Module M3102 – TP3

QoS : Compétition entre flux, video streaming adaptatif et actions aux routeurs

Ce qu'on va faire dans ce TP : Analyser la compétition entre flux de streaming vidéo et flux de transferts de fichiers, puis configurer le routeur pour traiter ces flux différemment, et analyser le résultat.

Pourquoi : Pour permettre de satisfaire les contraintes en QoS des différents types d'applications, c'est-à-dire les contraintes en terme de débit et délai ici, ou simplement assurer un minimum de bande passante à certains flux au détriment d'autres.

Comment : Par l'utilisation de MQC de Cisco, qui est la *Modular QoS CLI (Command Line Interface)*, ensemble de commandes de Cisco IOS avec lesquelles on va notamment appliquer les mécanismes d'ordonnancement de files CBWFQ, priority, de gestion de buffer WRED. Nous considérons également une application multimédia intelligente (HTTP Adaptive Streaming).

Consignes générales :

- Lisez la section I en entier et attentivement avant de commencer.
- 1 compte-rendu par étudiant·e. Vous récupérez le pdf et l'odt du sujet depuis http://www.i3s.unice.fr/~sassatelli/M3102/
- Vous vous loggez sous votre compte personel, machine physique en Debian.
- Joindre au rapport le fichier de configuration finale du routeur (copié-collé de la config finale).

Table of Contents

I. Introduction aux outils

Le réseau considéré est présenté et doit être créé en lisant la section I.1.
Une façon de s'adapter à la QoS fluctuante de l'Internet est de rendre les applications d'extrémité plus intelligentes. C'est en particulier le cas des applications de streaming vidéo : le player video utilisé ainsi que le processus de streaming est présenté en Section I.2.
La section I.3 présente la gestion de QoS en Cisco, MQC, qu'on va utiliser.

I.1. Réseau considéré

Ci-dessous en Fig. 1 la topologie sur laquelle on va travailler : un serveur SSH et vidéo à gauche, un client récupérant des fichiers en scp et regardant une vidéo à droite. Le détail du format de la vidéo, du player et du serveur sont donnés dans la prochaine section I.2. Créez la topologie ci-dessous dans GNS3, en suivant les instructions ci-dessous :

- Ouvrez GNS3. Edit → Preferences → Dynamips → clic sur *Test Settings*
- Edit → IOS images → vérifiez que l'image 7200 est listée. Si ce n'est pas le cas, rajoutez-la depuis /home/VBox/ios/. Puis clic sur *Idle PC Auto calculation* et *Save.*
- Utiliser la commande `createvm` pour créer 2 VM avec SrvTapas_Ubuntu_14.04 et ClientTapas_Ubuntu_14.04. Avant de démarrer les VM, vérifiez que chaque VM n'ait qu'une seule interface, sinon modifiez. Augmentez la mémoire vidéo à 30MB si elle est inférieure et cochez *Activer l'accélération 3D*. Idem pour Système→ Acceleration→ *Pagination*.
- Ajout des VM dans GNS3 : Edit → Preferences → VirtualBox → onglet guest → refresh list, sélectionnez le nom de la VM serveur, mettez-lui un ID name à « Serveur », décochez toutes les cases (dont Reserve first NIC...) puis Save. Faire de même pour la VM Client.
- Le routeur à utiliser est un Cisco c7200.
- Rajouter un slot de Port Adapter (PA) PA-4E. Cela ajoute 4 interfaces Ethernet 10BaseT, permettant donc un maximum de 10Mbps par interface. Attention : ne pas mettre de FastEthernet.
- Configurez les interfaces du routeur. Les interfaces et réseaux à utiliser sont indiqués en Fig. 1.
- Les interfaces des VM doivent déjà être configurées, vérifiez-le. Tous les logins/passwords sont rt/rt.

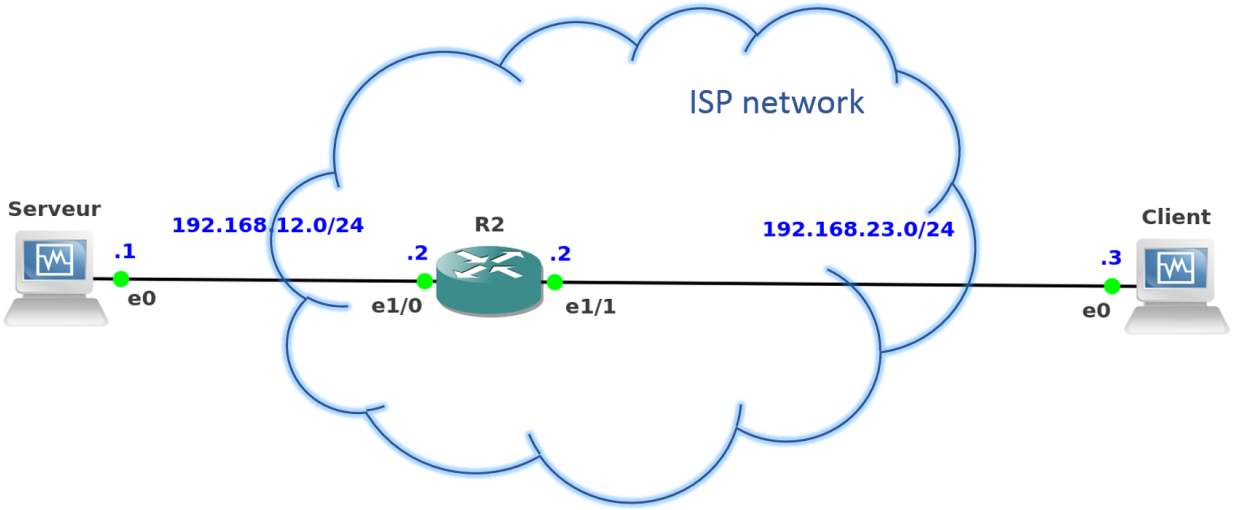


Illustration 1: Réseau considéré dans l'ensemble du TP

- Tester un ping entre les 2 VM. Si problème, vous debuggez.
- Tester accès page web : depuis VM droite, http://192.168.12.1/index.html
- Tester accès ssh : depuis VM droite dans un terminal, `ssh rt@192.168.12.1`

I.2. HTTP video streaming avec le player TAPAS : principe et scripts à utiliser

Les deux VMs sont des Ubuntu 14.04. La VM serveur est extrêmement simple, supportant uniquement un serveur SSH (Open SSH) et un serveur Web (Apache 2). Dans le répertoire du serveur Web sont stockés les fichiers vidéo (.mp4) ainsi que le fichier de manifest (.mpd) décrit ci-dessous.

I.2.1. Rappel sur le streaming vidéo adaptatif sur HTTP

- Re-lisez les slides 113 à 115 de votre cours.

- En streaming adaptatif sur HTTP, la même vidéo est disponible en plusieurs qualités au serveur : regardez dans le répertoire /var/www/html/ du serveur et listez ci-dessous les fichiers.

- La liste des .mp4 avec leurs URL, leurs résolutions (*height/width*) et les débits d'encodage (*Representation bandwidth*) de la vidéo sont listés dans le fichier de type XML nommé manifest.mpd. Ouvrez-le, constatez et observez en particulier les débits d'encodage.

- Le client va demander la vidéo par segments de D secondes. Ici, D=5s. Le client va d'abord remplir son buffer de lecture en demandant à la suite plusieurs segments, commencera la lecture quand le seuil `buffer_min` sera atteint, puis arrêtera de télécharger quand le seuil `buffer_max` sera atteint, et reprendra quand le nombre de secondes de vidéo en buffer deviendra inférieur à `buffer_min`. Vous déterminerez ces valeurs par l'expérience dans la suite.

- A chaque requête, le client demande le segment sous la forme d'une HTTP range-request, c'est-à-dire un morceau de l'objet HTTP, correspondant aux D secondes désirées. L'objet HTTP est le .mp4 sélectionné avec la qualité adaptée à l'état actuel du réseau : le client choisit le débit d'encodage (= nb de bits lus pendant 1s) maximum inférieur à la bande passante courante estimée (=nb de bits téléchargeables en 1s).

I.2.2. TAPAS : une application pour streaming video adaptatif sur HTTP

Pour étudier les performances d'un flux vidéo streamé, il faut avoir un vrai lecteur qu'on puisse instrumentaliser, c'est-à-dire dont on puisse modifier le code pour enregistrer tout ce qui se passe.

- Ceci correspond à créer des fichier de log. Un tel fichier est créé à chaque lecture : allez dans le client dans /home/rt/tapas-master/logs/conventional/ et ouvrez avec LibreOffice le fichier de log le plus récent.

- Observez attentivement la signification des colonnes et leurs valeurs.
Col. A : temps absolu, Col. C : nb de secondes de vidéo dans le buffer de lecture, Col. D : bw estimée par le client, Col. F : qualité demandée (0 à 5), Col. E : débit d'encodage de cette qualité, Col. R : byte range demandé

- A partir de ces fichiers de logs, nous ferons des tracés pour faciliter l'analyse.

Le player que nous utilisons a été créé par des chercheurs de Bari, et se nomme TAPAS [1]. Il est écrit en python.

[1] Luca De Cicco, Vito Caldaralo, Vittorio Palmisano, Saverio Mascolo, "TAPAS: a Tool for rApid Prototyping of Adaptive Streaming algorithms", in Proc. of ACM VideoNext Workshop, Sydney, Australia, December 2014.

I.2.3. Commandes et scripts à utiliser

Dans la suite nous utiliserons ces quelques commandes et scripts (toutes sur le client). Référez-vous à cette section dans la suite.

- C1 : Téléchargement unique de fichier avec (ne pas oublier . à la fin). A faire depuis ~/Downloads/

```
scp rt@192.168.12.1:/var/www/html/car*89.mp4 .
```

- C2 : Téléchargements parallèles alterné de 3 fichiers avec un script bash à ouvrir maintenant et à analyser (remarquez les intervalles de temps où les scp sont actifs ou inactifs). A faire depuis ~/Downloads/

```
bash launch3scps.sh
```

- C3 : Lecture en streaming de la vidéo (sans affichage). A faire depuis ~/tapas-master

```
DEBUG=2 python play.py -m nodec -u http://192.168.12.1/car-20120827-manifest.mpd
```

- C4 : Affichage des métriques relatives au player. Copiez le nom du dernier fichier de log (file browser en bas à droite, avec Rename puis Ctrl C) et copiez-le en remplacement du nom sur la ligne 1 dans *plot_metrics.sh*. A faire depuis ~/Downloads/. Ensuite exécutez le script :

```
bash plot_metrics.sh
```

- C5 : Lecture de la vidéo telle que streamée. A faire depuis ~/Downloads/

```
bash play_from_log.sh
```

I.3. QoS en Cisco

Plusieurs implémentations des mécanismes de QoS sont disponibles en Cisco, et sont décrites brièvement en Annexe 1 en fin de sujet. Ici, nous allons utiliser :
MQC = Modular QoS CLI (Command Line Interface). *Modular* car construction par blocs adaptables, tel que représenté Fig. 2.

Class-Map : commande pour créer des classes de trafic (selon application, entête IP, etc.)

Policy-Map :

- définit ce que vous voulez faire avec chaque classe de trafic
- politique applicable aux interfaces choisies

Pour le cas de ce TP, les classes et politiques seront telles que représentées Fig. 3. Attention : les classes seront à définir au fur et à mesure tel qu'indiqué plus tard dans le sujet, et la politique *regulation* sera changée plusieurs fois par rapport à la figure.

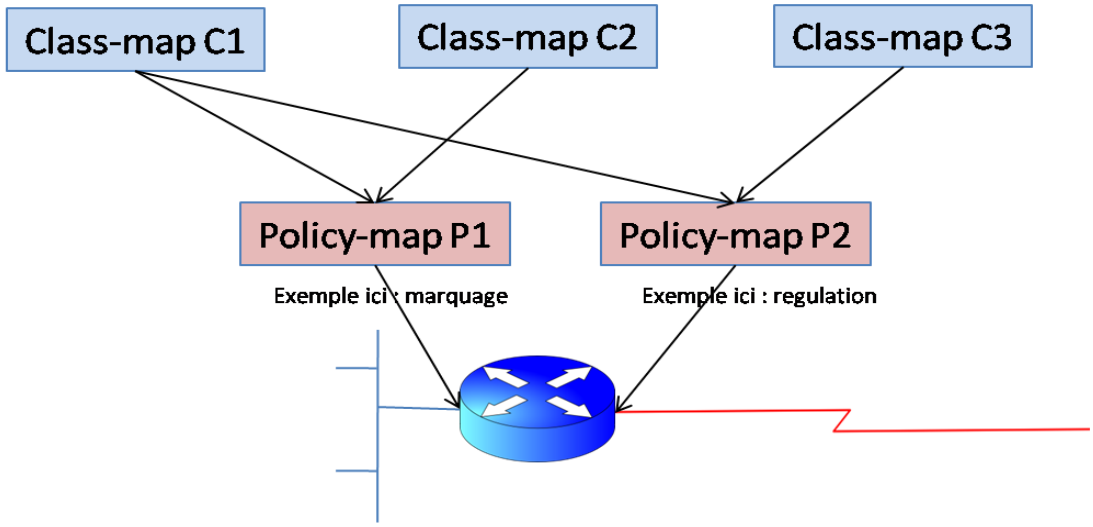


Illustration 2: Modules de Cisco MQC : des classes sont définies sur des critères variées du trafic d'entrée à classer, des politiques sont créées pour traiter certaines classes différemment sur chaque interface.

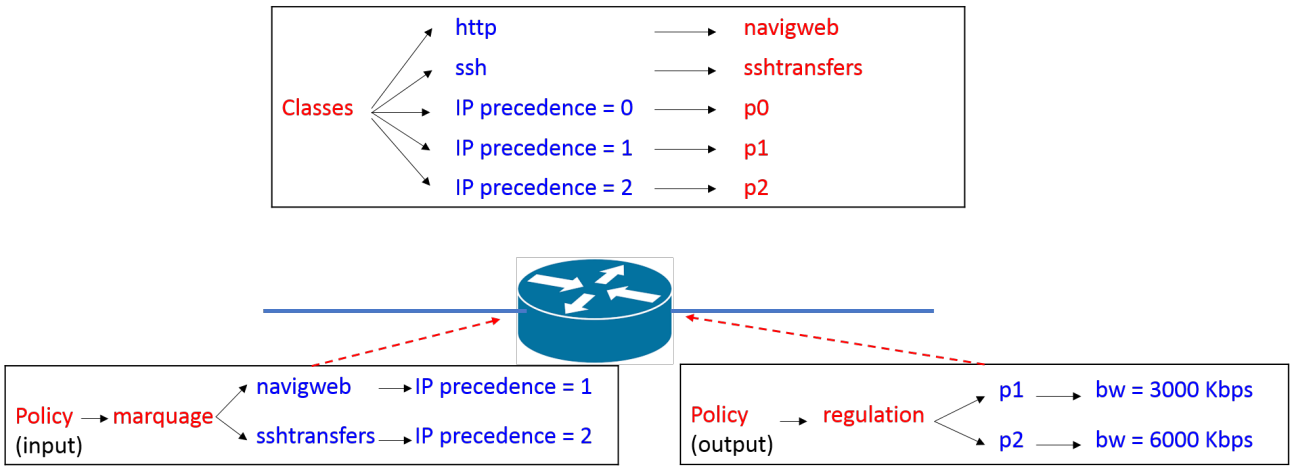


Illustration 3: Classes et politiques créées pendant le TP. Rouge: nom des classes et politiques, bleu: paramètres utilisés pour les définir.

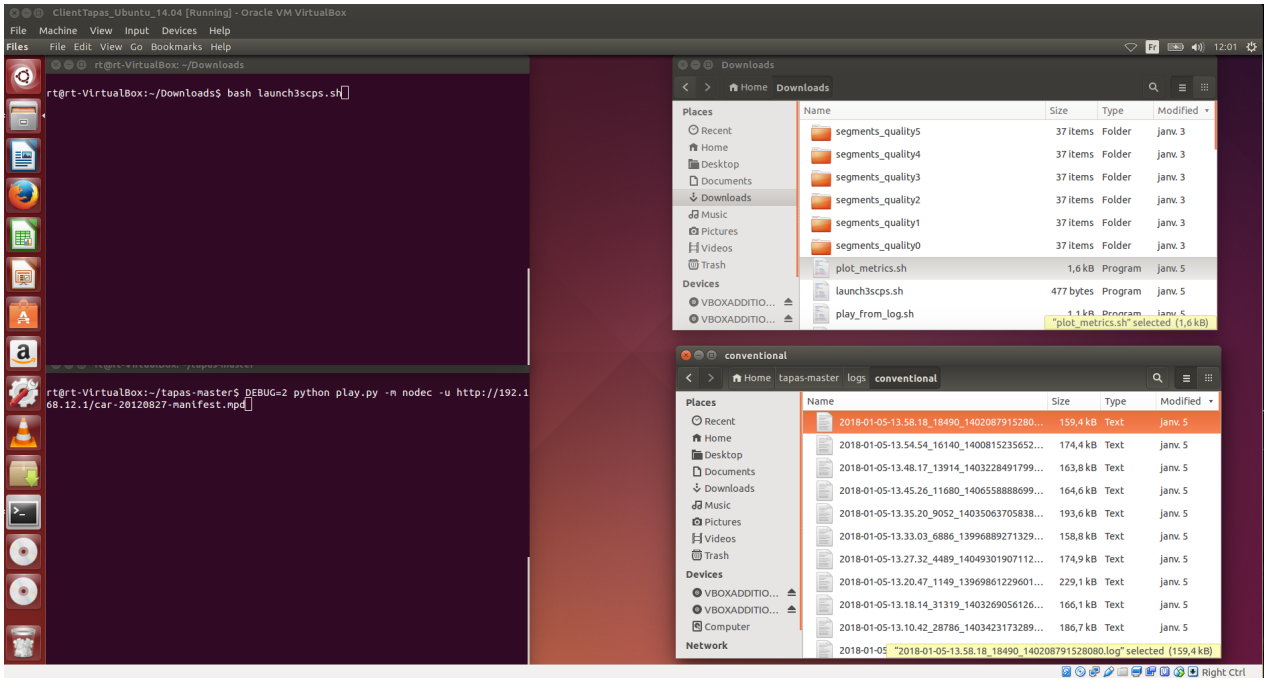


Illustration 4: Organisation de l'espace de travail sur VM client.

II. Performance des flux sans action spécifique du routeur

Nous analysons d'abord la performance (en débit, délai, qualité vidéo, durée d'interruption de lecture) des flux SSH de transfert de fichier, et vidéo, quand ces deux applications sont utilisées par le client en même temps.
Le routeur ne fait aucune action QoS spécifique, c'est-à-dire que chaque port de sortie correspond à une unique file d'attente FIFO gérée en droptail. Les paquets des différents flux sont tous traités de la même façon.

II.1. Flux de transferts de fichiers seuls

Q1 : Effectuez un transfert scp (commande C1 en I.2.3). Noter le débit moyen obtenu et le temps de transfert.
Q2 : Faîtes de même avec 2 transferts lancés en parallèle. Noter le débit moyen obtenu et le temps de transfert de chacun. Que pouvez-vous dire ? Quel est le protocole responsable de ce partage ?

Q3 : Lancez 3 transferts en parallèle (commande C2). Que pouvez-vous dire du débit obtenu ?

II.2. Flux vidéo seul

Agencez les différentes fenêtres comme indiqué Fig. 4 pour vous faciliter la tâche, et ouvrez le script *plot_metrics.sh*.

Q1 : Lancer le streaming (commande C3, dans le terminal en bas à gauche). L'affichage est désactivé. Arrêtez le streaming après 20 ou 30 sec.

Q2 : Tracez les métriques avec C4. Copiez la figure obtenue ci-dessous et commentez toutes les métriques tracées indiquées dans la légende, aussi en détail que possible. Quelle sont les valeurs de `buffer_min` et `buffer_max` mentionnées en Sec. I.2.1 ?

II.3. Concurrence entre vidéo et transferts de fichiers

Q1 : Lancez 3 scp en parallèle avec commande C2 dans terminal en haut à gauche, et simultanément relancez le streaming avec C3 dans le terminal en bas à gauche. Arrêtez le streaming 10 s après que la commande C2 se soit terminée seule.

Q2 : Tracez les métriques comme en Q2 ci-dessus. Comment varie la qualité de la vidéo, et le temps vidéo bufferisé, en fonction des scp (rouvrir *launch3scps.sh* pour interpréter) ?

Q3 : Visionnez (avec C5) la portion de vidéo que vous avez streamée, et constatez que les variations de qualité correspondent bien à celle représentées sur la figure ci-dessus.

III. Performance des flux avec traitements de QoS au routeur

Nous allons créer 5 classes comme représenté Fig. 2. Il est conseillé de vous référer à cette figure autant que nécessaire. Nous allons créer 2 politiques : la politique *marquage* pour les paquets entrant par e1/0 (depuis serveur) et la politique *regulation* pour les paquets sortant par e1/1 (vers client).

III.1. Création des classes navigweb, sshtransfers

Q1 : On crée une classe :

```
conf t
class-map ?
```

Vous voyez 3 possibilités :

- *match-all* : tous les critères qui vont être mentionnés ensuite doivent être vérifiés par le paquet pour qu'il appartienne à la classe

- *match-any* : au moins un des critères qui vont être mentionnés ensuite doit être vérifié par le paquet pour qu'il appartienne à la classe

- *WORD* : nom de la classe et prend par défaut match-all

Tapez :

```
class-map match-any navigweb
```

navigweb est le nom choisi pour la classe ici.

Q2 : On rentre alors automatiquement dans la configuration de la classe, et on regarde les commandes possibles :

```
(config-cmap)# ?
```

Match : on veut matcher (satisfaire en français) certains critères

```
match ?
```

Critères : IP address, application type, access-list, …

L'annexe 2 discute les différentes possibilités pour définir des classes et la parade au port tunneling.

Q3 : Pour définir la classe *navigweb*, choisissons les protocoles http et https:

```
(config)#class-map match-any navigweb
(config-cmap)#match protocol http
(config-cmap)#exit
```

Ce sera donc dans cette classe que sera mis le flux vidéo.

Q4 : Créez une autre classe, nommée *sshtransfers*, définie par le protocol ssh. Indiquez la suite de commandes utilisées.

III.2. Création de la politique marquage pour e1/0

Tout ce qui précède est uniquement pour classifier le trafic : identifier le plus correctement possible dans quel groupe chaque paquet tombe selon les critères qu'on désire.

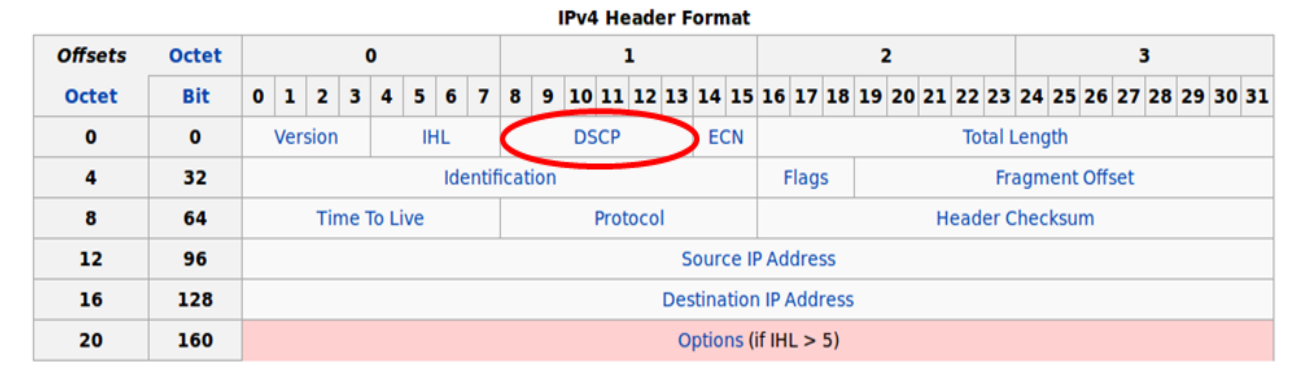
A présent il faut marquer le trafic. Rappel : marquer = attribuer une valeur au champ QoS, à la couche 2 (L2) ou à la couche 3 (L3).

3 façons :

- marquer CoS : layer 2 marking, pour que les switches appliquant politiques de QoS

- marquer IP Prec (precedence=priorité en français) : layer 3. Dans champ ToS. 8 niveaux, mais 6 vraiment (2 pour contrôle, ex : protocoles de routage)

- marquer DSCP : layer 3. Dans champ ToS. Façon actuelle. 12 niveaux.



Q1 : Vérifier les classes configurées jusqu'ici :

```
#show class-map
```

Q2 : On passe dans le bloc `policy-map` :

```
(config)#policy-map ?
```

et on donne un nom

```
(config)#policy-map marquage
(config-pmap)# ?
```

`class` va désigner la classe de trafic à laquelle on va appliquer la politique qu'on est en train de définir :

```
(config-pmap)# class navigweb
```

Donc on est en train de dire : dans cette politique *marquage*, pour cette classe *navigweb*, voici ce qu'on va faire. Regarder les options qu'on a alors :

```
(config-pmap-c)# ?
```

Indiquez ci-dessous ce qui vous est affiché et qui correspond à allouer du débit.

Q3 : Il y a l'option *set* :

```
(config-pmap-c)#set ?
```

On peut donc fixer ici la valeur du champ (=marquer) DSCP, IP precedence ou CoS. Pour le web, on va mettre *precedence* (« priorité ») à 1 :

```
(config-pmap-c)#set precedence 1
```

Q4 : Pour la classe *sshtransfers*, définir dans la politique *marquage* la priorité (*precedence*) 2.

Indiquez les commandes utilisées et recopiez le résultat de `show policy-map`.

Vous voyez donc (Fig. 1) que, dans le système Cisco MQC, une même politique peut être appliquée à plusieurs classes, et qu'une même classe peut subir plusieurs politiques.

Q5 : L'étape suivante est d'appliquer la politique à l'interface désirée. Ici l'ISP va marquer le trafic reçu depuis serveur. On va donc appliquer cette politique sur le trafic entrant dans e1/0 du PE :

```
(config)#interface e1/0
(config-if)#service-policy input marquage
```

Q6 : On a donc réalisé les 2 étapes de configuration de QoS sur un routeur : classifier le trafic en classes, et définir une politique (ici, *marquage*), puis l'associer à l'interface désirée pour le sens désiré(trafic entrant et/ou sortant).

On peut regarder les traitements effectivement réalisés sur une interface :

```
#show policy-map interface e1/0
```

Qu'est-ce que l'ISP peut y voir ?

III.3. Politique regulation sur e1/1 : impact de CBWFQ et priorité stricte

On vient de créer une politique *marquage* appliquée au trafic entrant dans e1/0. On va maintenant appliquer une politique *regulation* au trafic sortant de e1/1.

Q1 : Comme le routeur peut recevoir du trafic non marqué et déjà marqué, on va créer 3 classes, discriminant le trafic sur le niveau de precedence du paquet entrant:

```
(config)#class-map p0
(config-cmap)#match ip precedence 0
```

et faire pareil pour 1 et 2. On aura donc le flux vidéo dans la classe p1 et les ssh dans p2.

III.3.1. Video bandwidth 6000 / SSH bandwidth 3000

Q1 : On crée la politique *regulation* qui va allouer un minimum de bande passante. Cela signifie que les flux seront orientés dans des files physiques différentes en fonction de leurs classes, et ces files sont gérées en CBWFQ (cf. cours slide 139).

```
(config)#policy-map regulation
(config-pmap)#class p1
(config-pmap-c)# ?
```

On va assigner une certaine bw à cette classe.

```
(config-pmap-c)#bandwidth 6000
```

Ce qui signifie qu'on alloue 6000 Kbps au trafic de la classe p1. Ceci ne signifie pas que le trafic de p1 va être limité à 6 Mbps, il pourra avoir plus s'il n'y a pas de congestion, mais qu'en cas de congestion il sera envoyé au moins à 6 Mbps.

Faire de même en affectant 3Mbps à p2.

Q2 : Appliquer cette politique *regulation* sur le trafic sortant de e1/1. Vérifiez et indiquez ce que vous obtenez par

```
show policy-map interface e1/1
```

Q3 : Effectuez de nouveau les transferts vidéo et 3 ssh parallèles, et tracez les performances de la vidéo, comme indiqué en Q1 et Q2 de Sec. II.3. Commentez le résultat.

III.3.2. Video bandwidth 3000 / SSH bandwidth 6000

Q1 : Changer les bw affectées à chaque classe p1 et p2 aux valeurs ci-dessus. Enlever au préalable les anciennes avec `no bandwidth 6000`, etc.

Q2 : Effectuez de nouveau les transferts vidéo et 3 ssh parallèles, et tracez les performances de la vidéo, comme indiqué en Q1 et Q2 de Sec. II.3. Commentez le résultat.

III.3.3. Video priority 3000 / SSH bandwidth 6000

Au lieu de gérer la file de la classe p1 en CBWFQ pour lui assurer une bw minimum, on va maintenant la mettre en priorité stricte (cf. cours slide 137).

Q1 : Remplacez l'allocation de bw à 3000 Kbps par une priorité à 3000 Kbps :

```
(config)#policy-map regulation
(config-pmap)#class p1
(config-pmap-c)# priority 3000
```

Vérifiez que cette nouvelle configuration a bien été prise en compte avec

```
#show policy-map interface e1/1
```

Q2 : Effectuez de nouveau les transferts vidéo et 3 ssh parallèles, et tracez les performances de la vidéo, comme indiqué en Q1 et Q2 de Sec. II.3.

Q3 : Commentez le résultat en utilisant le tableau suivant pris de la page décrivant *priority queuing* sur le site de Cisco.

Function	bandwidth Command	priority Command
Minimum bandwidth guarantee	Yes	Yes
Maximum bandwidth guarantee	No	Yes
Built-in policer	No	Yes
Provides low latency	No	Yes

III.4. Traitement avec WRED

On va utiliser Weighted Random Early Detection pour améliorer la QoS d'une classe. Au lieu d'utiliser des files physiques différentes, on va donc appliquer des probabilités d'abandon différentes aux paquets des différentes classes tous stockés dans la même file de sortie, toujours en FIFO mais plus en droptail (en WRED donc).

Q1 : Désactiver la politique regulation en la désaffectant de e1/1 :

```
(config)#interface e1/1
(config-if)#no service-policy output regulation
```

Q2 : Configurez des seuils d'abandon plus élevés pour p1 que p2 comme suit :

```
(config)#interface e1/1
```

```
(config-if)# random-detect
(config-if)# random-detect precedence 1 100 200
(config-if)# random-detect precedence 2 20 100
```

Q3 : Effectuez de nouveau les transferts vidéo et 3 ssh parallèles, et tracez les performances de la vidéo, comme indiqué en Q1 et Q2 de Sec. II.3.

Q4 : Commentez le résultat en faisant le lien avec la politique WRED que vous avez configurée.

Sources

[1] Luca De Cicco, Vito Caldaralo, Vittorio Palmisano, Saverio Mascolo, "TAPAS: a Tool for rApid Prototyping of Adaptive Streaming algorithms", in Proc. of ACM VideoNext Workshop, Sydney, Australia, December 2014.

[2] *QoS fundamentals and MQC*, Demos de Nomi Beo, online

Annexe 1

Implémentations disponibles des mécanismes de QoS en Cisco:

- CLI : définition par interface de stratégies de QoS. Pas scalable et pas facile à contrôler.
- AutoQoS : Cisco l'a sorti en 2002-2003. Très simple avec cette commande par interface. Assure que des politiques cohérentes sont appliquées sur tous les équipements (template de QoS automatique). Regroupe des commandes MQC.
- QoS Policy Manager (QPM) : pour implémentation à l'échelle d'un réseau d'ISP de politiques choisies de QoS
- MQC = Modular QoS CLI (Command Line Interface). Ce qu'on va utiliser. *Modular* car construction par blocs adaptables.

Annexe 2

Détails sur la définitions des classes et la parade au port tunneling :

access-group : pointe sur access-list définie (par port, par IP@, etc.)

cos : class of service , marquage couche 2 (par exemple 802.1p). Donc on peut avoir un switch qui peut marquer au niveau 2, et le routeur re-marquer au niveau 3

Grosse liste : destination-address, FR mapping, etc

protocol : un firewall basique filtre sur port number, mais pas efficace car appli qui veulent les passer font du tunneling de port, comme applis P2P (BitTorrent à travers port 80). La solution :

```
match protocol ?
```

On voit des protocoles client-serveur classiques (http, ipsec, ssh, etc.) et plusieurs protocoles P2P (BitTorrent, Edonkey – pour emule, etc.).

La parade de Cisco aux applications utilisant le port tunneling s'appelle NBAR – *Network Based Application Recognition*. Il s'agit de faire du *Deep Packet Inspection*, jusqu'au payload de couche application. NBAR reconnaît les applications par des caractéristiques uniquement du payload.

Et si les applis changent de version ?

```
Exit
(config)#ip nbar ?
```

Pdlm est un fichier maintenu par Cisco qui décrit les caractéristiques à jour des applis pour nbar, à charger dans la flash du router (ne pas entrer cette commande)

```
(config)#ip nbar pdlm <chemin>
```