cet exemple indiquent chacun un des deux niveaux de conformité : conforme ou non conforme.

Suivant l'étape de mesure, se trouve une étape d'action dans certaines des branches. Les paquets soumis avec le DSCP 001001 (sortie du classeur A) qui sont réputés non conformes par le Mesureur1 sont comptés et éliminés tandis que les paquets qui sont conformes sont passés à la Queue1. Les paquets soumis avec le DSCP 001101 (sortie du classeur C) sont réputés non conformes par le Mesureur2 ; ils sont marqués à nouveau et ensuite les paquets conformes et non conformes sont multiplexés ensemble avant d'être passés à l'Abandonneur2/Queue3.

Les étapes Abandon algorithmique, Mise en file d'attente, et Programmation sont réalisées comme suit, illustrées dans la Figure 7. Noter que la figure ne montre aucun des liens de contrôle implicites entre les éléments qui permettent, par exemple, qu'un abandonneur algorithmique sente l'état actuel d'une file d'attente réussie.

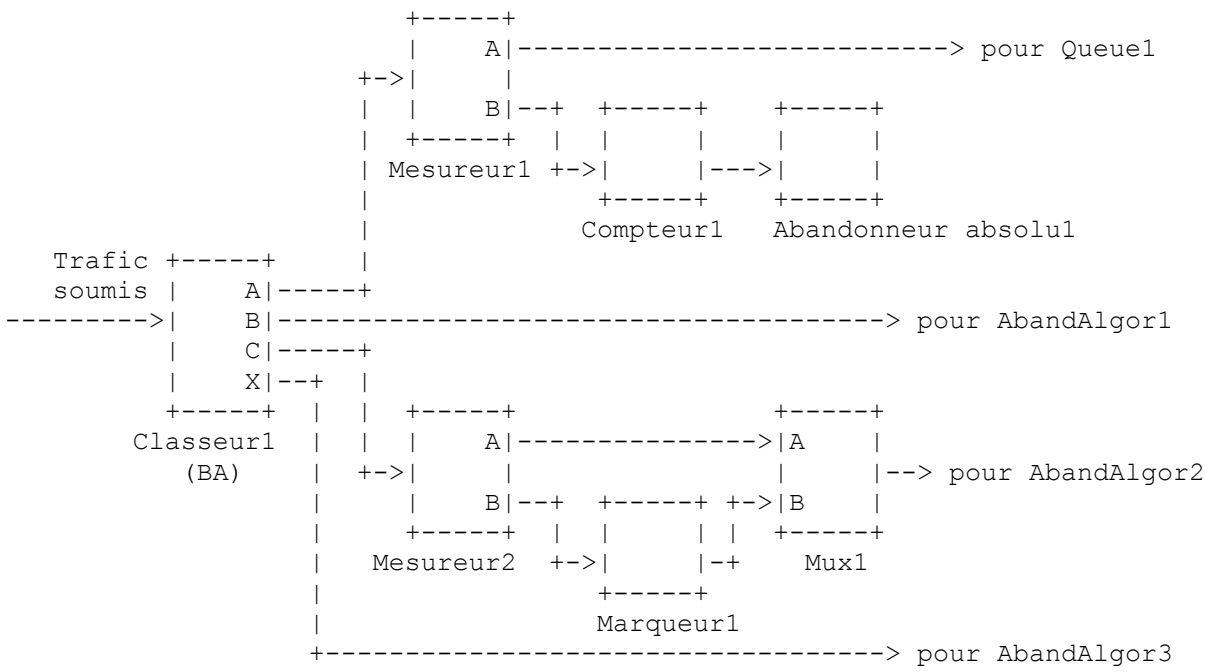

Figure 6 : Exemple de bloc de conditionnement de trafic (Partie 1)

Les paquets conformes au DSCP 001001 provenant de Mesureur1 sont passés directement à Queue1 : il n'y a pas de moyen, avec la configuration du programmateur suivant pour correspondre à la mesure, que ces paquets dépassent la profondeur de Queue1, de sorte qu'il n'y a pas d'exigence d'abandon à ce point. Les paquets marqués du DSCP 001100 doivent être passés à travers un abandonneur de queue, AbandAlgor1, qui sert à limiter la profondeur de la file d'attente suivante, Queue2 : les paquets qui arrivent à une file d'attente pleine seront éliminés. C'est probablement un cas d'erreur : le consommateur ne colle visiblement pas au profil objet de l'accord. De même, tous les paquets provenant du flux DSCP 001101 d'origine (dont certains peuvent avoir été marqués à nouveau à cette étape) sont passés à AbandAlgor2 et Queue3. Les paquets marqués pour tous les autres DSCP sont passés à AbandAlgor3 qui est un abandonneur algorithmique de type RED : sur la base des retours de la profondeur actuelle de Queue4, cet abandonneur est supposé éliminer assez de paquets de son flux d'entrée pour garder la profondeur de file d'attente sous contrôle.

Ces quatre éléments de file d'attente sont alors desservis par un élément programmeur Programmeur1 : cela doit être configuré pour donner à chacune de ses entrées une priorité appropriée et/ou une part de bande passante. Les entrées A et C reçoivent des garanties de bande passante, comme approprié pour les profils contractés. L'entrée B reçoit une limite à la bande passante qu'elle peut utiliser (c'est-à-dire, une discipline non conservatrice du travail) afin de réaliser le formatage désiré de ce flux. L'entrée D ne reçoit ni limite ni garantie mais une priorité inférieure aux autres files d'attente, appropriée pour son statut de flux au mieux. Le trafic sort alors du programmateur dans un seul flux ordonné.

Les interconnexions des éléments du TCB illustrés aux Figures 6 et 7 peuvent être représentés textuellement comme suit :

```
TCB1 :
  Classeur1 :
    FiltreA :       Mesureur1
    FiltreB :       Abandonneur1
    FiltreC :       Mesureur2
    Défaut :        Abandonneur3
```

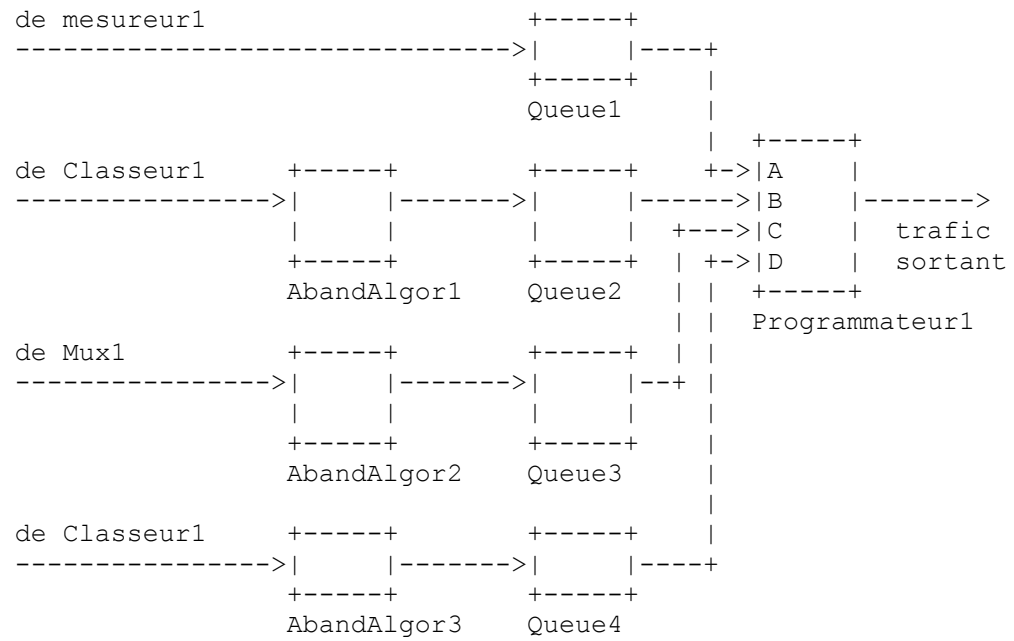

Figure 7 : Exemple de bloc de conditionnement de trafic (Partie 2)

```
  Mesureur1 :
    Type :                DébitMoyen
    Profil :              Profil4
    SortieConforme :      Queue1
    SortieNonConforme : Compteur1

  Compteur1 :
    Sortie :        AbandonneurAbsolu1

  Mesureur2 :
    Type :                DébitMoyen
    Profil :              Profil3
    SortieConforme :      Mux1.EntréeA
```

```
   SortieNonConforme : Marqueur1

Marqueur1 :
   Type :            MarqueurDSCP
   Marque :          001000
   Sortie :          Mux1.EntréeB

Mux1 :
   Sortie :          Abandonneur2

AbandAlgor1 :
   Type :            AbandonneurAlgorithmique
   Discipline :      Abandon sur seuil
   Déclencheur :     Queue2.Profondeur > 10koctets
   Sortie :          Queue2

AbandAlgor2 :
   Type :            AbandonneurAlgorithmique
   Discipline :      Abandonneur sur seuil
   Déclencheur :     Queue3.Profondeur > 20koctet
   Sortie :          Queue3

AbandAlgor3 :
   Type :            AbandonneurAlgorithmique
   Discipline :      RED93
   Déclencheur :     Interne
   Sortie :          Queue3
   SeuilMin :        Queue3.Profondeur > 20 koctet
   SeuilMax :        Queue3.Profondeur > 40 koctet
   <autres paramètres RED aussi>

Queue1 :
   Type :            FIFO
   Sortie :          Programmateur1.EntréeA

Queue2 :
   Type :            FIFO
   Sortie :          Programmateur1.EntréeB

Queue3 :
   Type :            FIFO
   Sortie:           Programmateur1.EntréeC

Queue4 :
   Type :            FIFO
   Sortie :          Programmateur1.EntréeD

Programmateur1 :
   Type :            Programmateur4Entrée
EntréeA :
   ProfilDébitMax :    aucun
   ProfilDébitMin :    Profil4
   Priorité :          20
EntréeB :
   ProfilDébitMax :    Profil5
   ProfilDébitMin :    aucun
   Priorité :          40
EntréeC :
   ProfilDébitMax :    none
   ProfilDébitMin :    Profil3
   Priorité :          20
EntréeD :
   ProfilDébitMax :    aucun
```

ProfilDébitMin : aucun
Priorité : 10

8.3 Exemple de TCB pour prendre en charge plusieurs consommateurs

Le TCB décrit ci-dessus peut être installé sur une interface d'entrée pour mettre en œuvre une TCS de fournisseur/consommateur si l'interface est dédiée au consommateur. Cependant, si une seule interface est partagée entre plusieurs consommateurs, alors le TCB ci-dessus ne va pas suffire, car il ne différencie pas le trafic provenant des différents consommateurs. Son étape de classement utilise seulement des classeurs BA.

La configuration est directement modifiée pour prendre en charge le cas de plusieurs consommateurs par interface, comme suit. D'abord, un TCB est défini pour chaque consommateur pour refléter le TCS avec ce consommateur : TCB1, défini ci-dessus est le TCB pour le consommateur 1. Des éléments similaires sont créés pour TCB2 et pour TCB3 qui reflètent les accords avec respectivement les consommateurs 2 et 3. Ces 3 TCB peuvent contenir ou non des éléments et paramètres similaires.

Finalement, un classeur est ajouté au devant pour séparer le trafic provenant des trois différents consommateurs. Cela forme un nouveau TCB, TCB4, qui est illustré à la Figure 8.

Une représentation de ce TCB multi consommateurs pourrait être :

TCB4 :
Classeur4 :
Filtre1 : pour TCB1
Filtre2 : pour TCB2
Filtre3 : pour TCB3
Pas de correspondance : AbandonneurAbsolu4

AbandonneurAbsolu4 :
Type : AbandonneurAbsolu
TCB1 : (comme défini ci-dessus)
TCB2 : (similaire à TCB1, peut-être avec des éléments ou paramètres numériques différents)
TCB3 : (similaire à TCB1, peut-être avec des éléments ou paramètres numériques différents)

et les filtres, sur la base de l'adresse MAC de source de chaque consommateur, pourraient être définis comme suit :

Filtre1 :

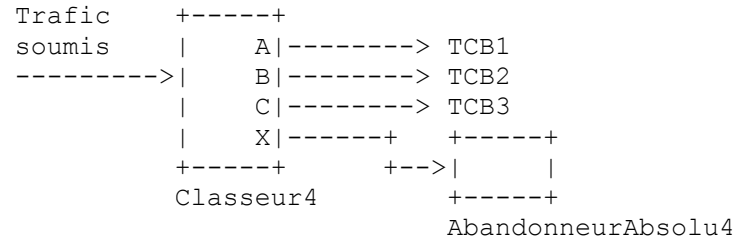

Figure 8 : Exemple de TCB multi consommateurs

Type : MacAddress
ValeurSource : 01-02-03-04-05-06 (Adresse MAC de source du consommateur 1)
GabaritSrc : FF-FF-FF-FF-FF-FF
ValeurDest : 00-00-00-00-00-00
GabaritDest 00-00-00-00-00-00

Filtre2 : (similaire à Filtre1 mais avec l'adresse MAC de source du consommateur 2 comme ValeurSource)

Filtre3 : (similaire à Filtre1 mais avec l'adresse MAC de source du consommateur 3 comme ValeurSource)

Dans cet exemple, le Classeur4 sépare le trafic soumis des différents consommateurs sur la base de l'adresse MAC de source dans les paquets soumis. Ces paquets avec des adresses MAC de source reconnues sont passées au TCB qui met en œuvre la

TCS avec le consommateur correspondant. Ces paquets avec des adresses MAC de source non reconnues sont passés à un abandonneur.

TCB4 a une étape Classeur et une étape d'élément d'action qui effectuent l'élimination de tout le trafic non correspondant.

8.4 TCB prenant en charge des services fondés sur le micro flux

Le TCB illustré ci-dessus décrit une configuration qui pourrait convenir pour mettre en application une SLS à l'entrée d'un routeur. Il suppose que le consommateur marque son propre trafic pour le niveau de service approprié. Il limite ensuit le débit du trafic agrégé soumis à chaque niveau de service, protégeant les ressources du réseau Diffserv. Il ne fournit aucun isolement entre les microflux individuels du consommateur.

Un exemple plus complexe pourrait être une configuration de TCB qui offre une fonctionnalité supplémentaire au consommateur. Il reconnaît les microflux individuels du consommateur et marque chacun indépendamment. Il isole aussi les microflux individuels du consommateur les uns des autres afin d'empêcher qu'un seul microflux ne saisisse une part non équitable des ressources disponibles au consommateur à un certain niveau de service. C'est ce qui est illustré à la Figure 9.

Supposons que le consommateur a une SLS qui spécifie deux niveaux de service, à identifier chez le fournisseur par le DSCP A et le DSCP B. Le trafic est d'abord dirigé sur un classeur MF qui classe le trafic sur la base de divers critères de classement, à une granularité suffisante pour identifier les microflux individuels de consommateur. Chaque microflux peut alors être marqué pour un DSCP spécifique. Les éléments de mesure limitent la contribution de chaque microflux du consommateur au niveau de service pour lequel il a été marqué. Les paquets qui excèdent la limite disponible pour le microflux sont abandonnés.

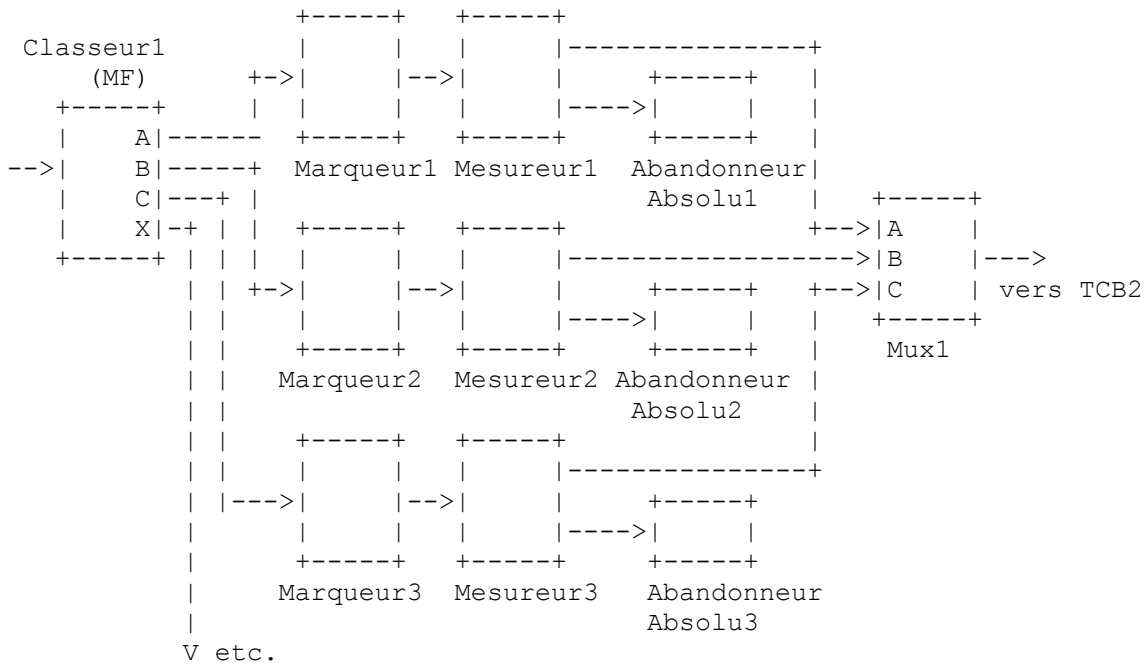

Figure 9 : Exemple de TCB de marquage et isolement de trafic

Ce TCB pourrait être spécifié formellement comme suit :

```
TCB1 :
  Classeur1 :   (MF)
    FilterA :      Marqueur1
    FiltreB :      Marqueur2
    FiltreC :      Marqueur3
    etc.

  Marqueur1 :
    Sortie :         Mesureur1

  Marqueur2 :
    Sortie :         Mesureur2
```

Marqueur3 :
Sortie : Mesureur3

Mesureur1 :
SortieConforme : Mux1.EntréeA
SortieNonConforme : AbandonneurAbsolu1

Mesureur2 :
SortieConforme : Mux1.EntréeB
SortieNonConforme : AbandonneurAbsolu2

Mesureur3 :
SortieConforme : Mux1.EntréeC
SortieNonConforme : AbandonneurAbsolu3
etc.

Mux1:
Sortie : vers TCB2

Noter que les déclarations d'élément de trafic détaillées ne sont pas montrées ici. Le trafic est soit éliminé par TCB1, soit émerge marqué pour un des deux DSCP. Ce trafic est alors passé à TCB2 qui est illustré à la Figure 10.

TCB2 pourrait alors être spécifié comme suit :

Classeur2 : (BA)
FiltreA : Mesureur5
FiltreB : Mesureur6

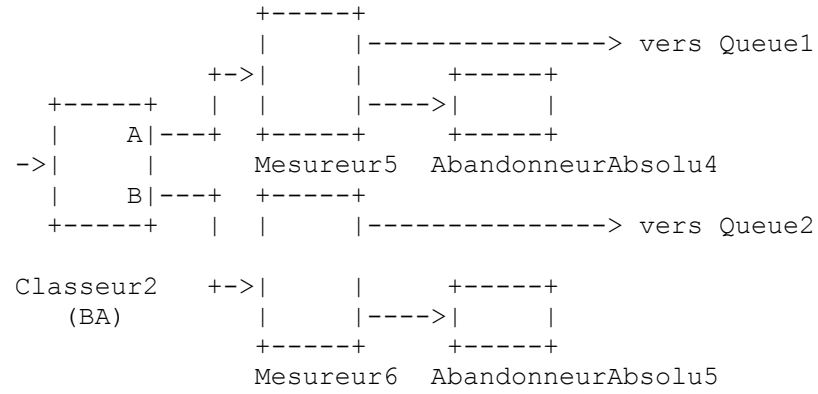

Figure 10 : Exemple supplémentaire TCB2

Mesureur5 :
SortieConforme : Queue1
SortieNonConforme : AbandonneurAbsolu4

Mesureur6 :
SortieConforme : Queue2
SortieNonConforme : AbandonneurAbsolu5

8.5 TCB en cascade

Rien dans le présent modèle n'empêche des scénarios plus complexes dans lesquels un microflux TCB en précède un autre (par exemple, pour des TCB qui mettent en œuvre des TCS séparées pour la source et pour un ensemble de destinations).

9. Considérations sur la sécurité

Les faiblesses de sécurité du fonctionnement d'un réseau Diffserv sont discutées dans la [RFC2475]. Le présent document décrit un modèle fonctionnel abstrait d'éléments de routeur Diffserv. Certaines attaques de déni de service comme celles résultant de la privation de ressource peuvent être atténuées par une configuration appropriée de ces éléments de routeur ; par exemple, en limitant un certain flux de trafic ou en authentifiant le trafic marqué pour une qualité de service plus élevée.

Il peut y avoir des scénarios de vol de service où un hôte malveillant va exploiter un régulateur de baquet de jetons lâche pour obtenir une QS légèrement meilleure que celle déclarée dans le TCS.

10. Remerciements

Les concepts, la terminologie, et le texte ont été librement empruntés à la [RFC3198], ainsi qu'à d'autres travaux de l'IETF sur les MIB et la gestion de politique. On souhaite remercier les auteurs de ces documents : Fred Baker, Michael Fine, Keith McCloghrie, John Seligson, Kwok Chan, Scott Hahn, et Andrea Westerinen de leurs contributions.

Le présent document a bénéficié des commentaires et suggestions de plusieurs participants au groupe de travail Diffserv, en particulier Shahram Davari, John Strassner, et Walter Weiss. Le présent document n'aurait jamais pu atteindre ce niveau de consensus sans les efforts incessants des co-présidents Brian Carpenter et Kathie Nichols, que les auteurs assurent de leur gratitude.

11. Références

[FJ95] Floyd, S. et V. Jacobson, "Link Sharing et Resource Management Models for Packet Networks", IEEE/ACM Transactions on Networking, Vol. 3 n° 4, août 1995.

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

[RFC2309] B. Braden et autres, "Recommandations sur la gestion de file d'attente et l'évitement d'encombrement dans l'Internet", avril 1998.

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

[RFC2475] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang et W. Weiss, "Architecture pour services différenciés", décembre 1998. *(MàJ par* RFC3260)

[RFC2597] J. Heinanen, F. Baker, W. Weiss, J. Wroclawski, "Groupe PHB Transmission assurée", juin 1999. (*MàJ par* RFC3260)*PS*

[RFC2697] J. Heinanen, R. Guerin, "Marqueur de trafic à débit unique à trois couleurs", septembre 1999. *(Information)*

[RFC2698] J. Heinanen, R. Guerin, "Marqueur de trafic à deux débits à trois couleurs", septembre 1999. *(Information)*

[RFC2998] Y. Bernet et autres, "Cadre de fonctionnement de services intégrés sur réseaux Diffserv", novembre 2000. *(Information)*

[RFC3086] K. Nichols, B. Carpenter, "Définition des comportements de services différenciés par domaine et règles de leur spécification", avril 2001. *(Information)*

[RFC3198] A. Westerinen et autres, "Terminologie pour la gestion fondée sur la politique", novembre 2001. *(Information)*

[RFC3246] B. Davie et autres, "Comportement par bond de transmission accélérée", mars 2002. *(P.S.)*

[RFC3260] D. Grossman, "Nouvelle terminologie et précisions pour Diffserv", avril 2002. *(Information)*

[RFC3289] F. Baker, K. Chan, A. Smith, "Base de données d'informations de gestion pour l'architecture de services différenciés", mai 2002. *(P.S.)*

[RFC3644] Y. Snir et autres, "Modèle d'informations de politique de qualité de service (QS)", novembre 2003. *(P.S.)*

[VIC] McCanne, S. et Jacobson, V., "vic: A Flexible Framework for Packet Video", ACM Multimedia '95, novembre 1995, San Francisco, CA, pp. 511-522. < ftp://ftp.ee.lbl.gov/papers/vic-mm95.ps.Z >

[802.1D] ISO/CEI 15802-3 : 1998, "Technologies de l'information - Télécommunications et échanges d'informations entre systèmes – Réseaux de zone locale et métropolitaine – Spécifications communes – Partie 3 : Ponts de contrôle d'accès au support (MAC) : Révision". Révision de ISO/CEI 10038 : 1993, 802.1j-1992 et 802.6k-1992. Elle incorpore P802.11c, P802.1p et P802.12e.

Appendice A. Discussion des baquets de jetons et des baquets à fuite

Les modèles de "baquet à fuite" et/ou de "baquet de jetons" sont utilisés pour décrire le contrôle de débit dans plusieurs architectures, incluant le relais de trame, l'ATM, les services intégrés et les services différenciés. Ces deux modèles sont, par définition, des relations théoriques entre une taille de salve définie, B, un débit, R, et un intervalle de temps, t :

R = B/t

Donc, un baquet de jetons ou un baquet à fuite peuvent spécifier un débit d'informations de 1,2 Mbit/s avec une taille de salve de 1500 octets. Dans ce cas, le débit de jetons est de 1 200 000 bit/s, la salve de jetons est de 12 000 bits et l'intervalle de jetons est de10 millisecondes. La spécification dit que le trafic conforme va, dans le pire des cas, venir dans 100 salves par seconde de 1500 octets chacune et à un débit moyen n'excédant pas 1,2 Mbit/s.

A.1 Baquets à fuite

Un algorithme de baquet à fuites est principalement utilisé pour formater le trafic lorsque il quitte une interface pour le réseau (traité par les files d'attente et les programmateurs dans ce modèle). Le trafic part théoriquement d'une interface à un débit d'un bit à chaque unité de temps (dans l'exemple, un bit toute les 0,83 microsecondes) mais, en fait, il part dans des unités multi-bits (paquets) à un taux proche du débit théorique, comme mesuré sur un intervalle plus long. Dans l'exemple, il pourrait envoyer un paquet de 1500 octets toutes les 10 ms ou peut-être un paquet de 500 octets toutes les 3,3 ms. Il est aussi possible de construire des baquets à fuite multi débits dans lesquels le trafic part de l'interface à des débits variables selon l'activité ou la non activité récentes. Les mises en œuvre cherchent généralement un taux de transmission aussi constant que possible. En théorie, un flux de transmission formaté à 10 Mbit/s provenant d'une mise en œuvre algorithmique et un flux qui s'écoule à 10 Mbit/s parce que sa liaison embouteillée a été une liaison Ethernet à 10 Mbit/s devraient être indistinguables. Selon la configuration, l'approximation d'un lissage théorique peut varier en déplaçant jusqu'à une MTU d'un intervalle de jeton à un autre. Le trafic peut aussi être perturbé par d'autre trafic en compétition pour les mêmes ressources de transmission.

A.2 Baquets de jetons

Un baquet de jetons, d'un autre côté, mesure le débit d'arrivée du trafic provenant d'un autre appareil. Ce trafic peut à l'origine avoir été formaté en utilisant un formateur en baquet à fuite ou son équivalent. Le baquet de jetons détermine si le trafic se conforme (encore) à la spécification. Les baquets de jetons multi débits (par exemple, les baquets de jetons avec à la fois un débit de crête et un débit moyen, et parfois plus) sont couramment utilisés, comme ceux décrits dans les [RFC2697] et [RFC2698]. Dans ce cas, on ne s'attend pas à un lissage absolu, mais à la conformité à un ou plusieurs des débits spécifiés.

En simplifiant, un flux de données est dit conforme à un simple baquet de jetons paramétré par {R, B} si le système reçoit dans tout intervalle de temps "t", au plus une quantité de données n'excédant pas (R * t) + B.

Pour le cas d'un baquet de jetons multi débits, le flux de données est dit conforme si, pour chacun des débits, le flux se conforme au profil de baquet de jetons approprié pour le trafic de cette classe. Par exemple, le trafic reçu qui arrive pré-classé comme un des débits en "excès" (par exemple, du trafic AF12 ou AF13 pour un appareil qui met en œuvre le PHB AF1x) n'est comparé qu'au profil de baquet de jetons par "excès" pertinent.

A.3 Conséquences

Le fait que les données du protocole Internet sont organisées en paquets de longueur variable introduit une certaine incertitude dans la décision de conformité prise par un mesureur en aval qui tente de déterminer la conformité à un profil de trafic qui est théoriquement conçu pour des unités de données de longueur fixe.

Lorsque elle est utilisée comme un formateur de baquet à fuites, la définition ci-dessus interagit avec la granularité d'horloge d'une façon assez inattendue. Un baquet à fuite ne libère un paquet que lorsque tous ses bits auront été admis : il ne fonde rien sur une capacité future. Si l'horloge a une granularité très fine, de l'ordre du débit binaire ou plus rapide, ce n'est pas un problème. Mais si l'horloge est relativement lente (et des horloges à la milliseconde ou plusieurs millisecondes ne sont pas inhabituelles dans les équipements de réseau) cela peut introduire de la gigue dans le flux formaté.

Cela pose un dilemme à une mise en œuvre de mesureur de baquet de jetons. Lorsque le nombre de jetons de bande passante, b, laissé dans le baquet de jetons, est positif mais inférieur à la taille du paquet sur lequel on travaille, L, une des trois actions suivantes peut être effectuée :

(1) La taille du paquet complet peut être soustraite du baquet, le laissant négatif, en se souvenant que, lorsque de nouveaux jetons sont ensuite ajoutés au baquet, la nouvelle allocation de jetons, B, doit être ajoutée à b plutôt que de simplement régler la baquet à "plein". Cette option met potentiellement plus que la taille de salve désirée de données dans cet intervalle de baquet de jetons et un nombre inférieur correspondant dans le prochain. Elle conserve, cependant, la quantité moyenne acceptée par intervalle de baquet de jetons égale à la salve de jetons. Cette approche accepte le trafic si tout bit dans le paquet aurait été accepté, et emprunte jusqu'à une MTU de capacité d'un ou plusieurs des intervalles suivants lorsque nécessaire. Une telle mise en œuvre de mesureur de baquet de jetons est dite offrir une conformité "lâche" au baquet de jetons.

(2) Autrement, le paquet peut être rejeté et la quantité de jetons dans le baquet laissée inchangée (et peut-être qu'on peut tenter d'accepter le paquet sous un autre seuil dans un autre baquet) en se souvenant que lorsque de nouveaux jetons sont ensuite ajoutés au baquet, la nouvelle allocation de jetons, B, doit être ajoutée à b plutôt que de simplement régler la baquet à "plein". Cette potentialité met moins que la taille de salve permissible de données dans cet intervalle de baquet de jetons et une quantité correspondante en plus dans le prochain. Comme dans la première option, on garde la quantité moyenne acceptée par intervalle de baquet de jetons égale à la salve de jetons. Cette approche n'accepte du trafic que si chaque bit du paquet aurait été accepté et emprunte jusqu'à une MTU de capacité d'un ou plusieurs intervalles précédents lorsque nécessaire. Une telle mise en œuvre de mesureur de baquet de jetons est dite offrir une conformité "stricte" (ou peut-être "plus stricte") au baquet de jetons. Cette option est cohérente avec les [RFC2697] et [RFC2698] et est souvent utilisée dans les mises en œuvre d'ATM et de relais de trame.

(3) La variable TB peut être réglée à zéro pour tenir compte de la première partie du paquet et le reste de la taille du paquet peut être prise dans le prochain baquet coloré. Ceci présente, bien sûr, une autre difficulté : le même paquet ne peut pas avoir des composants à la fois conformes et non conformes dans l'architecture Diffserv et n'est donc pas vraiment approprié ici, et on ne discutera pas plus avant de cette option ici.

Malheureusement, la chose qui ne peut pas être faite est de faire adhérer exactement la spécification de la salve de jetons à des paquets de taille aléatoire : donc les baquets de jetons dans un environnement de paquet de longueur variable affichent toujours une certaine variance avec la réalité théorique. Cela a aussi été observé dans la spécification de catégorie de service ATM débit de trame garanti (GFR, *Guaranteed Frame Rate*) et le relais de trame. On peut faire un certain nombre d'observations :

- Du point de vue du fonctionnement, un mesureur de baquet de jetons est raisonnable pour le trafic qui a été formaté par un formateur à baquet lâche ou une ligne de série. Cependant, le trafic dans l'Internet est rarement formaté de cette façon : TCP n'applique pas de formatage à son trafic, mais dépend plutôt d'un comportement d'horloge ACK à plus longue portée pour l'aider à approximer un certain débit et envoyer explicitement des salves de trafic durant le démarrage lent, la retransmission, et la récupération rapide. Les mises en œuvre de vidéo sur IP comme [VIC] peuvent avoir un formateur à baquet à fuites qui leur est disponible, mais ce n'est souvent pas le cas, et elles mettent simplement en queue le résultat de leur codec pour la transmission sur l'interface appropriée. Par suite, dans chacun de ces cas, un mesureur à baquet de jetons peut rejeter du trafic à court terme (sur un seul intervalle de jeton) qu'il aurait accepté si il avait eu une perspective à plus long terme et qu'il aura besoin d'accepter pour que l'application fonctionne correctement. Pour contourner cela, l'intervalle de jeton, B/R, doit approximer ou excéder le délai d'aller-retour de la ou des sessions en question et la taille de salve, B, doit s'accommoder de la plus grande salve que l'origine pourrait envoyer.

- Le comportement d'un baquet de jetons lâche est significativement différent de celui d'une description de baquet de jetons pour ATM et pour le relais de trame.

- Un baquet de jetons lâche n'accepte pas de paquets lorsque le compte de jetons est négatif. Cela signifie que, lorsque un

gros paquet a juste emprunté des jetons sur l'avenir, même un petit paquet entrant (par exemple, un ACK/SYN TCP de 40 octets) ne sera pas accepté. Donc, si un tel baquet de jetons lâche est configuré avec une taille de salve proche de la MTU, il faut qu'une certaine discrimination à l'égard des plus petits paquets puisse avoir lieu : l'utilisation d'une plus grande taille de salve évite ce problème.

- L'inverse de la situation ci-dessus est qu'un baquet de jetons strict n'accepte parfois pas les grands paquets alors qu'un lâche le ferait. Donc, si un tel baquet de jetons strict est configuré avec une taille de salve proche de la MTU, une certaine discrimination à l'égard des plus grands paquets peut avoir lieu : utiliser une plus grande taille de salve évite ce problème.

- Dans les déploiements réels, les MTU sont souvent plus grandes que la taille de salve offerte par un fournisseur de service réseau à la couche de liaison. Si il en est ainsi, il est alors possible qu'un mesureur de baquet de jetons strict trouve que le trafic ne correspond jamais au profil spécifié : cela peut être évité en ne permettant pas d'utiliser une telle spécification. Cette situation ne peut pas se produire avec un baquet de jetons lâche car la plus petite taille de salve qui puisse être configurée est 1 bit, limitant par définition un baquet de jetons lâche a avoir une taille de salve supérieure à une MTU.

- Les deux spécifications de baquet de jetons strict, comme spécifié dans les [RFC2697] et [RFC2698], et lâche, sont sujettes à une sous évaluation persistante. Cela accumule au fil du temps de la capacité de salve, jusqu'à la taille de salve maximum. Supposons que la taille de salve maximum soit exactement la taille des paquets envoyés – qu'on pourrait appeler la mise en œuvre de baquet de jetons "la plus stricte". Dans un tel cas, lorsque un paquet a été accepté, la profondeur de jeton devient zéro et recommence à s'accumuler. Si le prochain paquet est reçu à tout moment avant un intervalle de jeton ultérieur, il ne sera pas accepté. Si le prochain paquet arrive exactement à temps, il sera accepté et la profondeur de jeton sera à nouveau réglée à zéro. Si il arrive plus tard, cependant, l'accumulation de jetons aura été arrêtée parce qu'elle est couverte par la taille maximum de salve : durant l'intervalle entre le baquet qui devient plein et l'arrivée réelle du paquet, aucun nouveau jeton n'est ajouté. Par suite, la gigue qui s'accumule à travers les multiples bonds dans le réseau conspire contre l'algorithme de réduction du taux d'acceptation réel. Donc il y a du sens généralement à régler la taille maximum de baquet de jetons à un niveau un peu supérieur à celui de la MTU afin d'absorber une partie de la gigue et permettre un taux d'acceptation pratique plus en ligne avec le taux théorique désiré.

A.4 Définition mathématique d'une stricte conformité au baquet de jetons

Le comportement de conformité de baquet de jetons strict défini dans les [RFC2697] et [RFC2698] n'est obligatoire pour la conformité avec aucun des standard Diffserv actuels, mais on donne une définition mathématique du fonctionnement d'un baquet de jetons à deux paramètres qui est cohérente avec ces documents et qui peut aussi être utilisée pour définir un profil de formatage.

Définir un baquet de jetons avec une taille de baquet B, un taux d'accumulation de jeton R et une occupation instantanée de jeton b(t). Supposons que b(0) = B. Alors après un intervalle arbitraire sans arrivée de paquet, b(t) ne va pas changer car le baquet est déjà plein de jetons.

Supposons qu'un paquet de taille L arrive à l'instant t'. L'occupation du baquet est toujours B. Puis, tant que $L \leq B$, le paquet se conforme au mesureur, et ensuite

$$b(t') = B - L.$$

Supposons maintenant qu'une différence d'intervalle delta_t = t - t' s'écoule avant qu'arrive le prochain paquet, de taille $L' \leq B$. Juste avant cela, à l'instant t-, le baquet a accumulé delta_t*R jetons dans l'intervalle, jusqu'à un maximum de B jetons de sorte que :

$$b(t-) = \min\{ B, b(t') + delta\_t*R \}$$

Pour un baquet de jetons strict, l'essai de conformité est comme suit :

```
si (b(t-) - L' >= 0) {                    /* le paquet est conforme */
      b(t) = b(t-) - L' ;
   }
   autrement {                            /* le paquet n'est pas conforme */
      b(t) = b(t-) ;
   }
```

Cette fonction peut aussi être utilisée pour définir un profil de formatage. Si un paquet de taille L arrive à l'instant t, il sera éligible à la transmission à l'instant te donné comme suit (on suppose toujours $L \leq B$) :

$$te = \max\{ t, t'' \}$$

où t" = (L - b(t') + t'*R) / R et b(t") = L, l'instant où L crédits se sont accumulés dans le baquet, et quand le paquet serait conforme si le baquet de jetons était un mesureur.

te != t" seulement si t > t".

Une définition mathématique selon ces lignes pour la conformité d'un baquet de jetons lâche est laissée à titre d'exercice pour le lecteur.

Adresse des auteurs

Yoram Bernet
Microsoft
One Microsoft Way
Redmond, WA 98052
téléphone : +1 425 936 9568
mél : ybernet@msn.com

Steven Blake
Ericsson
920 Main Campus Drive, Suite 500
Raleigh, NC 27606
téléphone : +1 919 472 9913
mél : steven.blake@ericsson.com

Daniel Grossman
Motorola Inc.
20 Cabot Blvd.
Mansfield, MA 02048
téléphone : +1 508 261 5312
mél : dan@dma.isg.mot.com

Andrew Smith (editor)
Harbour Networks
Jiuling Building
21 North Xisanhuan Ave.
Beijing, 100089
PRC
mél : ah_smith@acm.org

Déclaration complète de droits de reproduction

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