14

La qualité de service sur IP

Par défaut, un réseau IP se contente d'acheminer les paquets au mieux de ses possibilités, et sans distinction. Tant que la bande passante (c'est-à-dire le débit) est suffisante, il n'y a pas de problème. Mais, en cas de saturation, les routeurs sont obligés de rejeter des paquets, invitant tous les émetteurs à réduire leur flux. En conséquence, l'utilisateur constate une dégradation des performances du réseau.

La notion de qualité de service (QoS, *Quality of Service*) introduit la possibilité de partager le plus équitablement possible une ressource devenant de plus en plus rare, car partagée par un grand nombre de flux applicatifs qui peuvent interférer les uns avec les autres. Elle introduit également la possibilité de déterminer différents niveaux de service en fonction de la nature de ce flux (une visioconférence, un transfert de fichier, etc.).

Au chapitre 10, vous avez sans doute remarqué que la gestion de la qualité de service est déjà prise en compte par des protocoles de niveau 2, tels qu'ATM et Frame Relay. Alors pourquoi gérer la QoS sur IP ? Parce que, lorsqu'une application génère des flux sur un réseau Ethernet, qui traversent ensuite un réseau ATM ou Frame Relay pour arriver sur un autre réseau local, le seul dénominateur commun est IP.

Dans ce chapitre, vous apprendrez ainsi :

- comment améliorer les performances de votre réseau ;
- les différents moyens permettant de gérer la qualité de service ;
- quelle politique de qualité de service choisir ;
- comment configurer votre réseau pour gérer cette qualité de service.

Améliorer les performances du réseau

Notre réseau est de plus en plus sollicité par de nouvelles applications aux besoins très divers : connexions Telnet pour se connecter à une machine Unix, transferts de fichiers, bases de données en mode client-serveur et, maintenant, flux audio et vidéo.

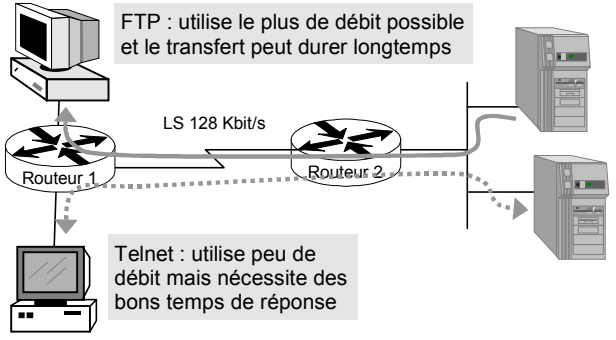
La charge du réseau augmente, et les flux générés se perturbent mutuellement : quel utilisateur n'a pas expérimenté des temps de réponse très longs alors que son voisin a lancé le transfert d'un gros fichier ?

Affecter des priorités sur les files d'attente

Le moyen le plus rudimentaire d'améliorer la qualité du service rendu par votre réseau est de définir manuellement, et sur tous vos routeurs, des priorités. Cette approche consiste surtout à améliorer les performances du réseau en favorisant des applications au détriment d'autres. La qualité est améliorée mais non garantie.

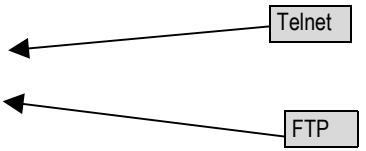
Le cas le plus couramment rencontré est celui d'une liaison WAN qui doit véhiculer des connexions Telnet (de type conversationnel) et des transferts de fichiers FTP. Les temps de réponse Telnet se dégradent dès qu'un transfert FTP est lancé. Aucune qualité de service n'étant gérée, la bande passante du réseau est, en effet, accaparée par celui qui en consomme le plus, cela bien sûr au détriment des autres.

Afin d'éviter que les applications Telnet ne soient gênées par un transfert de fichiers, le moyen le plus simple est de définir des priorités sur la base des ports TCP qui identifient les applications (23 pour Telnet et 20 pour le canal de données FTP).



En classant les paquets Telnet en priorité haute, les paquets IP comportant le port TCP 23 seront placés dans la file d'attente à priorité haute, tandis que les paquets FTP seront stockés dans la file d'attente à priorité basse. Nos routeurs, qui sont de marque Cisco, gèrent ainsi quatre files d'attente correspondant à quatre priorités : haute, moyenne, normale et basse (*high*, *medium*, *normal*, *low*).

Il existe un jeu de files d'attente par interface réseau. La priorité doit donc être activée sur les interfaces série qui émettent les flux FTP perturbateurs. Étant donné que le transfert de fichiers peut avoir lieu dans les deux sens, la même configuration doit être appliquée aux deux routeurs :




```
priority-list 1 protocol ip high tcp 23
priority-list 1 protocol ip low tcp 20
priority-list 1 protocol ip low tcp 21
int s 0
priority-group 1
```

En positionnant une priorité basse pour les flux FTP, les transferts de fichiers prendront plus de temps mais ne perturberont pas les sessions Telnet. Lorsque plusieurs paquets sont en file d'attente, le routeur enverra de préférence davantage de paquets Telnet que de paquets FTP.

Afin d'offrir le même comportement sur l'ensemble du réseau, il faut configurer de manière identique tous les routeurs. De même, les interfaces Ethernet pourraient être traitées de manière identique si les réseaux locaux étaient chargés. En fonctionnement normal, le débit de 10 Mbit/s suffit, en effet, à absorber les flux FTP et Telnet. Ce n'est qu'en cas de charge que la perturbation se manifeste et que l'activation des priorités permet de conserver le même niveau de service que celui obtenu en fonctionnement non chargé.

Il est intéressant de spécifier plus en détail un flux en utilisant une access-list :

```
priority-list 1 protocol ip low list 10
access-list 10 permit 10.0.0.1 0.255.255.255
```

Par défaut, tous les autres paquets seront traités de la manière suivante :

```
priority-list 1 default medium
```

Il est également possible de contrôler indirectement le débit en définissant le nombre maximal de paquets pouvant être en attente :

```
priority-list 1 queue-limit 20 40 60 80
```

Les quatre chiffres indiquent le nombre maximal de paquets pouvant être stockés dans les files d'attente haute, moyenne, normale et basse (les valeurs indiquées sont celles par défaut). Si la file d'attente est pleine, les paquets en excès sont rejetés, et un message ICMP *source-quench* est envoyé à l'émetteur pour lui indiquer de ralentir le flux.

Cette stratégie a des limites, car elle est à double tranchant lorsque le flux de paquets est important : si la file d'attente est de trop grandes dimensions, les paquets s'accumulent, ce qui a pour conséquence d'augmenter le temps de réponse global. Si, en revanche, elle est de trop

petite taille, des paquets peuvent être perdus, notamment entre deux réseaux de débits différents (depuis un réseau Ethernet à 10 Mbit/s vers un lien série, par exemple) : le routeur rejette tout paquet entrant tant que ses files d'attente sont saturées.

Dans notre exemple, en abaissant ainsi à 60 le nombre de paquets dans la file d'attente basse, le débit maximal du flux FTP sera encore abaissé.

Agir sur les files d'attente

L'étape suivante vers une gestion de la qualité de service consiste à agir sur le comportement des files d'attente des routeurs.

LE RÔLE DES FILES D'ATTENTE

Lorsqu'il arrive plus de paquets qu'il n'en sort du routeur, celui-ci les garde en mémoire en attendant que les plus anciens soient envoyés. Si d'autres paquets continuent d'arriver, la file d'attente se sature. Le débordement de la file d'attente se traduit par le rejet des paquets continuant d'arriver (le routeur les ignore).

Un algorithme de traitement des files a donc deux rôles essentiels :

- **traiter** en priorité tel ou tel paquet en cas de **congestion** ;
- **rejeter** en priorité tel ou tel paquet en cas de **saturation** de la file d'attente.

La priorité de traitement d'un paquet dépend de paramètres de qualité de service qui peuvent être prédéfinis dans l'algorithme, soit définis statiquement dans le routeur, soit définis dynamiquement par l'application.

Les paquets rejetés signifient pour TCP qu'il doit réduire sa fenêtre d'émission et donc son flux. Si le même paquet est rejeté plusieurs fois, l'utilisateur attend, et si l'attente se prolonge, la session risque d'être interrompue à l'expiration d'un timeout. Le choix des paquets à rejeter dépend de l'**algorithme** choisi pour traiter la file d'attente.

L'algorithme FIFO – Un fonctionnement simple

Le moyen le plus simple est de gérer les files d'attente sur le mode FIFO (*First In, First Out*) : le routeur traite les paquets au fil de l'eau dans leur ordre d'arrivée, au mieux de ses capacités. C'est le principe du *best effort*, comportement par défaut des routeurs. Aucun paramétrage n'est possible, si ce n'est de définir des priorités ce qui a pour effet de créer une file d'attente par niveau de priorité.

Gérer les congestions

Une méthode plus efficace est de traiter les files d'attente à l'aide de l'algorithme WFQ (*Weighted Fair Queueing*). Celui-ci identifie dynamiquement les flux et veille à ce que les applications générant peu de trafic ne soient pas perturbées par celles générant beaucoup de données.

Le principe de l'algorithme étant figé, la possibilité de paramétrage est donc limitée au seuil de rejet et au nombre de files d'attente :

```
int s 0
bandwidth 512
fair-queue 64 256 0  (valeurs par défaut)
```

Dans l'ordre, les paramètres indiquent :

- le seuil au-delà duquel les paquets vont commencer à être rejetés (c'est-à-dire la taille de la file d'attente moins une petite marge), ici 64 paquets par file ;
- le nombre de files d'attente pouvant être crées dynamiquement pour les flux sans qualité de service (de type *best effort*), ici 256 files pour traiter simultanément 256 flux ;
- le nombre de files d'attente pouvant être réservées par RSVP (voir plus loin), dans notre cas aucune.

Les paquets qui disposent des mêmes adresses IP source et destination, des mêmes ports source et destination et du même champ TOS (voir plus loin) correspondent à un même flux. Une fois identifiés, tous les paquets du même flux sont placés dans la même file d'attente.

La commande " bandwidth ", qui indique le débit du lien réseau en Kbit/s, permet à l'algorithme de définir le nombre nécessaire de tampons d'émission associés à l'interface (généralement quelques-uns).

Le mot " *weighted* " dans WFQ indique que l'algorithme prend en considération la priorité indiquée dans le champ " *IP precedence* " du paquet (voir le paragraphe suivant à ce sujet).

Prévenir les congestions

Alors que l'algorithme WFQ permet de gérer les situations de congestion, l'algorithme WRED (*Weighted Random Earle Detection*) permet de les prévenir. Dès qu'une congestion est détectée, l'algorithme rejette des paquets, ce qui contraint l'émetteur à ralentir son flux :

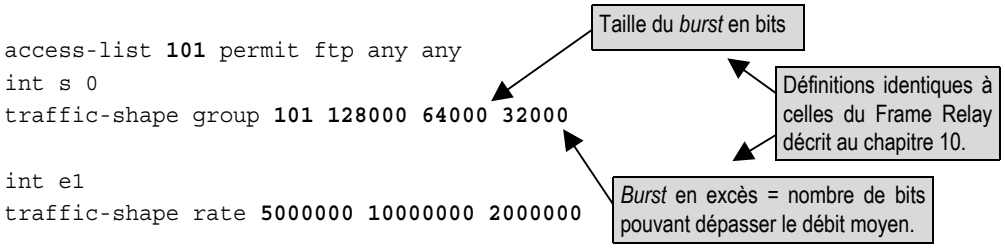
```
int s 0
bandwidth 512
random-detect
```

Tout comme WFQ, WRED permet de prendre en compte la priorité du paquet définie dans le champ " IP precedence ".

Réguler le trafic

Une autre alternative est de réguler le flux selon le principe du *trafic shaping* : le but de cet algorithme est de donner un caractère prévisible aux flux en convertissant un flux erratique en un flux a peu près constant.

Sur nos routeurs, l'algorithme peut être appliqué à l'ensemble du trafic, ou à une portion de celui-ci définie par une *access-list* :



La première commande indique que le débit moyen est limité à 128 Kbit/s, mais qu'un dépassement de 32 kilo-bits est autorisé pendant une demi-seconde (64/128). Cela signifie que le débit peut atteindre 192 Kbit/s (128 + 32/0,5) pendant une demi-seconde. Cette régulation s'applique aux flux FTP tel qu'indiqué dans l'access-list 101.

La deuxième commande est appliquée à une interface Ethernet afin de limiter le débit vers ce réseau à 5 Mbit/s en autorisant une pointe à 6 Mbit/s (5 + 2/(10/5)) pendant 2 secondes (10/5).

LA GESTION DES FILES D'ATTENTE

Comme dans la vie de tous les jours, une file d'attente se forme lorsque le flux entrant est plus important que le rythme de sortie des paquets, par exemple depuis une interface Ethernet à 10 Mbit/s vers une liaison série à 512 Kbit/s. Le routeur peut alors adopter différentes stratégies pour traiter les paquets en attente. Même si tout se passe en une fraction de seconde, le choix de l'une ou de l'autre d'entre elles peut avoir une grande influence sur le comportement général du réseau.

Le mode le plus simple est de type **FIFO** (*First In First Out*) qui consiste à traiter les paquets dans leur ordre d'arrivée. C'est celui qui consomme le moins de CPU et qui engendre le moins de latence pour les paquets. En cas de congestion, il se révèle, en revanche, moins performant car il n'y a aucune régulation du trafic.
Le principe du **WFQ** (*Weighted Fair Queueing*) repose quant à lui sur un contrôle de flux dynamique en fonction de discriminants propres à chaque protocole (le DLCI Frame Relay, les adresses IP et le port TCP, une priorité, etc.), le principe étant de privilégier le trafic à faible volume sur celui à fort volume. Pour cela, chaque flux est identifié (à partir des adresses IP source et destination et des port TCP/UDP source et destination), et son débit est mesuré.

Le principe de l'algorithme **RED** (*Random Early Detection*) consiste à prévenir les congestions. Lorsque la file d'attente commence à être saturée, des paquets correspondant à des flux sélectionnés aléatoirement sont rejetés. Les variantes **WRED** (*Weighted RED*) et **ERED** (*Enhanced RED*) permettent de sélectionner les flux, en fonction de priorités qui déterminent le rejet des paquets.

Le **Traffic shaping** est un mécanisme qui permet de réguler les flux de données, c'est-à-dire de fluidifier en sortie un trafic qui est erratique en entrée. La première implémentation, appelée **Leaky-bucket** (littéralement le seau percé), consiste à offrir un débit stable en sortie. La seconde variante, appelée **Token-bucket**, ne fluidifie pas le trafic erratique tant qu'il n'atteint pas un certain seuil (c'est-à-dire tant qu'il ne dépasse pas le nombre de jetons, un jeton équivalant à un certain nombre d'octets). Au-delà du seuil, tous les trafics sont fluidifiés. Selon les implémentations, le *Traffic shaping* utilise sa propre file d'attente ou peut opérer conjointement avec une file FIFO ou WFQ.

Quelle file d'attente choisir pour son réseau ?

Chacune des fonctions qui viennent d'être décrites présente des avantages et des inconvénients. Le problème est alors de savoir dans quelle situation et à quel endroit du réseau utiliser l'une ou l'autre de ces fonctionnalités.

Nous l'avons vu précédemment, une **file d'attente FIFO** ne permet par de gérer des situations de congestion. En conséquence, une file d'attente de ce type ne peut être utilisée qu'avec des réseaux non saturés et offrant suffisamment de débit par rapport au trafic.

L'affectation de **priorités sur les files d'attente**, qui a été l'objet de notre première démarche, consiste en une programmation arbitraire, statique, et qui ne distingue qu'individuellement les types de paquets. Lorsqu'une congestion survient, les paquets prioritaires sont traités tant qu'il y en a dans la file d'attente correspondante. L'effet de bord qui en résulte est que les autres paquets restent bloqués dans la file d'attente, ce qui peut entraîner un ralentissement conséquent, voire la coupure des sessions correspondantes à ces paquets non prioritaires. C'est néanmoins la méthode d'affectation de priorité qui utilise le moins de CPU car l'algorithme est simple. L'utilisation de cette fonction sera donc limitée à des liens à bas débit sur lesquels le trafic est bien identifié.

La **file d'attente WFQ** identifie chaque flux, et traite tous les paquets d'un même flux de la même manière. En cas de congestion, l'algorithme traite équitablement tous les flux en privilégiant ceux à faible volume, mais pas au détriment de ceux à fort volume. Bien qu'efficient, cet algorithme complexe consomme de la ressource CPU, ce qui le destine plutôt à des liens à bas et moyen débits. Du fait qu'il prend en compte les priorités indiquées dans les paquets, on activera cet algorithme au sein du réseau et non dans sa périphérie.

La **file d'attente WRED** permet de prévenir les congestions en rejetant aléatoirement des paquets. À partir d'un certain seuil, le taux de rejet de paquets augmente à mesure que la file d'attente se remplit. Cet algorithme est de ce fait particulièrement bien adapté au protocole TCP qui est prévu pour réduire sa fenêtre d'émission en cas de perte de paquet (ce qui n'est pas le cas des protocoles IPX et AppleTalk).

À l'inverse de la file d'attente pour laquelle les priorités sont fixées dans le routeur, les algorithmes WFQ et WRED prennent en compte la priorité qui est indiquée dans le paquet IP.

Enfin, le ***Traffic shaping*** est un mécanisme qui permet de lisser le trafic et de fixer un débit à chaque type de flux. Il est donc bien adapté aux routeurs situés en bordure du réseau, c'est-à-dire sur les réseaux locaux, là où utilisateurs et serveurs émettent leurs données. Ce trafic erratique peut ainsi être régulé lorsqu'il entre dans le réseau longue distance, ce qui permet ensuite aux routeurs situés au sein de ce réseau d'utiliser les files d'attente WFQ et WRED.

La file d'attente de type…	… doit être utilisée
FIFO	Lorsque le réseau n'est pas saturé
Priorité sur file d'attente	Sur des liens à bas débit
WFQ	Au sein du réseau, sur des liens à bas et moyen débits (< 2 Mbit/s)
WRED	Au sein du réseau, sur des liens à haut débit
Traffic shaping	En entrée du réseau et pour adapter le flux entre des liaisons ayant des débits différents

Gérer la qualité de service

À la base, un réseau IP tel que l'Internet ne garantit pas que tous les paquets émis seront délivrés au destinataire. Il assure simplement que les paquets effectivement remis le seront sans erreur ni duplication en routant les paquets au mieux de ses possibilités (principe du *best effort*).

C'est également le principe de la poste qui s'efforce d'acheminer le courrier dans les meilleurs délais. Mais, pour s'assurer qu'un colis arrivera à destination en temps et en heure, il faut payer un service supplémentaire. De même, pour les réseaux, il faut mettre en œuvre des moyens supplémentaires afin de garantir ce niveau de service.

Alors que les flux de données classiques (Telnet, FTP, etc.) se contentent du service de base (celui du meilleur effort), le transport des flux audio et vidéo nécessite plus que cela. Il faut en effet garantir :

- qu'une application disposera du minimum de débit réseau nécessaire à son bon fonctionnement ;
- un temps de réponse ;
- que le temps de réponse variera peu dans le temps.

Afin de gérer la qualité de service nécessaire au traitement des flux multimédias, deux modèles sont actuellement proposés :

- La **différenciation de service** (modèle appelé **DiffServ**) repose sur l'affectation de priorités et de classes de service dont les valeurs sont transportées dans les paquets IP. Le flux est formaté (classé) à l'entrée du réseau, puis la qualité de service est appliquée de la même manière dans tous les routeurs en fonction de la valeur indiquée dans les paquets.
- L'**intégration de service** (modèle appelé **IntServ**) consiste à réserver les ressources tout le long du chemin qu'emprunteront les paquets, puis à appliquer à tout le flot de paquets qui suivent la qualité de service demandée lors de la réservation.

La qualité de service selon DiffServ

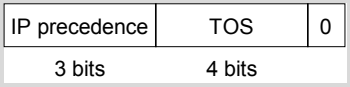
Le modèle DiffServ est une norme en cours de spécification qui ne précise pas encore les classes de service. C'est pour cela que les implémentations actuelles ne traitent que de la priorité (champ *IP Precedence*).

Le champ TOS

Dès l'origine, les concepteurs des protocoles TCP/IP ont pensé à intégrer la notion de qualité de service dans un champ du paquet IP, appelé TOS (*Type of Service*). Ce champ était destiné à transporter des informations relatives à la priorité et à la classe de service, mais il n'a jamais été réellement utilisé, jusqu'à ce que la qualité de service soit d'actualité. Mais, entre temps, les besoins ont évolué, et différents groupes de travail au sein de l'IETF ont proposé de modifier la signification de ce champ.

LE CHAMP TOS (RFC 791 ET 1349)

Le champ **TOS** (*Type of Service*) consiste en un unique octet :



Le champ **IP precedence** (bits 0 à 2) détermine une priorité allant de 0 à 7 :

0 = Routine
1 = Priority
2 = Immediate
3 = Flash
4 = Flash override
5 = Critical
6 = Internetwork control
7 = Network control

Les bits **TOS** (la RFC utilise malencontreusement le même terme TOS pour l'octet entier ainsi que pour ces 4 bits) indiquent la classe de service souhaitée :

Bit	Description	Signification des valeurs
3	minimize Delay	(0 = Normal delay ; 1 = Low Delay)
4	maximize Throughput	(0 = Normal throughput ; 1 = High throughput)
5	maximize Reliability	(0 = Normal reliability ; 1 = Reliability)
6	minimize monetary cost	(ajouté par la RFC 1349)

Ces 4 bits positionnés à 0 indiquent tout simplement le service normal (le *best effort*).
Le dernier bit est réservé à un usage futur.

Les algorithmes WFQ et WRED prennent en compte le champ *Precedence* (d'où le " W " pour *Weighted*). L'activation de ces files d'attente est donc particulièrement judicieuse au sein du réseau, car elles permettent de réguler le flux en fonction des priorités fixées à l'entrée de celui-ci.

L'exemple suivant, qui montre quelques valeurs par défaut, permet d'expliquer comment WRED interagit avec les priorités :

```
random-detect precedence 0 109 218 10
random-detect precedence 1 122 218 10
...
random-detect precedence 7 194 218 10
```

La première valeur (de 0 à 7) correspond à la priorité indiquée dans le champ *Precedence*. Les trois paramètres suivants indiquent quant à eux :

- le **seuil minimal**, en nombre de paquets dans la file d'attente, à partir duquel les paquets commenceront à être rejetés ;
- le **seuil maximal**, en nombre de paquets dans la file d'attente, à partir duquel les paquets seront rejetés au rythme indiqué dans le paramètre suivant ;
- la **fraction** de paquets rejetés lorsque le seuil maximal est atteint, par défaut un sur dix.

Afin de favoriser les paquets prioritaires, le seuil minimal est d'autant plus élevé que la priorité est grande. L'algorithme WRED calcule la taille moyenne de la file d'attente et la compare aux seuils.

Le seuil maximal est déterminé automatiquement en fonction du débit de l'interface et de la mémoire disponible, tandis que le seuil minimal correspond à une fraction du seuil maximal dépendant de la priorité (par exemple, 1/2 pour 0, 10/18 pour 1, 11/18 pour 2, etc.). Les commandes " *sh queueing* " et " *sh int random-detect* " permettent de visualiser les paramètres actifs.

Configuration des routeurs

Les routeurs sont de nos jours destinés à opérer sur le réseau WAN et donc à être enfouis au sein du réseau. En vertu de cette conception, seule la fonction de *policing* du modèle Diff-Serv est implémentée sur nos routeurs, les autres fonctions opérant en entrée.

À la différence du *Trafic shaping* qui régule un flux erratique, le *policing* veille au respect du flux selon le profil de flux qui lui est indiqué. Il peut alors **rejeter** les paquets non conformes ou les **marquer** pour une régulation ultérieure sur d'autres équipements au sein du réseau :

```
int e0
rate-limit output 5000000 32000 45000 conform-action transmit exceed-
action drop
```

Le mot clé " *output* " indique que la classification du flux est appliquée en sortie de l'interface :

- le premier paramètre indique le débit moyen autorisé, ici 5 Mbit/s ;
- le deuxième précise la taille du burst, ici 32 Ko ;
- la troisième valeur est le burst en excès, ici 45 Ko.

Si le flux est conforme à ce profil (*conform-action*), le paquet est placé dans la file d'attente tel quel. Dans le cas contraire (*exceed-action*), il est rejeté.

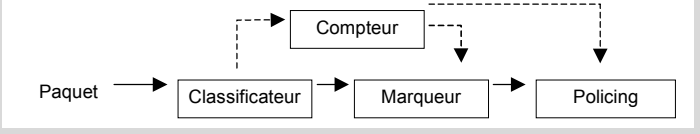
LE POINT SUR DIFFSERV (RFC 2474 ET 2475)

Le modèle de différentiation de service repose sur le transport dans les paquets IP (champ TOS) de la **priorité** et de la **classe de service**. Il se propose en plus de définir la manière d'implémenter la qualité de service.
Le champ TOS est à l'occasion renommé **DS** (*Differentiated Service*) et structuré différemment, ce qui ne va pas sans poser de problèmes de compatibilité pour les routeurs traitant ce champ sur l'ancien mode (discussion objet de la RFC 2873).

DiffServ conserve les 3 bits du champ Precedence et modifie les 3 suivants de manière à définir 64 **codepoints** répartis en trois pools :

xx0 32 codepoints pour les actions standards ;
x11 16 codepoints pour un usage expérimental et une utilisation locale ;
x01 16 codepoints pour un usage expérimental, une utilisation locale ou une extension du premier pool.

Le codepoint xxx000 (issu du pool 1) permet d'assurer la compatibilité ascendante avec le champ Precedence. Les deux derniers bits 6 et 7 ne sont actuellement pas utilisés.
Le **classificateur** affecte une priorité et une classe de service au paquet en fonction de règles paramétrables (adresses IP source et destination, port TCP/UDP, etc.).
Le **conditionneur** applique la qualité de service en fonction du champ DS. Il analyse le flux de données (fonction de comptage — *meter*) et le compare à la QoS demandée. Il peut alors redéfinir la classe de service (fonction de marquage — *marker*), réguler le trafic (fonction de régulation — *shaper*) ou encore rejeter les paquets (fonction de rejet — *dropper*) pour l'adapter à la classe de service demandée.



À l'heure actuelle, deux classes de service sont définies :

- *Assured Forwarding* (RFC 2597), pour les flux nécessitant une bande passante limitée, le trafic en excès pouvant être rejeté progressivement selon un mécanisme de priorité à 12 niveaux (4 classes × 3 priorités de rejet).
- *Expedited Forwarding* (RFC 2598), également appelé "Premium service", pour les flux requérant une bande passante garantie avec des faibles taux de perte, de gigue et de latence.

Ces classes de service réservent des codepoints et décrivent la manière d'implémenter la gestion de la QoS au sein du réseau (*per-hop behavior*).

La fonction *policing* peut également marquer le paquet, c'est-à-dire lui affecter une priorité en fonction du respect ou non du profil de flux. On peut augmenter la granulométrie de cette dernière en précisant le protocole (et éventuellement les réseaux, voire les adresses source et de destination) sur lequel va s'appliquer le marquage :

```
rate-limit output access-group 101 5000000 24000 32000 conform-action
set-prec-transmit 1 exceed-action set-prec-transmit 0

access-list 101 permit tcp any any eq www
```

Nous avons décidé ici de limiter le flux des navigateurs web (protocole HTTP), à 5 Mbit/s. La priorité est fixée à 1 (prioritaire) s'il se conforme à la QoS, et à 0 (*best effort*) dans le cas contraire.

Sur notre backbone WAN à 2 Mbit/s, connecté en 100bT au LAN de notre site central, nous avons décidé de limiter le débit à 1 Mbit/s pour les paquets de priorité 1. Pour ceux de priorité 0, il est limité à 500 Kbit/s. Le burst autorisé est de 500 Ko, et de 500 Ko en excès.

```
int e 1/0

rate-limit input access-group rate-limit 100 1000000 500000 500000 con-
form-action transmit exceed-action drop

rate-limit input access-group rate-limit 101 500000 250000 250000 con-
form-action transmit exceed-action drop

access-list rate-limit 100 1
access-list rate-limit 101 0
```

Priorité (0 à 7) écrite dans le champ *IP Precedence*.

La commande " *access-list rate-limit* " est ici utilisée pour affecter une priorité basse si le volume du flux est important, et haute dans le cas contraire. Les différents équipements au sein du réseau se chargeront d'appliquer la QoS, par exemple, au niveau de leurs files d'attente WRED ou WFQ.

Configuration des commutateurs de niveau 2

En fait, il est bien plus judicieux de marquer les paquets à la source, c'est-à-dire sur les équipements directement en contact avec les PC. Il s'agit bien sûr des commutateurs Ethernet 10/100bT.

Les postes de travail sont, en effet, généralement connectés à des commutateurs de niveau 2, alors que l'on réserve la commutation de niveau 3 pour le réseau fédérateur en raison de son coût. Cependant, ils gèrent rarement les priorités. De ce fait, les commutateurs transforment les trames Ethernet en trames 802.1q pour gérer la qualité de service et les VLAN. Lorsqu'elles sont envoyées vers un port de sortie, ces trames sont reformatées en trames Ethernet 802.3.

Étant donné que toutes les trames ne sont pas obligatoirement de type 802.1q, par défaut le commutateur ne prend donc pas en compte le champ COS (*Class of Service*), et la priorité doit donc être fixée au niveau du port.

Par défaut, le port est en mode *untrusted* : la COS est fixée à la valeur par défaut du port, c'est-à-dire à 0. Il est possible de changer la valeur par défaut associée à chaque port :

```
set qos enable
set port qos 1/4 trust untrusted
set port qos 1/4 cos 6
```

Valeur par défaut du champ COS = 0

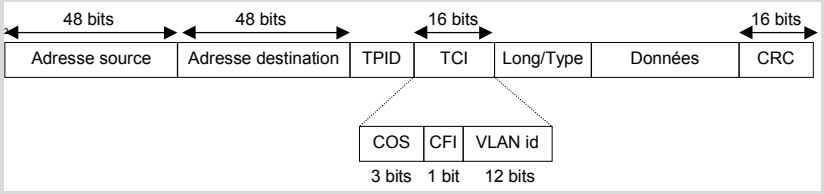
Fixe la valeur par défaut du champ COS

Toutes les trames arrivant par le port 1/4 auront un champ COS fixé d'office à 0, à moins qu'une autre valeur par défaut ne soit affectée au port.

Il n'est ainsi pas nécessaire de gérer la qualité de service au niveau du poste de travail et des serveurs. Chacun d'eux étant raccordé à un port du commutateur, celui-ci se charge de marquer les paquets. L'inconvénient est que ce marquage ne distingue pas le type de flux.

LA QUALITÉ DE SERVICE SUR ETHERNET (IEEE 802.1P)

La norme initiale 802.1q consiste à ajouter un champ à l'en-tête de la trame Ethernet initiale (802.3) à la fois pour gérer les VLAN et des **classes de service** (802.1p) .



Cette trame est véhiculée uniquement entre les commutateurs. Ces derniers enlèvent le champ 802.1q lorsqu'ils transmettent la trame à un équipement terminal (PC, serveur, etc.).

Sept classes de service (**COS**) sont ainsi définies :
0 = Best effort
1 = Tâche de fond (batch)
2 = Réservé
3 = Excellent effort
4 = Applications à contrôle de charge (streaming audio/vidéo)
5 = Vidéo
6 = Voix
7 = Network control

Certains pilotes de carte réseau permettent de gérer la priorité directement au niveau du PC.

Si la priorité est déjà fixée par un équipement terminal ou un autre commutateur (donc générant une trame 802.1q), il est possible de prendre en compte le champ COS :

```
set qos enable
set port qos 1/3 trust trust-cos
```

Le champ COS des trames 802.1q arrivant par le port 1/3 sera ainsi pris en compte.

Configuration des commutateurs de niveau 3

Équipé d'une carte de commutation de niveau 3, le commutateur peut assurer toutes les fonctions DiffServ : classification, marquage et *policing*.

Par défaut, une politique de qualité de service est associée à un port, mais il est possible de l'associer à un VLAN :

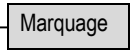
```
set qos enable
set port qos 1/1 port-based
set port qos 1/2 vlan-based
```

La configuration de nos commutateurs consiste à définir une **politique de qualité de service** contenant une règle de **marquage**, une règle de ***policing*** et une règle de **classification** (l'ordre de définition ne respecte pas celui du séquencement des opérations).

Définir une règle de marquage

Le commutateur associe aux ports une **politique par défaut** dont il est possible de modifier la règle de marquage :

```
set qos acl default-action ip dscp 0
```



Cette politique ne contient qu'une règle de marquage, qui consiste à marquer le champ DSCP (*Differentiated Service* Code *Point*) de tous les paquets avec la valeur indiquée (0 par défaut).

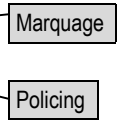
D'autres marquages sont possibles en remplaçant le mot clé *dscp* et sa valeur :

- *trust-dscp* : le champ DSCP n'est pas modifié et est accepté tel quel.
- *trust-ipprec* : le champ DSCP est modifié en prenant la valeur du champ *IP precedence*.
- *trust-cos* : le champ DSCP est modifié en fonction de la valeur contenue dans le champ COS de la trame Ethernet 802.1q.

Définir une règle de policing

Il est également possible d'associer plusieurs règles de *policing* à la politique par défaut :

```
set qos acl default-action ip dscp 0
microflow micro_de_base aggregate aggr_de_base
```



La création d'une règle de *policing* s'effectue comme suit :

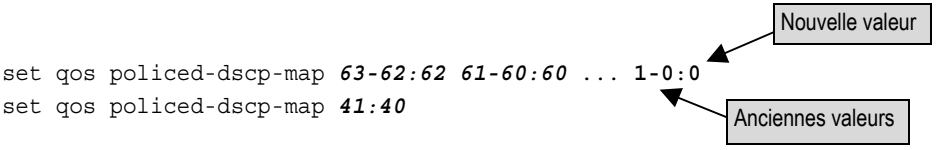
```
set qos policier microflow micro_de_base rate 1000 burst 1000 drop

set qos policier aggregate aggr_de_base rate 1000 burst 1000 policed-dscp
```

Ces deux règles limitent le débit moyen à 1 Mbit/s et le dépassement (*burst*) à 1 Mo.

Le mot clé *microflow* précise que la règle s'applique à chaque flux considéré individuellement, tandis que le mot clé *aggregate* précise qu'elle s'applique à l'ensemble des flux.

Vient ensuite le type de traitement à appliquer au paquet lorsque le flux ne correspond pas au profil : le rejeter (*drop*) ou le marquer à nouveau (*policed-dscp*).

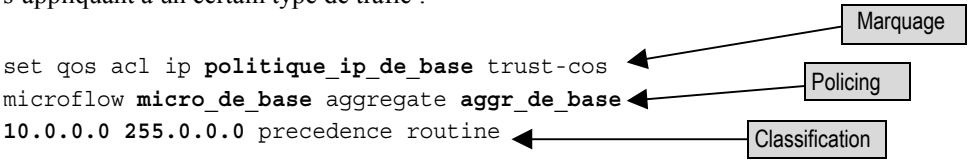
Dans ce dernier cas, la commande suivante permet de définir les correspondances, soit par plages de valeurs, soit individuellement :



Les valeurs montrées ici sont celles par défaut qui sont conservées si elles ne sont pas explicitement redéfinies.

Définir une règle de classification

La politique par défaut qui vient d'être décrite ne contient pas de règle de classification puisqu'elle s'applique à tous les paquets IP. Par contre, nous pouvons créer une règle s'appliquant à un certain type de trafic :

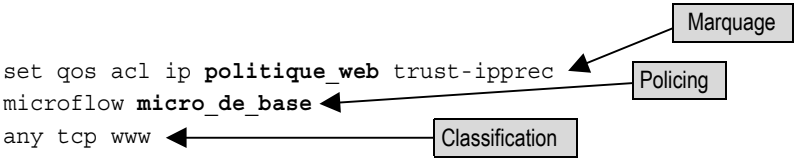


La politique ainsi définie s'applique aux paquets IP provenant du réseau 10.0.0.0 et ayant leur champ precedence à 0 (mot clé *routine*). Les paquets se conformant aux profils " micro_de_base " et " aggr_de_base " seront affectés de la priorité dérivée du champ COS de la trame 802.1q (mot clé *trust-cos*).

Les valeurs qui suivent le mot clé *precedence* sont celles décrites dans le champ TOS : *routine*, *priority*, *immediate*, *flash*, *flash-override*, *critical*, *internet* et *network*.

À la place de l'adresse IP et de son masque, on peut mettre le mot clé *any* ou le mot clé *host* suivi d'une adresse IP.

Nos commutateurs nous offrent la possibilité de décrire d'autres types de politiques s'appliquant aux trafics TCP, UDP, ICMP, IGMP, etc. Par exemple, il est ainsi possible d'affecter une règle à chaque application (qui se distingue par son port TCP ou UDP) :



Associer une politique à un port

La dernière étape consiste à associer la politique que nous venons de créer à un port du commutateur ou à un VLAN :

```
set qos acl map politique_ip_de_base 1/1
set qos acl map politique_ip_de_base VLAN 1
```

Les ports ou le trafic associés au VLAN 1 se verront appliquer la politique IP de base que nous venons de définir.

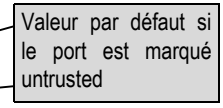
Affecter des valeurs au champ DSCP

Par défaut, tous les ports fonctionnent sur le mode *untrusted*, ce qui signifie que les priorités (CoS et DSCP) des paquets y entrant sont ignorées.

La carte de commutation de niveau 3 peut néanmoins prendre en compte les priorités des trames et paquets pour un port auquel est connecté une machine capable de générer des trames 802.1q et/ou des paquets IP DiffServ.

La manière de prendre en compte ces valeurs peut être définie soit au niveau de la règle de marquage de la politique qui est ensuite affectée au port comme nous l'avons déjà fait, soit être définie directement par port :

```
# Soit définie au niveau de la politique
set qos acl ip politique_ip_de_base trust-cos ...
set qos acl map politique_ip_de_base 1/1

# Soit définie directement au niveau du port
set qos acl ip politique_par_port dscp 0 ...
set port qos 1/4 trust untrusted
set port qos 1/3 trust trust-cos
set port qos 1/2 trust trust-ipprec
set port qos 1/1 trust trust-dscp
```



La signification des mots clés est la suivante :

- *untrusted* : le champ DSCP est fixé à 0 ou par la valeur indiquée dans la politique de qualité de service.
- *trust-cos* : le champ DSCP est fixé à la valeur du champ COS de la trame Ethernet.
- *trust-ipprec* : le champ DSCP est fixé à la valeur du champ *precedence* du paquet IP.
- *trust-dscp* : le champ DSCP du paquet est conservé tel quel.

Les correspondances COS (0 à 7) → valeur DSCP (0 à 63) et *IP precedence* (0 à 7) → DSCP par défaut conviennent, mais il est possible de les modifier comme suit :

```
set qos cos-dscp-map 0 8 16 24 32 40 48 56
set qos ipprec-dscp-map 0 8 16 24 32 40 48 56
```

Chaque valeur correspond à un DSCP compris entre 0 et 63. Les valeurs indiquées dans cet exemple sont celles par défaut.

Lorsque le paquet est finalement envoyé sur le port de sortie, le champ COS de la trame est dérivé de la valeur DSCP (conservée ou marquée à nouveau) :

```
set qos dscp-cos-map 0-7:0 8-15:1 ... 48-55:6 56-63:7
```

Là encore, les valeurs indiquées sont celles par défaut.

Configuration des postes de travail

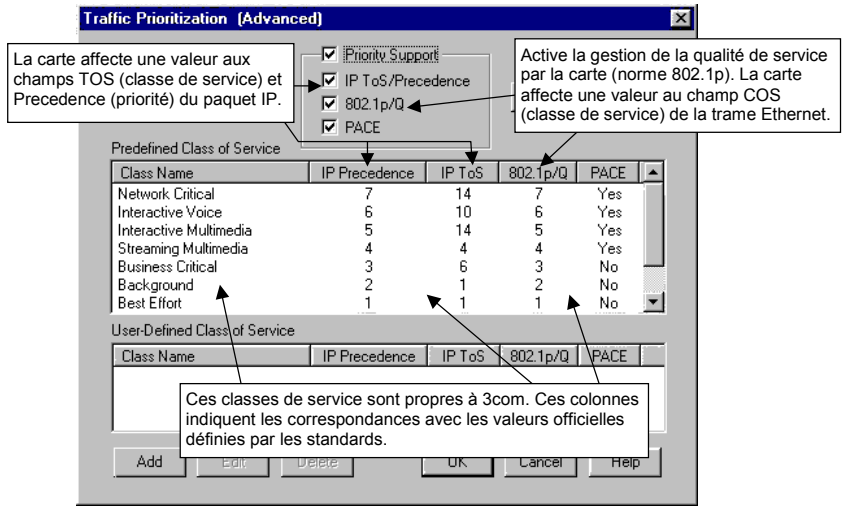
Nous venons de le voir, il est possible de gérer la qualité de service au niveau du WAN et du LAN. Cette gestion s'appuie sur des valeurs positionnées par les routeurs et les commutateurs de niveau 3 dans les paquets IP (champs TOS et Precedence), ainsi que par les commutateurs de niveau 2 dans les trames Ethernet (champ COS).

La question se pose alors de savoir où positionner ces paramètres. On peut le faire :

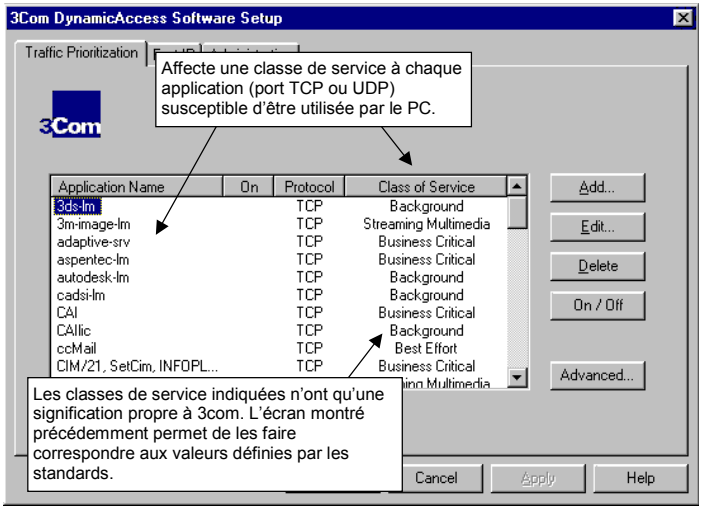
- au niveau des commutateurs de niveau 2 afin de gérer la qualité de service sur le LAN uniquement ;
- au niveau des commutateurs de niveau 2 et 3 afin de gérer la qualité de service sur le LAN et le WAN ;
- au niveau des routeurs pour gérer la qualité de service sur le WAN.

Cette dernière solution est préférable si les sites ne sont pas tous équipés de cartes de commutation de niveau 3. Par ailleurs, la bande passante est généralement suffisante sur le LAN pour permettre de se passer d'une gestion élaborée de la qualité de service.

Une dernière solution est proposée par les constructeurs : l'affectation des priorités au niveau des cartes Ethernet des postes de travail et des serveurs, c'est-à-dire à la source d'émission des trames et paquets.



Il faut donc affecter une classe de service à chaque application que sont susceptibles d'utiliser les postes de travail et les serveurs. Et il faut, bien évidemment, que toutes les cartes soient configurées de manière identique.



Cette solution est cependant difficile à mettre en œuvre pour plusieurs raisons :

- Tous les PC et serveurs doivent être équipés de cartes supportant cette fonctionnalité.
- Toutes les cartes doivent, de préférence, provenir du même constructeur, afin de simplifier l'exploitation et de garantir l'homogénéité des paramètres.

- Toutes les configurations doivent être identiques.
- Le paramétrage du driver doit pouvoir être réalisé à distance.
- Le nombre de nœuds à configurer est considérablement plus important que le nombre de commutateurs sur lesquels ce même paramétrage peut être défini.

Toutes ces contraintes rendent donc difficile la gestion de la qualité de service au niveau des postes de travail.

La qualité de service selon IntServ

À la différence de DiffServ qui repose sur le transport de la qualité de service dans les paquets IP, le modèle IntServ distribue ces informations (débit, temps de réponse, etc.) à tous les routeurs concernés par le flux, afin d'assurer la même qualité de service de bout en bout, c'est-à-dire entre les applications émettrices et destinataires. Cette cohérence nécessite deux fonctions distinctes :

- la définition des **spécificateurs** (paramètres) qui permettent de caractériser la qualité de service ;
- le protocole de **signalisation** permettant aux routeurs d'échanger les spécificateurs ainsi que des informations relatives à l'état du réseau.

La première fonction fait l'objet des RFC 2211 à 2216 qui définissent les éléments d'un réseau à intégration de service. La seconde est couverte par les RFC 2205 à 2209 qui définissent le protocole RSVP (*Resource reSerVation Protocol*).

La RFC 2210 établit la liaison entre les deux fonctions : elle traite du fonctionnement de RSVP au sein d'un réseau à intégration de service.

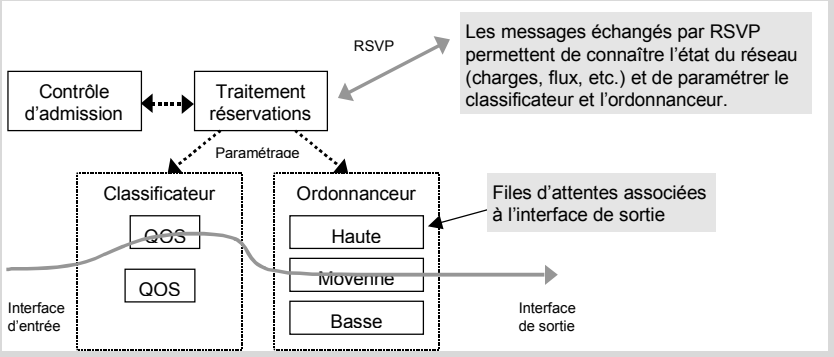
RFC	Sujet traité
2211	Classe de service par contrôle de charge
2212	Classe de service par garantie de service
2213	MIB de l'intégration de service
2214	MIB de l'intégration de service — Extension à la garantie de service
2215	Définition des paramètres permettant de calculer une QoS
2216	Définitions de la QoS pour les éléments du réseau (définit un cadre général sur lequel s'appuient les RFC 2211 et 2212)
2210	Utilisation de RSVP dans un réseau à intégration de service
2205	Spécifications fonctionnelles de RSVP
2206	MIB de RSVP
2207	Extensions pour les flux IPSEC
2208	Préconisations pour la mise en œuvre de RSVP
2009	Règles de traitement des messages

LE POINT SUR INTSERV (RFC 1633)

Il existe différentes façons d'assurer une qualité de service : affectation des priorités, choix de l'algorithme pour gérer les files d'attente, etc. Mais, si chaque routeur gère un niveau de priorité qui lui est propre ou gère des files d'attente selon des paramètres différents, la qualité de service demandée par une application sera interprétée différemment par les routeurs et ne pourra donc pas être assurée de bout en bout.

Pour que tout le monde parle le même langage, la RFC 1633 définit un cadre fonctionnel, appelé **intégration de service**. Quatre nouveaux composants doivent être intégrés aux routeurs :

- Le **contrôle d'admission** dont le rôle est de vérifier que la QoS demandée pour un nouveau flux ne remettra pas en cause les QoS accordées aux autres flux existants.
- Le **classificateur** de paquets dont le rôle est d'affecter une classe de service aux paquets entrant.
- L'**ordonnanceur** de paquets, qui gère les files d'attente pour les paquets sortant.
- Un mécanisme de **réservation** des ressources permettant de paramétrer le classificateur et l'ordonnanceur.



Une demande de réservation est traitée par le module de contrôle d'admission qui vérifie que l'application a le droit d'effectuer une telle requête et que le réseau peut offrir la QoS demandée sans remettre en cause celles déjà accordées.

L'implémentation logicielle de l'ordonnanceur dépend du routeur. La plupart d'entre eux utilisent néanmoins un algorithme standard de type WFQ (*Weighted Fair Queue*) ou WRED (*Random Early Detection*) pour la gestion des files d'attente.

De même, le classificateur peut utiliser différents éléments pour classer un paquet dans tel ou tel niveau de service : l'adresse IP et le port TCP/UDP sont des informations de base qui peuvent être utilisées.

Enfin, le mécanisme de réservation repose sur le protocole de signalisation **RSVP** (*Resource reSerVation Protocol*) qui permet de demander à tous les routeurs concernés de réserver des ressources.

La réservation des ressources

La signalisation RSVP est le protocole qui permet de négocier et de mettre en place une qualité de service dans des routeurs. Dans la pile TCP/IP, il est situé au même niveau que ICMP et IGMP (paquet IP de type 46, adresse unicast ou multicast identique à celle du flux). Il est cependant possible de l'encapsuler dans un paquet UDP (ports 1698 et 1699, adresse 224.0.0.14).

Les paquets RSVP sont routés saut par saut, c'est-à-dire de routeur à routeur : à chaque routeur traversé, les paquets remontent donc à la couche RSVP, puis sont réémis. Le routage des paquets est assuré par un protocole de routage quelconque, unicast ou multicast selon le type d'adresse.

Le protocole doit être activé sur chaque interface :

```
int s 0
ip rsvp bandwidth
```

Par défaut, 75% de la bande passante est géré par RSVP

Un émetteur de flux multimédia envoie périodiquement une annonce de chemin à un destinataire unicast ou à un groupe multicast (message *Path*). Chaque routeur, situé entre l'émetteur et le (ou les) destinataire, garde une trace de cette annonce avec les interfaces d'entrée et de sortie du paquet.

Chaque destinataire peut demander à réserver des ressources correspondant à une qualité de service telle que définie dans la RFC 2210 (message *Resv*).

Le logiciel RSVP, qui reçoit une demande de réservation, procède aux traitements tels que spécifiés par la RFC 1633 :

- contrôle d'admission afin de garantir que la demande ne remettra pas en cause la QoS d'autres flux (l'application a-t-elle le droit de faire une telle demande ? Est-elle raisonnable ? etc.) ;
- affectation du flux à une classe de service permettant d'obtenir la QoS désirée.

Si ces contrôles sont positifs, le message *Resv* est envoyé à l'émetteur en suivant la même route que le message *Path* émis. Chaque routeur situé entre le destinataire et l'émetteur du flux garde une trace de cette demande, et procède au même contrôle que décrit précédemment. Chaque routeur peut donc accepter ou refuser la demande (voir *figure 14-1* page 308).

RSVP ne définit que le moyen d'établir une qualité de service entre un émetteur et un destinataire en garantissant que les routeurs réserveront suffisamment de ressources (mémoire, CPU, files d'attente, bande passante sur les liaisons, etc.) pour assurer cette qualité de service.

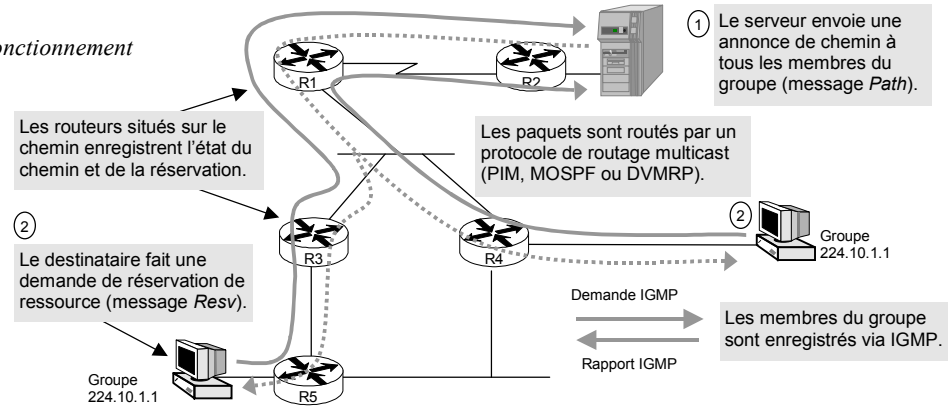


Figure 14-1.
Principe de fonctionnement de RSVP.

Sur nos routeurs, il est possible de définir quelle bande passante va pouvoir être gérée par RSVP :

```
int s 0
ip rsvp bandwidth 200 20
fair-queue 64 256 10
```

200 kbit/s gérée par RSVP

20 kbit/s par flux

10 sites d'attente pour RSVP

L'algorithme WFQ a également été activé pour gérer la qualité de service demandée. Dix files d'attente sont ainsi dédiés à RSVP : c'est, en effet, le nombre de flux auquel nous nous attendons (10 × 20 Kbit/s = 200 Kbit/s). Les valeurs par défaut, 64 et 256, ont été conservées pour les autres files d'attente.

LE POINT SUR RSVP (RFC 2205 À 2210)

L'émetteur d'un flux (vidéo, par exemple) envoie régulièrement aux destinataires un message **Path** leur permettant, ainsi qu'aux routeurs situés sur le chemin emprunté par le paquet RSVP, de déterminer la route empruntée par les paquets IP du flux. Les destinataires renvoient régulièrement à l'émetteur un message **Resv** qui est pris en compte par tous les routeurs situé sur le chemin qui vient d'être déterminé. Cette méthode permet de s'assurer que les ressources seront réservées sur le chemin emprunté par les paquets du flux multimédia. Au sein d'un réseau IP, le chemin retour peut, en effet, être différent du chemin aller.

Une demande de réservation consiste en un descripteur de flux composé d'un spécificateur de flux, **flowspec** (c.a.d. la QoS désirée qui permet de paramétrer l'ordonnanceur de paquets) et d'un filtre de flux, **filterspec** (c.a.d. les caractéristiques du trafic qui permettent de paramétrer le classificateur de paquet).

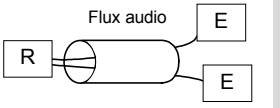
Un filtre consiste, au minimum, en une adresse IP et en un numéro de port TCP/UDP. Une forme plus évoluée peut inclure des données issues des couches applicatives ou le champ TOS du paquet IP.

●●●

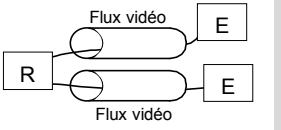
LE POINT SUR RSVP (SUITE)

Il existe plusieurs **styles de réservation** selon que la demande est partagée ou non par plusieurs flux et qu'elle concerne explicitement ou non les émetteurs :

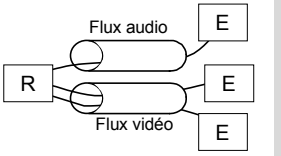
WF (*Wildcard-Filter*) : une seule réservation est faite pour les flux de tous les émetteurs (par exemple, pour une audioconférence où le nombre de personnes parlant en même temps est limité).



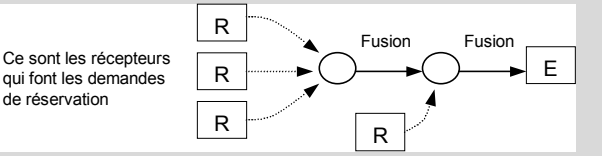
FF (*Fixed-Filter*) : une réservation est faite par flux (par exemple, une réservation par flux vidéo émis par chaque participant d'une visioconférence).



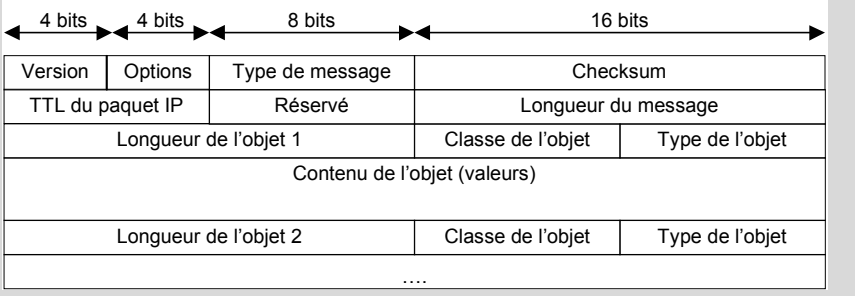
SE (*Shared Explicit*) : une seule réservation est faite pour les flux de quelques émetteurs (elle est adaptée, par exemple, à une visioconférence associée à une conférence audio).



Un routeur peut recevoir des demandes de réservation en provenance de différents destinataires qui concernent un même flux émis par un même émetteur. Les QoS demandées seront alors fusionnées en une seule qui sera remontée, et ainsi de suite jusqu'à l'émetteur du flux. La réservation qui est remontée jusqu'à l'émetteur du flux dépend alors du style des réservations émises par les destinataires.



Un message RSVP comprend plusieurs objets, eux-mêmes structurés en plusieurs champs. Par exemple, un message *Resv* comprend les adresses IP de l'émetteur et du destinataire du message RSVP, la périodicité d'envoi du message et le style de réservation, suivi des descripteurs de flux.
Afin de tenir compte des changements de topologie (apparition et disparition des routeurs et des membres de groupe), les messages *Path* et *Resv* sont envoyés périodiquement.



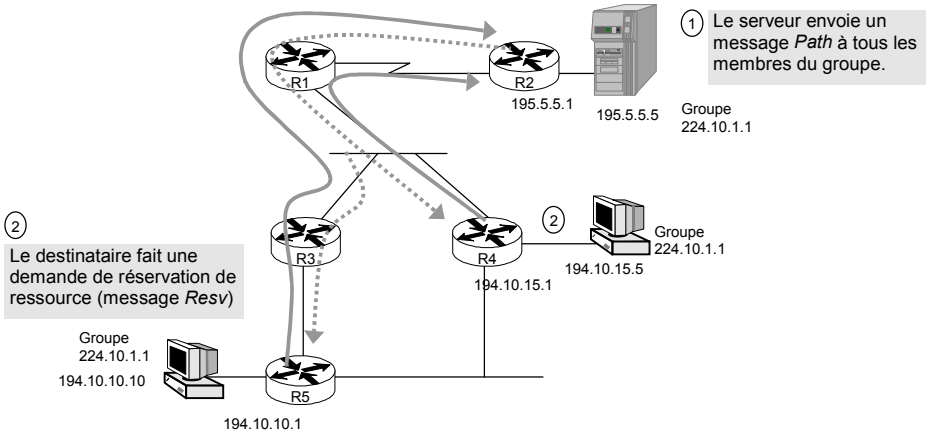
Ce qui vient d'être expliqué implique deux choses :

- que les piles TCP/IP des PC et des serveurs supportent le protocole RSVP ;
- et que les applications sachent décrire et demander une QoS.

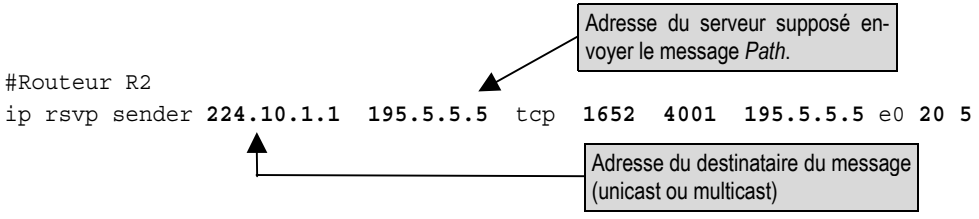
Le premier point est résolu avec Windows 98, Windows NT 5.0 et Windows 2000. Le second est, quant à lui, résolu par les API RSVP disponibles dans Winsock 2.0. et sous la plupart des versions d'Unix. Ainsi, certaines applications, telles qu'Internet Explorer ou Netmeeting, prennent directement en charge ce protocole pour les flux audio et vidéo.

Mais peu d'applications prennent actuellement en charge RSVP et les spécificateurs de QoS. Heureusement, nos routeurs offrent une fonctionnalité intéressante qui consiste à simuler l'émission des messages *Path* et *Resv* par un serveur ou un PC.

Dans notre cas, qui sert d'exemple, un serveur doit diffuser une vidéoconférence à différents utilisateurs *via* le protocole Proshare d'Intel. L'architecture est la suivante :



La commande suivante permet de simuler la réception de messages *Path par notre serveur vidéo* :



Le routeur agit comme si une application sur le serveur d'adresses IP 195.5.5.5 lui envoyait un message *Path* pour l'adresse de groupe multicast 224.10.1.1.

Les paramètres qui suivent sont :

- les ports destination (tcp 1652 = vidéoconférence Proshare) et source (4001) – les mots clés " udp " ou un port IP quelconque sont également acceptés ;
- l'adresse du saut précédent, donc celle du serveur ou du routeur le plus proche du serveur ;
- l'interface physique par laquelle est censé transiter le message émis par le serveur ;
- la bande passante, en Kbit/s, à réserver (ici 20 Kbit/s) ;
- Le dépassement autorisé (*burst*), en Ko (ici 5 Ko).

En retour, il est nécessaire de simuler l'envoi d'un message *Resv* par le destinataire du flux :

```
#Routeur R5
ip rsvp reservation 224.10.1.1 195.5.5.5 tcp 1652 4001 194.10.10.10 e0 SE
rate 20 5
```

Le routeur agit comme si une application sur le PC envoyait un message *Resv* pour l'adresse de groupe multicast 224.10.1.1 et vers le serveur 195.5.5.5.

Les paramètres qui suivent sont :

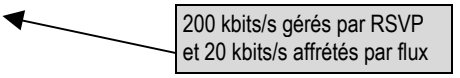
- les ports destination (1652 = vidéoconférence Proshare) et source – le mot clé " udp " ou un port IP quelconque sont également acceptés ;
- l'adresse du saut précédent, donc celle du PC ou du routeur le plus proche du PC ;
- l'interface physique par laquelle est censé transiter le message émis par le serveur ;
- le style de réservation : FF (*Fixed-Filter*), WF (*Wildcard-Filter*) ou SE (*Shared-Explicit*) que nous avons retenu pour notre flux vidéo ;
- la classe de service demandée (voir paragraphe suivant) : " *rate* " (QoS garantie) ou " *load* " (contrôle de charge) ;
- la bande passante, en Kbit/s, à réserver (ici 20 Kbit/s) ;
- le dépassement autorisé (*burst*), en Ko (ici 5 Ko).

Nous devons également nous assurer que les flux Telnet disposeront toujours de la bande passante suffisante et des temps de réponse corrects :

```
#Routeur R2
Message Path    Émetteur  |Récepteur     |Flux telnet |Émetteur
ip rsvp sender 195.5.5.5 194.10.10.10 tcp 23 1025 195.5.5.1 e0

#Routeur R5
Message Resv          Émetteur  |Récepteur     |Flux telnet|Récepteur
ip rsvp reservation 195.5.5.5 194.10.10.10 tcp 23 1025 194.10.10.1 e0 FF
load 32 1
```

Toutes les interfaces de sortie (par rapport au flux du serveur vers le PC) de tous les routeurs situés sur le chemin emprunté par le flux vidéo doivent pouvoir gérer la qualité de service demandée, ce qui est réalisé *à travers* la file d'attente WFQ. Si l'on ne considère que notre flux vidéo, cela donne :

```
# Routeur
int e1        (int e0 pour R2, in s0 pour R1, int e2 pour R4)
ip rsvp bandwidth 200 20
fair-queue 64 256 10
```



Pour les routeurs situés au sein de notre réseau qui utiliseraient l'algorithme WRED, il est possible de modifier les seuils de rejet appliqués par défaut à RSVP :

```
random-detect precedence rsvp 205 218 10
```

Les paramètres montrés sont ceux par défaut ; leur signification est identique à celle donnée en page 296.

La description de la qualité de service

RSVP est un protocole qui permet aux applications et aux routeurs de mettre en place une qualité de service en échangeant des informations. Mais, comment une application peut-elle décrire la qualité de service dont elle a besoin ? Quels paramètres doit-elle fournir aux routeurs ? Comment les routeurs configurent-ils leur classificateur et leur ordonnanceur ?

La réponse à ces questions passe par le respect d'un langage commun, qui est décrit dans les RFC 2210 à 2216. Ce langage commun repose sur trois types de paramètres qui permettent de caractériser la qualité de service :

- la classe de service demandée ;
- les paramètres décrivant la classe de service ;
- les paramètres décrivant les caractéristiques du flux de données pour lequel cette demande de qualité de service est faite.

Ces paramètres sont regroupés dans des objets transportés dans les messages *Resv*.

Les classes de service

Le support d'une classe de service implique que les routeurs paramétreront leur classificateur et leur ordonnanceur de manière identique et que, en définitive, ils se comporteront de la même manière, afin d'assurer une qualité de service uniforme, celle qu'attend l'application.

Deux classes de service sont actuellement définies :

- Un service de **contrôle de charge** (RFC 2211). En prenant en charge ce type de service, le routeur garantit que la plupart des paquets seront routés sans erreur et que le délai de transit n'excédera pas un certain seuil pour la plupart des paquets routés.
- Un **service garanti** (RFC 2212). En prenant en charge ce type de service, le routeur garantit que le délai de transit des paquets ne dépassera pas un seuil fixé et que l'application disposera de la bande passante demandée.

Description des classes de service

La RFC 2215 définit, quant à elle, les paramètres utilisés pour calculer les classes de service :

- *non-is_hop* : information décrivant qu'un routeur ne gère aucune QoS (routeurs sans intégration de service).
- *number_of_is_hops* : nombre de sauts à intégration de service (c'est-à-dire de routeurs gérant une QoS).
- *available_path_bandwidth* : bande passante disponible le long d'une route (qui traverse plusieurs routeurs et plusieurs réseaux).
- *minimum_path_latency* : délai minimal de transit d'un paquet le long d'une route (dépend notamment de la bande passante et du délai de traitement des routeurs).
- *path_mtu* : taille maximale des paquets le long d'une route : c'est la plus grande MTU (*Maximum Transmission Unit – 1 500 octets pour Ethernet*) autorisée par les différents types de réseaux empruntés par un flot de données.

Caractéristiques des flux

La même RFC 2215 définit également les paramètres que l'application communique aux routeurs *via* RSVP (regroupés dans une structure appelée *token_bucket_tspec*). Elle décrit les caractéristiques du flux pour lequel l'application fait une demande de classe de service. Il s'agit :

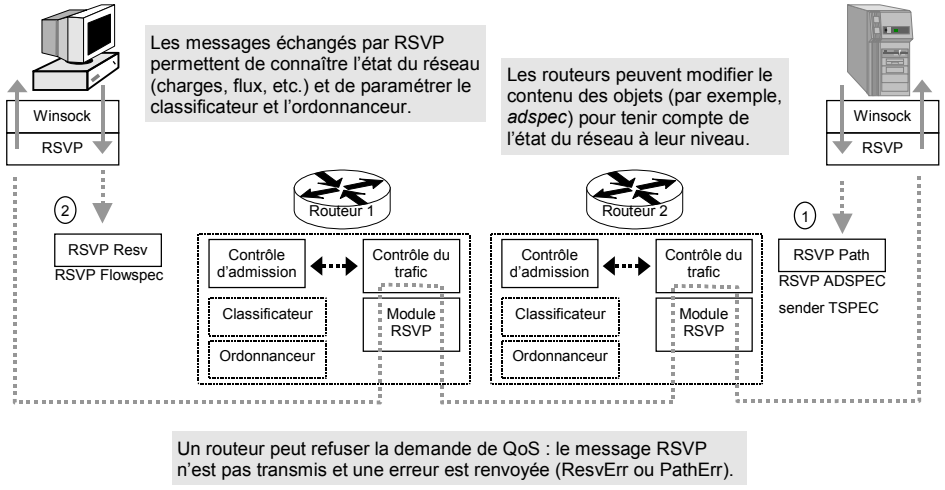
- du débit moyen du flux (*token rate*), en octets par secondes ;
- du dépassement de débit (*bucket size*) autorisé pendant une courte période, en d'autres termes le *burst*, exprimé en nombre d'octets ;
- du débit en pointe (*peak rate*), en octets par secondes ;
- de l'unité de contrôle minimale, en octets, qui correspond à la taille minimale des paquets. Le routeur contrôle en permanence le débit, et l'ajuste par rapport au *token-bucket* (*rate* + *size*) et à la taille des paquets. Les paquets dont la taille est inférieure à cette valeur seront traités, dans le calcul de la QoS, comme ayant cette valeur ;
- de la taille maximale des paquets en nombre d'octets.

Objets RSVP

Ces paramètres sont regroupés dans des objets transportés dans les messages *Resv* de RSVP. Ces objets sont décrits dans la RFC 2210 :

- *flowspec* : est généré par le récepteur vers le ou les émetteurs d'un flux afin d'indiquer la QoS désirée. Les routeurs peuvent modifier les informations *flowspec* en cours de chemin, notamment lors de la fusion RSVP.
- *sender_tspec* : indique les caractéristiques des flux que le ou les émetteurs vont générer. Les routeurs intermédiaires prennent en compte cette information sans la modifier.
- *rsvp_adspec* : est généré par le ou les émetteurs d'un flux, mais modifié par les routeurs intermédiaires. Cet objet contient des informations collectées dans le réseau, relatives aux délais de transit dans chaque nœud, à la bande passante mesurée sur les liaisons, à la classe de service supportée, etc. Chaque routeur modifie ou ajoute ses propres informations dans ce paquet.

Figure 14-2.
RSVP avec l'intégration de service (RFC 2210).



L'objet *flowspec* contient les paramètres suivants :

- la classe de service demandée : garantie ou contrôle de charge ;
- tspec (*traffic specification*) : correspond aux paramètres token_bucket_tspec ;
- *rspec* (*request specification*) : si la classe de service demandée est de type garantie. Cet objet décrit les caractéristiques de la classe de service, c'est-à-dire le débit (exprimé en paquets par seconde) et la gigue maximale autorisée (exprimée en microsecondes).

L'objet *sender_tspec* contient uniquement les paramètres *token_bucket_tspec*.

L'objet *rsvp_adspec* contient tout ou partie des paramètres de description de classe de service. Ils sont modifiés ou ajoutés par les routeurs pour décrire des caractéristiques locales et, en définitive, décrire les caractéristiques globales pour la route.

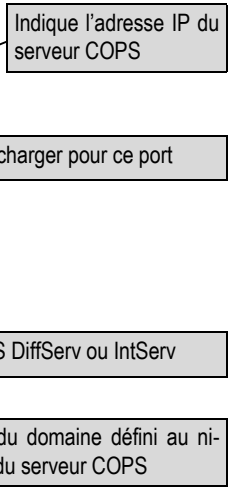
Définir une politique de qualité de service

L'application manuelle d'une politique de qualité de service sur l'ensemble des routeurs et des commutateurs peut se révéler fastidieuse et être source d'erreur.

Il est possible d'automatiser ce processus grâce au protocole COPS (*Common Open Policy Service*). La politique de qualité de service est alors définie à l'aide d'une interface graphique au niveau d'un serveur COPS, puis diffusée à partir de celui-ci aux routeurs du réseau. La création des règles et leur diffusion sont ainsi simplifiées.

Il faut avant tout indiquer aux routeurs de télécharger leur politique en utilisant le protocole COPS à partir d'un serveur :

```
# Sur un routeur :
ip rsvp policy cops servers 10.0.20.2

# Sur un commutateur :
set qos policy-source cops
set port qos 1/1 roles regulation
set cops server 10.0.20.2 diff-serv

# Sur un autre commutateur :
set qos policy-source cops
set cops server 10.0.20.2 rsvp
set cops domain-name domaine_multimedia
```



Concernant le serveur lui-même et son interface graphique, il existe différents produits, tels que *QoS Policy Manager* (Cisco), *Orchestream* (Network Computing), *PolicyXpert* (HP) et *RealNet Rules* (Lucent).