Guide pratique de la gestion de bande passante d’une ligne ADSL

Version française du ADSL Bandwidth Management HOWTO

Dan Singletary

`<dvsing CHEZ sonicspike POINT net>`

François Romieu

Adaptation française `<francois CHEZ ueimor POINT eu POINT org>`

Guillaume Lelarge

Adaptation française `<gleu CHEZ wanadoo POINT fr>`

Version : 1.3.fr.1.0

16 juillet 2004

Historique des versions		
Version 1.3.fr.1.0	2004-07-16	GL
Mise à jour de la traduction française.		
Version 1.3	2003-04-07	DS
Ajout d'une section de liens (*Added links section*).		
Version 1.2.fr.1.0	2003-03-01	FR, GL, JPG
Première traduction française.		
Version 1.2	2002-09-26	DS
Ajout d'un lien vers la nouvelle liste de diffusion. Ajout d'une petite information dans la section d'avertissement concernant la nouvelle QoS améliorée pour Linux, créée spécifiquement pour l'ADSL et bientôt disponible (*Added link to new Email Discussion List. Added small teaser to caveat section regarding new and improved QoS for Linux designed specifically for ADSL to be released soon*).		
Version 1.1	26-08-2002	DS
Quelques corrections (merci au nombreuses personnes m'ayant montré les problèmes!). Ajout d'une information dans la section d'implémentation. (*A few corrections (Thanks to the many that pointed them out!). Added informational caveat to implementation section.*)		
Version 1.0	21-08-2002	DS
Meilleur contrôle au niveau de la bande passante, plus de théorie, mise à jour pour les noyaux 2.4. (*Better control over bandwidth, more theory, updated for 2.4 kernels*)		
Version 0.1	06-08-2002	DS
Première publication (*Initial publication*)		

Résumé

Ce document décrit la configuration d'un routeur Linux pour gérer efficacement le trafic sortant à destination d'un modem ADSL ou de tout autre équipement de bande passante similaire (modem câble, RNIS, etc). Un accent particulier est apporté à la diminution de latence pour le trafic de type interactif et ce même durant les périodes de congestion.

Table des matières

1. Introduction

Ce document suggère une méthode de gestion de la bande passante pour le trafic sortant avec une connexion ADSL (ou modem câble) à Internet. Le problème provient du fait que de nombreuses lignes ADSL sont bridées aux environs de 128 kbps pour le trafic montant. La situation s’aggrave lorsque la file d’attente du modem ADSL demande de deux à trois secondes pour se libérer quand elle est pleine. Lorsque la bande passante dans le sens montant est saturée, une trame peut mettre jusqu’à trois secondes pour atteindre Internet. Les applications interactives comme telnet et les jeux en réseau sont dégradées.

1.1. Mises à jour du document

La dernière version de ce document se trouve sur l’Internet à l’adresse: http://www.tldp.org.

Les mises à jours seront reproduites dans divers sites web et ftp Linux tels que le LDP: http://www.tldp.org.

1.2. Liste de diffusion

Inscrivez-vous à la liste de diffusion de gestion de la bande passante pour l’ADSL http://jared.sonics-pike.net/mailman/listinfo/adsl-qos afin de poser des questions ou de recevoir des informations de mise à jour.

1.3. Avertissement

L'auteur, les distributeurs et les contributeurs de ce guide pratique ne sont en aucun cas responsables des dommages physiques, financiers, moraux ou de tout autre type occasionnés suite aux suggestions de ce texte.

1.4. Copyright

Important

Le texte ci-dessous est la licence de ce document. Ce texte fait foi. Il est composé de la licence (en anglais) du document original, suivi de la licence (en français) de sa traduction.

1.5. Droits d'utilisation

1.6. Retours d'expérience et corrections

Merci de faire parvenir à l'auteur vos questions et commentaires relatifs à ce document, en anglais, via l'adresse : dvsing@sonicspike.net.

N'hésitez pas à faire parvenir tout commentaire relatif à la version française de ce document à `<commentaires CHEZ traduc POINT org>` en précisant son titre, sa date et sa version.

2. Cadre d’utilisation

2.1. Prérequis

La méthode décrite dans ce document devrait s’appliquer à n’importe quelle configuration Linux. Toutefois, elle n’a pour l’instant été testée que dans la configuration suivante :

- Red Hat Linux 7.3
- Un noyau 2.4.18-5 avec une prise en charge de la QoS (éventuellement sous forme de module) qui inclut les correctifs suivants :
  - file d’attente HTB - http://luxik.cdi.cz/~devik/qos/htb/

    Remarque : on signale que les noyaux Mandrake au delà de la version 2.4.18-3 (versions 8.1 et 8.2 de la distribution Mandrake) incluent le correctif HTB.
  - périphérique IMQ - http://luxik.cdi.cz/~patrick/imq/

  Ces correctifs se retrouveront probablement intégrés en standard dans des versions ultérieures du noyau.
- iptables v1.2.6a ou suivantes. Le module length est absent de la version d’iptables distribuée avec la Red Hat 7.3

Note

Les versions antérieures de ce document impliquaient une méthode de contrôle de la bande passante qui passait par l’application d’un correctif à la queue sch_prio existante. Il s’est avéré que ce correctif était inutile. De surcroît, l’approche retenue dans ce texte donne de meilleurs résultats (quoiqu’à la date de rédaction de ce document, l’application de *2* correctifs au noyau soit nécessaire :o)).

2.2. Organisation

Toutes les mentions de périphériques réseau et de configuration dans ce guide pratique se rapportent au schéma suivant :

```
             <-- 128kbit/s      +------------+    <-- 10Mbit/s -->
Internet <--------------------> | Modem ADSL | <-------------------+
              1.5Mbit/s -->     +------------+                     |
                                                                   | eth0
                                                                   V
                                                       +---------------+
                                                       |               |
                                                       | Routeur Linux |
                                                       |               |
                                                       +---------------+
                                                        | .. | eth1..ethN
```

```
                                                          |     |
                                                          V     V

                                                       Réseau local
```

2.3. Files d’attente des paquets

Les files d’attente contiennent les données destinées à un périphérique réseau quand celles-ci ne peuvent pas être expédiées immédiatement. La plupart des files d’attente sont du type premier entré/premier sorti (FIFO/first in, first out) sauf lorsqu’elles sont explicitement configurées pour appliquer une autre stratégie. Cela signifie que lorsqu’une file d’attente est remplie, le paquet qui y a été placé en dernier n’est émis qu’après tous ceux qui étaient déjà présents dans la file.

2.3.1. Le lien montant

La bande passante d’un modem ADSL est asymétrique avec des valeurs typiques de 1,5 Mbit/s en descente et 128 kbit/s en trafic montant. Par rapport au débit de cette ligne, le routeur Linux et le modem ADSL sont généralement associés par un lien à 10 Mb/s ou plus. Si l’interface du routeur avec le réseau local est également à 10 Mb/s il n’y a aucune mise en attente au niveau du routeur lorsque les paquets transitent à destination d’Internet. Les paquets sont transmis via eth0 aussi rapidement qu’ils sont reçus du réseau local. Les paquets séjournent dans la file d’attente du modem ADSL puisqu’ils arrivent à 10 Mb/s et sont renvoyés à 128 kb/s. Une fois la file d’attente du modem saturée, les nouveaux paquets sont jetés. TCP s’adapte à ce phénomène et ajuste la taille de sa fenêtre de transmission pour employer toute la bande passante disponible.

Si les files d’attente et TCP s’accordent pour utiliser toute la bande passante, des tailles de FIFO importantes augmentent la latence du trafic à vocation interactive.

Les files d’attente à n voies sont similaires aux files d’attente de type FIFO à cette différence près qu’elles comprennent plusieurs files. Les paquets sont placés dans l’une des n files en fonction de leurs caractéristiques. Chaque file se voit attribuer une priorité et les paquets sont émis à partir de la file de plus haute priorité qui n’est pas vide. Avec cette stratégie, les paquets FTP peuvent être mis dans une file de priorité plus basse que les paquets destinés à telnet de telle sorte qu’un simple paquet telnet est capable de franchir la file d’attente immédiatement même lors d’un transfert FTP.

Ce document a été repris pour faire usage d’une nouvelle file d’attente dans Linux, dite de type HTB (Hierarchical Token Bucket). La file HTB ressemble à la file à n voies décrite précédemment mais elle rend possible la limitation de trafic dans chaque classe. En outre, elle autorise la création d’une hiérarchie de classes de trafic. Une description complète d’HTB dépasse le cadre de ce document. Davantage d’informations sont disponibles sur le site http://www.lartc.org

2.3.2. Le lien descendant

Le trafic entrant dans le modem ADSL en provenance d’Internet est mis en file d’attente de la même façon que le trafic sortant à ceci près que la file d’attente se situe chez le FAI. Il n’est donc guère possible de contrôler les priorités relatives des flux ou d’appliquer un traitement préférentiel à certains. La seule façon de maintenir une latence décente consiste à s’assurer que les interlocuteurs n’envoient pas les données trop vite. Il n’y a malheureusement pas de méthode directe. Comme une bonne partie du trafic est de type TCP, il est toutefois possible de ralentir les émetteurs :

- Jeter volontairement les paquets entrants. TCP est conçu pour employer la bande passante disponible tout en évitant la congestion du lien. Durant un échange TCP, les données sont émises jusqu'à ce qu'un paquet soit perdu. TCP remarque la perte et diminue sa fenêtre de transmission. Le cycle reprend, avec un rythme de progression des envois plus faible au cours du transfert et garantit une transmission aussi rapide que possible.

- Jouer avec les annonces de fenêtre de réception. Au cours d'un transfert TCP, le récepteur envoie un flux permanent de paquets d'acquittements (ACK). Les paquets ACK incluent une annonce de taille de fenêtre qui précise la quantité maximale de données non-acquittées qui peut être reçue. Ceci permet de ralentir le rythme d'émission. Il n'existe à ce jour pas de mise en oeuvre libre de ce type de contrôle de flux pour Linux (quoique je puisse être en train d'y travailler).

3. Fonctionnement

L'optimisation de la bande passante montante s'effectue en deux étapes. Tout d'abord il faut éviter que le modem ADSL ne mette les paquets en attente car on ne peut pas contrôler la façon dont il gère sa file. Ceci s'effectue en limitant la quantité de données émise par le routeur via eth0 un peu en dessous de la bande passante disponible. Le routeur met en file les paquets qui arrivent du réseau local plus vite qu'il ne les émet.

La seconde étape consiste à mettre en oeuvre une stratégie de file d'attente au niveau du routeur. On examinera une file d'attente qui peut être configurée pour donner la priorité au trafic interactif tel que telnet ou les jeux à plusieurs participants en réseau.

Note

Les files HTB permettent à la fois la limitation de trafic et l'envoi prioritaire tout en assurant qu'aucune classe de priorité n'étouffe les autres. Ce dernier point n'était pas possible avec la méthode préconisée par la version 0.1 de ce document.

La dernière étape porte sur le marquage des paquets au niveau du pare-feu pour affecter des priorités aux paquets avec fwmark.

3.1. Limitation du trafic sortant avec HTB Linux

Bien que le lien entre le routeur et le modem soit à 10 Mb/s, voire plus, le modem n'émet au mieux les données qu'à 128 kbit/s. Au delà de cette vitesse, les données sont mise en attente dans la file du modem. Un paquet de ping peut atteindre le modem immédiatement mais devoir attendre quelques secondes avant de rejoindre Internet si la file d'attente du modem est remplie. La plupart des modems ADSL ne permettent pas de contrôler la façon dont les paquets sont retirés de la file d'attente ni quelle est la taille de cette dernière. Le premier objectif consiste donc à déplacer le point de congestion des paquets sortants à un endroit où on peut exercer suffisamment de contrôle.

La file HTB limite le rythme d'envoi des paquets au modem ADSL. Bien que le rythme montant puisse atteindre 128 kb/s, on doit le brider à une valeur légèrement inférieure. Pour limiter la latence, il faut être certain du fait qu'aucun paquet n'est mis en attente au niveau du modem. L'expérience m'a indiqué qu'une limitation à 90 kb/s du trafic sortant me donne 95 % de la bande passante atteinte en l'absence de HTB. L'activation de HTB à ce rythme prévient la mise en file d'attente par le modem ADSL.

3.2. Gestion des files de priorité avec HTB

Note

Remarque: les affirmations antérieures de cette section (concernant la mise en file d'attente à N voies) se sont avérées erronées. En l'occurrence, il était possible de classer les paquets dans les éléments de la file de priorité juste au moyen du champ fwmark. Cette fonctionnalité n'était toutefois guère documentée lors de la rédaction de la version 0.1 du guide pratique.

Pour l'instant, les performances n'ont pas été modifiées. La file de type FIFO des paquets a simplement été déplacée du modem ADSL au routeur. Comme Linux a une longueur de file égale à 100 paquets par défaut, la situation s'est probablement aggravée (mais pas pour longtemps).

Chaque classe adjacente dans une file HTB peut se voir attribuer une priorité. En plaçant différents types de trafic dans des classes distinctes et en affectant à ces classes des priorités spécifiques, l'ordre dans lequel les paquets sont extraits de la file puis émis est contrôlable. HTB le permet tout en évitant qu'une classe ne soit éteinte puisqu'une bande passante minimale pour chaque classe peut être garantie. En outre, HTB autorise une classe à utiliser jusqu'à un certain niveau la bande passante laissée libre par les autres classes.

Une fois les classes en place, on installe des filtres qui répartissent le trafic dans les classes. Plusieurs approches sont envisageables mais le document s'appuie sur les utilitaires courants ipchains/iptables pour marquer les paquets avec un indicateur fwmark. Les filtres dirigent le trafic dans les classes de la file HTB selon leur indice fwmark. De cette façon, les règles de reconnaissance d'iptables aiguillent certains trafics suivant leur classe.

3.3. Classification des paquets sortant avec iptables

Note

Remarque: cette documentation reposait à l'origine sur ipchains pour trier les paquets. iptables est à présent utilisé.

La dernière étape de configuration du routeur pour augmenter la priorité du trafic interactif consiste à spécifier au filtre comment identifier le trafic. Ceci s'effectue avec le champ fwmark des paquets.

Sans trop rentrer dans les détails, voici une description simplifiée du cheminement des paquets entre quatre classes dont celle d'indice 0x00 a la priorité la plus élevée:

1. On marque tous les paquets avec l'indice 0x03. Ceci les place tous dans la file de priorité la plus basse.

2. On marque les paquets ICMP avec l'indice 0x00 afin que ping fournisse la latence des paquets de priorité la plus élevée.

3. On marque tous les paquets dont le port destination est inférieur à 1024 comme 0x01. Les services système tels telnet et ssh voient leur priorité augmenter. Le port de contrôle de ftp rentre dans cette catégorie. Toutefois, les transferts de données ftp ont lieu avec des ports situés plus haut et ils restent donc dans la catégorie 0x03.

4. On marque tous les paquets à destination du port 25 (SMTP) avec un indice 0x03 afin d'éviter que l'envoi d'un courrier muni d'un attachement volumineux n'empiète sur le trafic interactif.

5. On marque tous les paquets en direction d'un serveur de jeu avec un indice 0x02. Les joueurs fous auront une bonne latence sans écraser les applications système qui demandent une latence faible.

   On marque tous les paquets de petite taille avec un indice 0x02. Les paquets de type ACK des téléchargements doivent être retournés rapidement afin d'assurer des transferts efficaces. Ceci est possible grâce au module adéquat (c'est-à-dire length) d'iptables.

Tout ce qui précède est bien sûr personnalisable en fonction des besoins.

3.4. Des réglages supplémentaires...

Deux choses sont encore susceptibles d'améliorer la latence. La MTU peut être positionnée en dessous de la valeur par défaut de 1500 octets. Sa diminution se répercute sur le temps moyen d'attente lors de l'envoi d'un paquet prioritaire lorsqu'un paquet de faible priorité est déjà en cours de transmission. La bande passante en souffre puisque le poids relatif des informations d'en-tête augmente (40 octets pour le couple TCP et IP).

La longueur de queue de 100 paquets par défaut, qui demande jusqu'à 10 secondes pour se vider avec une ligne ADSL et une MTU de 1500 octets, peut également être abaissée.

3.5. Une tentative de limitation du trafic entrant

L'emploi d'un périphérique de file intermédiaire (IMQ ou Intermediate Queuing Device) place tous les paquets entrants dans une file d'une façon analogue au traitement des paquets sortants. La gestion de la priorité en est simplifiée. On met tout le trafic hors TCP dans la classe 0x00, le trafic TCP étant quant à lui dans la classe 0x01. Les petits paquets vont également dans la classe 0x00 car il s'agit vraisemblablement d'acquittements de données sortantes déjà émises. Une file d'attente FIFO standard est appliquée à la classe 0x00 et une queue de rejet anticipé aléatoire (Random Early Drop/RED) traite la classe 0x01. RED est meilleur qu'une FIFO pour le contrôle de TCP en ce qu'il rejette des paquets avant que la file ne déborde afin de ralentir le trafic qui semble sur le point de devenir incontrôlable. Les deux classes sont également bornées supérieurement à une valeur inférieure à la capacité maximale de la ligne ADSL.

3.5.1. Problèmes soulevés par la limitation de trafic entrant

On souhaite limiter le trafic entrant afin de ne pas remplir la file d'émission du côté du FAI qui est susceptible de contenir jusqu'à 5 secondes de données. La seule façon de limiter le trafic entrant consiste à jeter des paquets tout à fait valables. Ces paquets ont déjà consommé leur quote part de bande passante et le routeur Linux les jette afin de limiter le rythme d'arrivée des paquets futurs. Ces paquets seront sûrement retransmis et consommeront davantage de bande passante. La limitation du trafic porte sur le rythme auquel les paquets vont être acceptés. Comme le taux *courant* est bien supérieur en raison des paquets jetés, la bande passante descendante doit être *bien* inférieure à la capacité de la ligne pour maintenir une latence faible. En pratique, j'ai dû brider ma connexion ADSL descendante de 1,5Mb/s à 700kb/s afin de conserver une bonne latence avec 5 téléchargements simultanés. Plus les sessions TCP sont nombreuses, plus la bande passante perdue dans les paquets jetés est élevée et plus il faut limiter le taux de transfert.

Une meilleure méthode consisterait à jouer sur la fenêtre TCP mais je ne connais pas d'outil libre pour le faire sous Linux.

4. Réalisation

Après toutes ces explications, il est temps de passer à la mise en oeuvre sous Linux.

4.1. Avertissements

Brider le rythme d'émission des données vers le modem ADSL n'est pas aussi simple qu'il y parait. La plupart des modems DSL établit juste un pont Ethernet entre le routeur Linux et la passerelle du côté du FAI. La couche de liaison de données s'appuie le plus souvent sur ATM qui envoie les données au moyen de cellules de 53 octets. Cinq octets sont consommés pour l'en-tête ATM laissant ainsi 48 octets de données utiles. La transmission d'un simple octet de données isolé ne peut donc pas demander moins de 53 octets au niveau du lien ATM. Dans le cas d'un acquittement TCP typique qui comprend 0 octet de données, 20 octets d'en-tête TCP, autant d'en-tête IP et 18 de préambule Ethernet, les données utiles (40 octets) sont inférieures au minimum de 48 octets requis par une trame Ethernet. Afin d'envoyer les 64 (48+16) octets via ATM, deux cellules ATM sont nécessaires, d'où une consommation de 106 octets de bande passante au niveau du lien ADSL. Chaque paquet TCP d'acquittement gaspille donc 42 octets de bande passante. Ceci ne serait pas gênant si Linux prenait en compte l'encapsulation due au modem ADSL. Il ne peut malheureusement pas le faire et se cantonne aux en-têtes TCP et IP ainsi qu'aux 14 octets d'adresse MAC (les quatre octets de CRC gérés au niveau matériel sont ignorés). Linux ne prend pas en compte la taille minimale de trame Ethernet ni la taille de cellule ATM.

Tout ceci pour convaincre qu'il faut limiter le trafic sortant à une valeur sensiblement inférieure à la véritable capacité de la ligne (jusqu'à ce qu'on dispose d'un ordonnanceur de paquets capable de prendre en compte les différentes encapsulations employées). Vous pouvez penser avoir trouvé un bon réglage mais constater des sauts de latence au delà de trois secondes lors d'un téléchargements important. Ceci viendra probablement d'une mauvaise évaluation de la bande passante consommée par les petits paquets d'acquittement.

Je travaille depuis quelques mois sur une solution à ce problème et j'ai presque terminé quelque chose qui sera publié afin que chacun puisse le tester. Ma solution repose sur une file en espace utilisateur à la place de la QoS de Linux pour limiter le rythme de transfert des paquets. Il s'agit d'une variante d'HTB en espace utilisateur. Pour l'instant cette solution a été capable de réguler le trafic sortant lors de téléchargements massifs (plusieurs flux) et de transferts point-à-point en volume (gnutella, sic) avec une telle efficacité que la latence dépasse au plus de 400 ms la latence nominale de 15 ms inhérente à mon lien ADSL. Pour davantage d'informations, abonnez-vous à la liste de diffusion ou surveillez les mises à jour de ce document.

4.2. Script: mon_limiteur

Ci-suit le source d'un script que j'emploie pour gérer la bande passante de mon routeur Linux. Il s'appuie sur les concepts expliqués dans ce guide. Le trafic sortant est ventilé entre une des sept files disponibles en fonction de son type. Le trafic entrant est placé dans une file parmi deux, le trafic TCP étant jeté en premier en cas de surcharge de la ligne. Les valeurs conviennent à mon installation mais les résultats peuvent être différents ailleurs.

Note

Ce script s'inspire du WonderShaper ADSL disponible sur le site du LARTC.

```
#!/bin/bash
#
# mon_limiteur - Limiteur et classificateur de trafic pour modem Cable ou ADSL.
#                Inspiré de WonderShaper (www.lartc.org)
#
# Écrit par Dan Singletary (7/8/02)
#
# Remarque - ce script suppose que le noyau a été patché avec les files
#            HTB et IMQ disponibles ici (les noyaux à venir ne demanderont
#            pas forcément l'application d'un correctif):
#       http://luxik.cdi.cz/~devik/qos/htb/
#       http://luxik.cdi.cz/~patrick/imq/
#
# Options de configuration pour mon_limiteur:
#  DEV    - correspond au périphérique ethX connecté au modem
#  RATEUP - à positionner à une valeur inférieure à la bande
#           passante montante de la ligne.
#           Pour ma ligne ADSL en 1500/128, RATEUP=90 convient au rythme
#           montant de 128kbps. A vous d'ajuster.
#  RATEDN - à positionner en dessous de la bande passante descendante de
#           la ligne.
#
#
#  Principe d'utilisation d'imq pour limiter le trafic entrant:
#
#    Il est impossible de limiter directement le rythme auquel les
#  données vous sont envoyées depuis l'Internet. Afin de limiter le
#  trafic entrant, on s'appuie sur les mécanismes anti-congestion de
#  TCP. Ceci signifie que SEUL LE TRAFIC TCP PEUT SE LIMITER. Le
#  trafic hors TCP est placé dans une queue prioritaire car le jeter
#  ne conduit vraisemblablement qu'à une retransmission ultérieure
#  qui accroît la bande passante consommée.
#  On limite le trafic TCP en jetant les paquets lorsqu'ils débordent
#  de la file HTB qui les limitera à un certain rythme (RATEDN)
#  légèrement inférieur à la capacité réelle de la ligne. Jeter ces
#  paquets revient à en singer la perte par la file d'émission du
#  côté du FAI. Ceci a l'avantage d'éviter la congestion de la file
#  d'émission chez le FAI puisque TCP ralentira avant qu'elle ne
#  se remplisse. L'usage d'une stratégie de mise en attente basée sur
#  la classification des paquets par priorité permet de ne PAS jeter
#  certains types de paquets (ssh, telnet, etc). Les paquets ne sont
#  retirés des files d'attente de faible priorité qu'une fois que
#  chaque classe a atteint un seuil minimum (1/7 de la bande passante
#  dans ce script).
#
#  Résumé:
#   * La perte d'un paquet TCP diminue le rythme de réception de la
#     connexion associée via les mécanismes de contrôle de congestion.
#   * Jeter des paquets TCP n'apporte rien. S'ils sont importants, ils
#     seront retransmis.
#   * Limiter le rythme des connexions TCP entrantes en dessous de la
#     capacité de la ligne DEVRAIT éviter la mise en attente des paquets
#     du côté du FAI (DSLAM, concentrateur de cables, etc). L'expérience
#     indique que ces files contiennent 4 secondes de trafic à 1500kbps,
#     soit 6Mb de données. A ce niveau, l'absence de mise en attente
#     diminue la latence.
#
#  Avertissements:
#   * Est-ce que la limitation de bande passante diminue l'efficacité de
#     transferts TCP massifs ?
```

```
#     - Apparemment non. L'augmentation de priorité des paquets
#       d'acquittement maximise le débit en évitant de perdre de la bande
#       passante à retransmettre des paquets déjà reçus.
#

# NOTE: La configuration ci-dessous fonctionne avec ma connexion ADSL
# 1.5M/128K via Pacific Bell Internet (SBC Global Services)

DEV=eth0
RATEUP=90
RATEDN=700  # Nettement inférieur à la capacité de la ligne de 1500.
            # On n'a donc pas à limiter le trafic entrant jusqu'à ce
            # qu'une meilleure réalisation telle que la modification
            # de fenêtre TCP soit disponible.

#
# Fin des options de configuration
#

if [ "$1" = "status" ]
then
        echo "[qdisc]"
        tc -s qdisc show dev $DEV
        tc -s qdisc show dev imq0
        echo "[class]"
        tc -s class show dev $DEV
        tc -s class show dev imq0
        echo "[filter]"
        tc -s filter show dev $DEV
        tc -s filter show dev imq0
        echo "[iptables]"
        iptables -t mangle -L MONLIMITEUR-OUT -v -x 2> /dev/null
        iptables -t mangle -L MONLIMITEUR-IN -v -x 2> /dev/null
        exit
fi

# Remise à zéro
tc qdisc del dev $DEV root    2> /dev/null > /dev/null
tc qdisc del dev imq0 root 2> /dev/null > /dev/null
iptables -t mangle -D POSTROUTING -o $DEV -j MONLIMITEUR-OUT 2> /dev/null > /dev/null
iptables -t mangle -F MONLIMITEUR-OUT 2> /dev/null > /dev/null
iptables -t mangle -X MONLIMITEUR-OUT 2> /dev/null > /dev/null
iptables -t mangle -D PREROUTING -i $DEV -j MONLIMITEUR-IN 2> /dev/null > /dev/null
iptables -t mangle -F MONLIMITEUR-IN 2> /dev/null > /dev/null
iptables -t mangle -X MONLIMITEUR-IN 2> /dev/null > /dev/null
ip link set imq0 down 2> /dev/null > /dev/null
rmmod imq 2> /dev/null > /dev/null

if [ "$1" = "stop" ]
then
        echo "Limitation de débit désactivée sur $DEV."
        exit
fi

###########################################################
#
# Limitation de trafic sortant (limite supérieure à RATEUP)

# positionnement de la taille de la file d'émission pour obtenir
# une latence d'environ 2 secondes pour les paquets de la file
# de faible priorité.
ip link set dev $DEV qlen 30

# modification de MTU du périphérique sortant.
# - Diminuer la MTU abaisse la latence mais dégrade le débit en raison de
#   la surcharge IP et TCP.
ip link set dev $DEV mtu 1000

# ajout de la stratégie HTB
tc qdisc add dev $DEV root handle 1: htb default 26

# ajout de la classe de limitation principale
tc class add dev $DEV parent 1: classid 1:1 htb rate ${RATEUP}kbit

# ajout des classes filles:
# - chaque classe dispose AU MOINS de son quota de bande passante. Aucune
#   classe n'est donc étouffée par les autres. Chaque classe peut également
#   consommer toute la bande passante si aucune autre classe ne l'emploie.
tc class add dev $DEV parent 1:1 classid 1:20 htb rate $[$RATEUP/7]kbit ceil ${RATEUP}kbit prio 0
tc class add dev $DEV parent 1:1 classid 1:21 htb rate $[$RATEUP/7]kbit ceil ${RATEUP}kbit prio 1
tc class add dev $DEV parent 1:1 classid 1:22 htb rate $[$RATEUP/7]kbit ceil ${RATEUP}kbit prio 2
tc class add dev $DEV parent 1:1 classid 1:23 htb rate $[$RATEUP/7]kbit ceil ${RATEUP}kbit prio 3
tc class add dev $DEV parent 1:1 classid 1:24 htb rate $[$RATEUP/7]kbit ceil ${RATEUP}kbit prio 4
tc class add dev $DEV parent 1:1 classid 1:25 htb rate $[$RATEUP/7]kbit ceil ${RATEUP}kbit prio 5
tc class add dev $DEV parent 1:1 classid 1:26 htb rate $[$RATEUP/7]kbit ceil ${RATEUP}kbit prio 6

# ajout de la stratégie aux classes filles
# - SFQ offre un traitement sensiblement équitable de chaque classe.
tc qdisc add dev $DEV parent 1:20 handle 20: sfq perturb 10
tc qdisc add dev $DEV parent 1:21 handle 21: sfq perturb 10
tc qdisc add dev $DEV parent 1:22 handle 22: sfq perturb 10
tc qdisc add dev $DEV parent 1:23 handle 23: sfq perturb 10
tc qdisc add dev $DEV parent 1:24 handle 24: sfq perturb 10
tc qdisc add dev $DEV parent 1:25 handle 25: sfq perturb 10
tc qdisc add dev $DEV parent 1:26 handle 26: sfq perturb 10

# répartition du trafic en classe via fwmark
# - le trafic est réparti en classes de priorité suivant l'indicateur
#   fwmark des paquets (ceux-ci sont positionnés avec iptables un peu plus
#   loin). La classe de priorité par défaut a été mise à 1:26 de telle sorte
#   que les paquets qui ne sont pas marqués se retrouvent dans la classe de
#   priorité la plus faible.
tc filter add dev $DEV parent 1:0 prio 0 protocol ip handle 20 fw flowid 1:20
tc filter add dev $DEV parent 1:0 prio 0 protocol ip handle 21 fw flowid 1:21
tc filter add dev $DEV parent 1:0 prio 0 protocol ip handle 22 fw flowid 1:22
tc filter add dev $DEV parent 1:0 prio 0 protocol ip handle 23 fw flowid 1:23
tc filter add dev $DEV parent 1:0 prio 0 protocol ip handle 24 fw flowid 1:24
tc filter add dev $DEV parent 1:0 prio 0 protocol ip handle 25 fw flowid 1:25
tc filter add dev $DEV parent 1:0 prio 0 protocol ip handle 26 fw flowid 1:26
```

```
# ajout de MONLIMITEUR-OUT à la table de modification des paquets d’iptables
# - ceci déclare la table employée pour filtrer et classer les paquets
iptables -t mangle -N MONLIMITEUR-OUT
iptables -t mangle -I POSTROUTING -o $DEV -j MONLIMITEUR-OUT

# ajout de fwmark pour classer les différents types de trafic
# - fwmark est positionné de 20 à 26 suivant la classe. 20 correspond à la
#   priorité la plus forte.
iptables -t mangle -A MONLIMITEUR-OUT -p tcp --sport 0:1024 -j MARK --set-mark 23 # Trafic sur les ports bas
iptables -t mangle -A MONLIMITEUR-OUT -p tcp --dport 0:1024 -j MARK --set-mark 23 # ""
iptables -t mangle -A MONLIMITEUR-OUT -p tcp --dport 20 -j MARK --set-mark 26     # Port ftp-data, faible priorité
iptables -t mangle -A MONLIMITEUR-OUT -p tcp --dport 5190 -j MARK --set-mark 23   # Messagerie Immédiate AOL
iptables -t mangle -A MONLIMITEUR-OUT -p icmp -j MARK --set-mark 20               # ICMP (ping) - forte priorité (impressionnez vos amis)
iptables -t mangle -A MONLIMITEUR-OUT -p udp -j MARK --set-mark 21                # DNS (petits paquets)
iptables -t mangle -A MONLIMITEUR-OUT -p tcp --dport ssh -j MARK --set-mark 22    # shell sécurisé
iptables -t mangle -A MONLIMITEUR-OUT -p tcp --sport ssh -j MARK --set-mark 22    # shell sécurisé
iptables -t mangle -A MONLIMITEUR-OUT -p tcp --dport telnet -j MARK --set-mark 22 # telnet (hum...)
iptables -t mangle -A MONLIMITEUR-OUT -p tcp --sport telnet -j MARK --set-mark 22 # telnet (hum...)
iptables -t mangle -A MONLIMITEUR-OUT -p ipv6-crypt -j MARK --set-mark 24         # IPSec - la surcharge n’est pas connue...
iptables -t mangle -A MONLIMITEUR-OUT -p tcp --sport http -j MARK --set-mark 25   # Serveur WWW local
iptables -t mangle -A MONLIMITEUR-OUT -p tcp -m length --length :64 -j MARK --set-mark 21 # Petits paquets (des ACK probablement)
iptables -t mangle -A MONLIMITEUR-OUT -m mark --mark 0 -j MARK --set-mark 26      # Répétition - on marque les paquets restants à 26 (faible priorité)

# Fin de la limitation sortante
#
####################################################

echo "Limitation de trafic sortant activé sur $DEV.  Débit: ${RATEUP}kbit/sec."

# Décommenter la ligne suivante pour n’avoir que la limitation de trafic montant.
# exit

####################################################
#
# Limitation du trafic entrant (débit maximal de RATEDN)

# on force le chargement du module imq

modprobe imq numdevs=1

ip link set imq0 up

# ajout de la stratégie de mise en file d’attente
# - par défaut une classe 1:21 à faible priorité

tc qdisc add dev imq0 handle 1: root htb default 21

# ajout de la classe de limitation principale
tc class add dev imq0 parent 1: classid 1:1 htb rate ${RATEDN}kbit

# ajout des classes filles
# - trafic TCP en 21, le reste en 20
#
tc class add dev imq0 parent 1:1 classid 1:20 htb rate $[$RATEDN/2]kbit ceil ${RATEDN}kbit prio 0
tc class add dev imq0 parent 1:1 classid 1:21 htb rate $[$RATEDN/2]kbit ceil ${RATEDN}kbit prio 1

# ajout de la stratégie de limitation aux classes filles
# - voir les remarques ci-dessus sur SFQ.
tc qdisc add dev imq0 parent 1:20 handle 20: sfq perturb 10
tc qdisc add dev imq0 parent 1:21 handle 21: red limit 1000000 min 5000 max 100000 avpkt 1000 burst 50

# répartition du trafic en classe via fwmark
# - le trafic est réparti en classes de priorité suivant l’indicateur
#   fwmark des paquets (ceux-ci sont positionnés avec iptables un peu plus
#   loin). La classe de priorité par défaut à été mise à 1:26 de telle sorte
#   que les paquets qui ne sont pas marqués se retrouvent dans la classe de
#   priorité la plus faible.
tc filter add dev imq0 parent 1:0 prio 0 protocol ip handle 20 fw flowid 1:20
tc filter add dev imq0 parent 1:0 prio 0 protocol ip handle 21 fw flowid 1:21

# ajout de MONLIMITEUR-IN à la table de modification des paquets d’iptables
iptables -t mangle -N MONLIMITEUR-IN
iptables -t mangle -I PREROUTING -i $DEV -j MONLIMITEUR-IN

# ajout de fwmark pour classer les différents types de trafic
# - fwmark est positionné de 20 à 21 suivant la classe. 20 correspond à la
#   priorité la plus forte.
iptables -t mangle -A MONLIMITEUR-IN -p ! tcp -j MARK --set-mark 20              # Forte priorité pour les paquets non TCP
iptables -t mangle -A MONLIMITEUR-IN -p tcp -m length --length :64 -j MARK --set-mark 20 # Les petits paquets TCP sont probablement des ACK
iptables -t mangle -A MONLIMITEUR-IN -p tcp --dport ssh -j MARK --set-mark 20    # shell sécurisé
iptables -t mangle -A MONLIMITEUR-IN -p tcp --sport ssh -j MARK --set-mark 20    # shell sécurisé
iptables -t mangle -A MONLIMITEUR-IN -p tcp --dport telnet -j MARK --set-mark 20 # telnet (hum...)
iptables -t mangle -A MONLIMITEUR-IN -p tcp --sport telnet -j MARK --set-mark 20 # telnet (hum...)
iptables -t mangle -A MONLIMITEUR-IN -m mark --mark 0 -j MARK --set-mark 21              # Répétition - les paquets sans marque sont positionnés à 21 (faible priorité)

# on envoie les paquets précédents à l’interface imq0.
iptables -t mangle -A MONLIMITEUR-IN -j IMQ

# Fin de la limitation de trafic entrant.
#
####################################################

echo "Limitation de trafic entrant activée sur $DEV.  Débit: ${RATEDN}kbit/sec."
```

5. Test

La méthode de test la plus facile consiste à saturer le lien montant avec du trafic à faible priorité. Si le trafic telnet et le ping ont une priorité élevée par opposition aux transferts FTP, l’augmentation de délai des pings lorsqu’un transfert FTP s’établit devrait être négligeable par rapport à ce qu’elle devient en l’absence de file d’attente. Des pings en dessous de 100 ms dépendent de la configuration. Des pings au dessus d’une ou deux secondes signalent une anomalie.

6. Ça fonctionne. Et maintenant ?

Maintenant que la bande passante est gérée, il faut s’en servir. Après tout, elle a été payée !

- Utilisez un client Gnutella et PARTAGEZ VOS FICHIERS sans dégrader les performances de votre réseau.

- Hébergez un site web sans que la consultation ne ralentisse vos sessions Quake.

7. Liens

- Contrôleur de bande passante pour Windows (*Bandwidth Controller for Windows*) - http://www.bandwidthcontroller.com

- dsl_qos_queue - (bêta) pour Linux. Pas de correctifs pour le noyau et de meilleurs performances