Présentation de la mise en file d'attente pondérée basée sur les classes sur les interfaces ATM

Contenu

Introduction

Ce document présente la mise en file d'attente du trafic à l'aide de la technologie CBWFQ (Weighted Fair Queuing) basée sur les classes.

La mise en file d’attente pondérée (WFQ) permet aux liaisons à faible débit, telles que les liaisons série, de fournir un traitement équitable pour tous les types de trafic. Il classe le trafic en différents flux (également appelés conversations) en fonction des informations des couches 3 et 4, telles que les adresses IP et les ports TCP. Il le fait sans que vous ayez à définir des listes d’accès. Cela signifie que le trafic à faible bande passante a effectivement priorité sur le trafic à bande passante élevée, car le trafic à bande passante élevée partage le support de transmission en

proportion de sa masse attribuée. Cependant, WFQ a certaines limites :

- Il n'est pas évolutif si le débit augmente considérablement.
- La norme WFQ native n'est pas disponible sur les interfaces haut débit telles que les interfaces ATM.

CBWFQ offre une solution à ces limitations. Contrairement à la WFQ standard, CBWFQ vous permet de définir des classes de trafic et d'appliquer des paramètres, tels que la bande passante et les limites de file d'attente, à ces classes. La bande passante que vous affectez à une classe est utilisée pour calculer le « poids » de cette classe. Le poids de chaque paquet qui correspond aux critères de classe est également calculé à partir de cette valeur. WFQ est appliqué aux classes (qui peuvent inclure plusieurs flux) plutôt qu'aux flux eux-mêmes.

Pour plus d'informations sur la configuration de CBWFQ, cliquez sur les liens suivants :

Mise en file d'attente pondérée par classe par circuit virtuel (CBWFQ par circuit virtuel) sur les routeurs Cisco 7200, 3600 et 2600.

Mise en file d'attente pondérée par classe par circuit virtuel sur des plates-formes RSP.

Avant de commencer

Conventions

Pour plus d'informations sur les conventions des documents, référez-vous aux Conventions utilisées pour les conseils techniques de Cisco.

Conditions préalables

Aucune condition préalable spécifique n'est requise pour ce document.

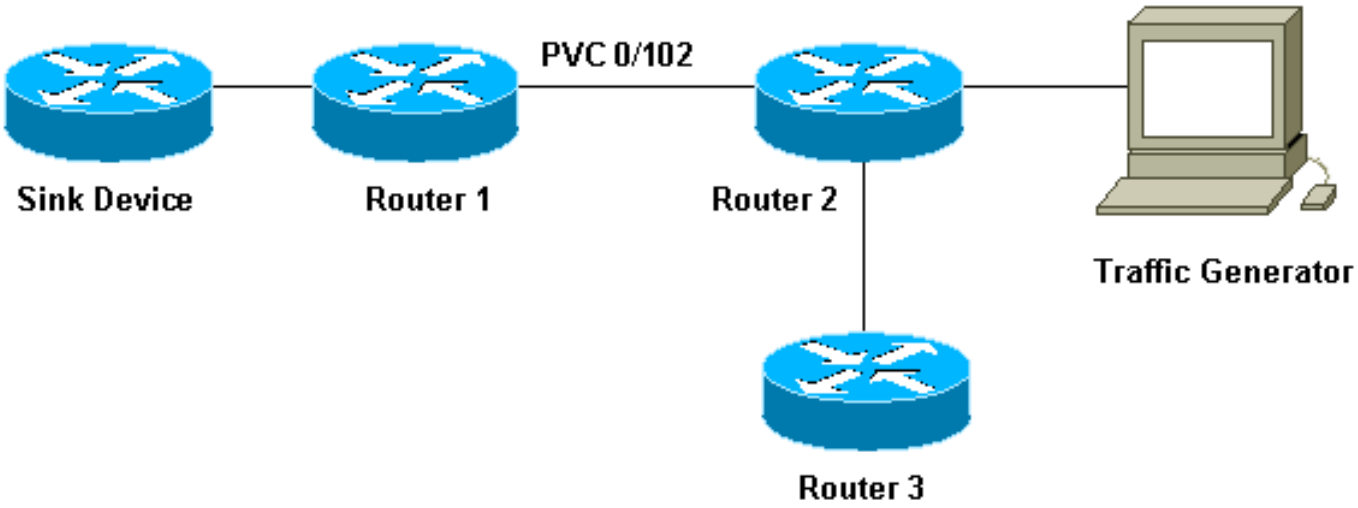
Components Used

Ce document n'est pas limité à des versions de matériel et de logiciel spécifiques.

Les informations présentées dans ce document ont été créées à partir de périphériques dans un environnement de laboratoire spécifique. All of the devices used in this document started with a cleared (default) configuration. Si vous travaillez dans un réseau opérationnel, assurez-vous de bien comprendre l'impact potentiel de toute commande avant de l'utiliser.

Diagramme du réseau

Afin d'illustrer le fonctionnement de WFQ, utilisons la configuration suivante :



Dans la configuration que nous utilisons ici, les paquets peuvent être stockés dans l'une des deux files d'attente suivantes :

- La file d'attente FIFO (matériel First In First Out) de la carte de ports et du module réseau.
- File d'attente du logiciel Cisco IOS® (sur la mémoire d'entrée/sortie [E/S] du routeur) dans laquelle des fonctionnalités de qualité de service (QoS) telles que CBWFQ peuvent être appliquées.

La file d'attente FIFO de la carte de port stocke les paquets avant qu'ils ne soient segmentés en cellules pour transmission. Lorsque cette file d'attente est pleine, la carte de ports ou le module réseau signale au logiciel IOS que la file d'attente est congestionnée. Ce mécanisme est appelé contre-pression. À la réception de ce signal, le routeur arrête d'envoyer des paquets à la file d'attente FIFO de l'interface et stocke les paquets dans le logiciel IOS jusqu'à ce que la file d'attente soit à nouveau décongestionnée. Lorsque les paquets sont stockés dans IOS, le système peut appliquer des fonctions QoS telles que CBWFQ.

Définition de la limite de sonnerie de transmission

Un problème avec ce mécanisme de mise en file d'attente est que plus la file d'attente FIFO sur l'interface est grande, plus le délai avant que les paquets à la fin de cette file d'attente ne puissent être transmis est long. Cela peut entraîner de graves problèmes de performances pour le trafic sensible aux retards, tel que le trafic vocal.

La commande permanent virtual circuit (PVC) **tx-ring-limit** vous permet de réduire la taille de la file d'attente FIFO.

```
interface ATMx/y.z point-to-point
      ip address a.b.c.d M.M.M.M
      PVC A/B
       tx-ring-limit
       service-policy output test
```

La limite (x) que vous pouvez spécifier ici est un nombre de paquets (pour les routeurs Cisco 2600 et 3600) ou une quantité de particules (pour les routeurs Cisco 7200 et 7500).

La réduction de la taille de l'anneau de transmission présente deux avantages :

- Il réduit le temps d'attente des paquets dans la file d'attente FIFO avant d'être segmenté.
- Il accélère l'utilisation de la QoS dans le logiciel IOS.

Impact de la limite de sonnerie de transmission

Examinons l'impact de la limite de l'anneau de transmission à l'aide de la configuration indiquée dans notre schéma de réseau ci-dessus. Nous pouvons supposer ce qui suit :

- Le générateur de trafic envoie du trafic (paquets de 1 500 octets) au périphérique récepteur, et ce trafic surcharge le circuit virtuel permanent 0/102 entre le routeur 1 et le routeur 2.
- Le routeur 3 tente d’envoyer une requête ping au routeur 1.
- CBWFQ est activé sur le routeur 2.

Maintenant, regardons deux configurations en utilisant des limites d'anneau de transmission différentes afin de voir l'impact que cela a.

Exemple A

Dans cet exemple, nous avons défini la sonnerie de transmission sur trois (**TX-ring-limit=3**). Voici ce que nous voyons lorsque nous envoyons une requête ping à router1 à partir de router3.

```
POUND#ping ip
     Target IP address: 6.6.6.6
     Repeat count [5]: 100000
     Datagram size [100]:
     Timeout in seconds [2]:
     Extended commands [n]:
     Sweep range of sizes [n]:
     Type escape sequence to abort.
     Sending 100000, 100-byte ICMP Echos to 6.6.6.6, timeout is 2 seconds:
     !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
     !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
     !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
     !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
     !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
     !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
     !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
     !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!![snip]
     Success rate is 98 percent (604/613), round-trip min/avg/max = 164/190/232 ms
```

Exemple B

Dans cet exemple, nous avons défini la sonnerie de transmission sur 40 (**TX-ring-limit=40**). Voici ce que nous voyons lorsque nous utilisons la même requête ping que dans l’exemple A :

```
POUND#ping ip
     Target IP address: 6.6.6.6
     Repeat count [5]: 10000
     Datagram size [100]: 36
     Timeout in seconds [2]: 10
     Extended commands [n]:
     Sweep range of sizes [n]:
     Type escape sequence to abort.
     Sending 10000, 36-byte ICMP Echos to 6.6.6.6, timeout is 10 seconds:
     !!!!!!!!!!!!.
```

```
Success rate is 92 percent (12/13), round-trip min/avg/max = 6028/6350/6488
```

Comme vous pouvez le voir ici, plus la limite de la sonnerie de transmission est grande, plus le temps de transmission de la requête ping (RTT) est grand. Nous pouvons en déduire qu'une grande limite de cycle de transmission peut entraîner des retards importants dans la transmission.

Fonctionnement de CBWFQ

Maintenant que nous avons vu l'impact de la taille de la file d'attente FIFO matérielle, voyons exactement comment CBWFQ fonctionne.

La WFQ native affecte un poids à chaque conversation, puis planifie le temps de transmission pour chaque paquet des différents flux. Le poids est fonction de la priorité IP de chaque flux et le temps de planification dépend de la taille du paquet. Cliquez ici pour plus de détails sur WFQ.

CBWFQ attribue un poids à chaque classe configurée au lieu de chaque flux. Ce poids est proportionnel à la bande passante configurée pour chaque classe. Plus précisément, le poids est fonction de la bande passante de l'interface divisée par la bande passante de la classe. Par conséquent, plus le paramètre de bande passante est grand, plus le poids est faible.

Nous pouvons calculer le temps de planification des paquets à l'aide de la formule suivante :

`scheduling tail_time= queue_tail_time + pktsize * weight`

Division de la bande passante de l'interface totale

Examinons comment le routeur répartit la bande passante totale de l'interface entre les différentes classes. Afin de servir les classes, le routeur utilise des files d'attente de calendrier. Chacune de ces files d'attente de calendrier stocke les paquets qui doivent être transmis au même time_queue_de_planification. Le routeur gère ensuite ces files d'attente de calendrier une par une. Examinons ce processus :

1. Si un encombrement survient sur la carte de port lorsqu'un paquet arrive sur l'interface de sortie, cela entraîne la mise en file d'attente dans IOS (CBWFQ dans ce cas).
2. Le routeur calcule une heure de planification pour ce paquet entrant et la stocke dans la file d'attente du calendrier correspondant à cette heure de planification. Un seul paquet par classe peut être stocké dans une file d'attente de calendrier spécifique.
3. Lorsqu'il est temps de traiter la file d'attente du calendrier dans laquelle le paquet a été stocké, l'IOS vide cette file d'attente et envoie les paquets à la file d'attente FIFO sur la carte de port elle-même. La taille de cette file d'attente FIFO est déterminée par la limite de l'anneau de transmission décrite ci-dessus.
4. Si la file d'attente FIFO est trop petite pour tenir compte de tous les paquets qui sont conservés dans la file d'attente du calendrier traité, le routeur replanifie les paquets qui ne peuvent pas être stockés pour la prochaine période de planification (correspondant à leur poids) et les place dans la file d'attente du calendrier correspondant.
5. Lorsque tout cela est fait, la carte de port traite les paquets dans sa file d'attente FIFO et envoie les cellules sur le câble et l'IOS se déplace vers la file d'attente de calendrier suivante.Grâce à ce mécanisme, chaque classe reçoit statistiquement une partie de la bande passante de l'interface correspondant aux paramètres configurés pour elle.

Mécanisme de file d'attente du calendrier par rapport à la taille de la sonnerie de transmission

Examinons la relation entre le mécanisme de la file d'attente du calendrier et la taille de l'anneau de transmission. Un petit anneau de transmission permet à la QoS de démarrer plus rapidement et réduit la latence des paquets en attente de transmission (ce qui est important pour le trafic sensible aux retards, comme la voix). Cependant, si elle est trop petite, elle peut entraîner un débit inférieur pour certaines classes. En effet, il se peut que beaucoup de paquets doivent être reprogrammés si l'anneau de transmission ne peut pas les accueillir.

Malheureusement, il n'existe aucune valeur idéale pour la taille de l'anneau de transmission et la seule façon de trouver la meilleure valeur est d'expérimenter.

Partage de bande passante

Nous pouvons examiner le concept de partage de bande passante à l'aide de la configuration présentée dans notre diagramme de réseau, ci-dessus. Le générateur de paquets produit différents flux et les envoie au périphérique d'égout. La quantité totale de trafic représentée par ces flux est suffisante pour surcharger le circuit virtuel permanent. Nous avons mis en oeuvre CBWFQ sur Router2. Voici à quoi ressemble notre configuration :

```
access-list 101 permit ip host 7.0.0.200 any
   access-list 101 permit ip host 7.0.0.201 any
   access-list 102 permit ip host 7.0.0.1 any
   !
   class-map small
     match access-group 101
   class-map big
     match access-group 102
   !
   policy-map test
   policy-map test
     small class
      bandwidth <x>
     big class
      bandwidth <y>
    interface atm 4/0.102
     pvc 0/102
       TX-ring-limit 3
       service-policy output test
       vbr-nrt 64000 64000
```

Dans notre exemple, Router2 est un routeur Cisco 7200. Ceci est important car la limite de l'anneau de transmission est exprimée en particules et non en paquets. Les paquets sont mis en file d'attente FIFO de la carte de port dès qu'une particule libre est disponible, même si plusieurs particules sont nécessaires pour stocker le paquet.

Qu'est-ce qu'une particule ?

Au lieu d'allouer une partie de mémoire contiguë à une mémoire tampon, la mise en mémoire tampon des particules alloue des morceaux de mémoire non contigus (éparpillés), appelés particules, puis les relie ensemble pour former une seule mémoire tampon de paquets logique. C'est ce qu'on appelle un tampon de particules. Dans un tel schéma, un paquet peut alors être réparti sur plusieurs particules.

Dans le routeur 7200 que nous utilisons ici, la taille des particules est de 512 octets.

Nous pouvons vérifier si les routeurs Cisco 7200 utilisent des particules à l'aide de la commande **show buffers** :

```
router2#show buffers
     [snip]
     Private particle pools:
     FastEthernet0/0 buffers, 512 bytes (total 400, permanent 400):
          0 in free list (0 min, 400 max allowed)
          400 hits, 0 fallbacks
          400 max cache size, 271 in cache
     ATM4/0 buffers, 512 bytes (total 400, permanent 400):
          0 in free list (0 min, 400 max allowed)
          400 hits, 0 fallbacks
          400 max cache size, 0 in cache
```

Test A

Les classes « Small » et « Big » que nous utilisons pour ce test sont remplies de la manière suivante :

- Petite classe : nous avons configuré les paramètres de bande passante à 32 kbits/s. Cette classe stocke dix paquets de 1 500 octets à partir de 7.0.0.200, suivis de dix paquets de 1 500 octets à partir de 7.0.0.201
- Grande classe : nous avons configuré le paramètre de bande passante à 16 kbits/s. Cette classe stocke un flux de dix paquets de 1 500 octets de 7.0.0.1.

Le générateur de trafic envoie une rafale de trafic destinée au périphérique récepteur à 100 Mbits/s au routeur 2 dans l'ordre suivant :

1. Dix paquets de 7.0.0.1.
2. Dix paquets de 7.0.0.200.
3. Dix paquets de 7.0.0.201.

Vérification du poids du flux

Regardons le poids appliqué aux différents flux. Pour ce faire, nous pouvons utiliser la commande **show queue ATM x/y.z**.

```
alcazaba#show queue ATM 4/0.102
     Interface ATM4/0.102 VC 0/102
     Queueing strategy: weighted fair
     Total output drops per VC: 0
     Output queue: 9/512/64/0 (size/max total/threshold/drops)
        Conversations  2/3/16 (active/max active/max total)
        Reserved Conversations 2/2 (allocated/max allocated)

  (depth/weight/total drops/no-buffer drops/interleaves) 7/128/0/0/0
     Conversation 25, linktype: ip, length: 1494
     source: 7.0.0.201, destination: 6.6.6.6, id: 0x0000, ttl: 63, prot: 255

  (depth/weight/total drops/no-buffer drops/interleaves) 2/256/0/0/0
     Conversation 26, linktype: ip, length: 1494
     source: 7.0.0.1, destination: 6.6.6.6, id: 0x0000, ttl: 63, prot: 255
```

Lorsque tous les paquets de 7.0.0.200 ont été mis en file d'attente hors du routeur, les éléments suivants s'affichent :

```
alcazaba#show queue ATM 4/0.102
     Interface ATM4/0.102 VC 0/102
     Queueing strategy: weighted fair
     Total output drops per VC: 0
     Output queue: 9/512/64/0 (size/max total/threshold/drops)
        Conversations  2/3/16 (active/max active/max total)
        Reserved Conversations 2/2 (allocated/max allocated)

  (depth/weight/total drops/no-buffer drops/interleaves) 7/128/0/0/0
     Conversation 25, linktype: ip, length: 1494
     source: 7.0.0.201, destination: 6.6.6.6, id: 0x0000, ttl: 63, prot: 255

  (depth/weight/total drops/no-buffer drops/interleaves) 2/256/0/0/0
     Conversation 26, linktype: ip, length: 1494
     source: 7.0.0.1, destination: 6.6.6.6, id: 0x0000, ttl: 63, prot: 255
```

Comme vous pouvez le voir ici, les flux de 7.0.0.200 et 7.0.0.201 ont le même poids (128). Ce poids correspond à la moitié du poids attribué au débit du 7.0.0.1 (256). Cela correspond au fait que notre petite bande passante de classe est deux fois plus grande que notre grande.

Vérification de la distribution de bande passante

Comment vérifier la distribution de bande passante entre les différents flux ? La méthode de mise en file d'attente FIFO est utilisée dans chaque classe. Notre petite classe est remplie de dix paquets du premier flux et de dix paquets du second flux. Le premier flux est retiré de la petite classe à 32 kbits/s. Dès qu'ils ont été envoyés, les dix paquets de l'autre flux sont également envoyés. En attendant, les paquets de notre grande classe sont retirés à 16 kbits/s.

Nous pouvons voir que, puisque le générateur de trafic envoie une rafale à 100 Mbits/s, le circuit virtuel permanent sera surchargé. Cependant, comme il n'y a pas de trafic sur le circuit virtuel permanent au démarrage du test et que les paquets de 7.0.0.1 sont les premiers à atteindre le routeur, certains paquets de 7.0.0.1 seront envoyés avant que CBWFQ ne démarre en raison d'une congestion (en d'autres termes, avant que l'anneau de transmission ne soit plein).

Puisque la taille des particules est de 512 octets et que la taille des anneaux de transmission est de trois particules, nous pouvons voir que deux paquets de 7.0.0.1 sont envoyés avant que la congestion ne se produise. L'une est immédiatement envoyée sur le câble et l'autre est stockée dans les trois particules formant la file d'attente FIFO de la carte de port.

Nous pouvons voir les débogages ci-dessous sur le périphérique de l'évier (qui est simplement un routeur) :

```
Nov 13 12:19:34.216: IP: s=7.0.0.1 (FastEthernet0/1), d=6.6.6.6, len 1482, rcvd 4
   Nov 13 12:19:34.428: IP: s=7.0.0.1 (FastEthernet0/1), d=6.6.6.6, Len 1482, rcvd 4

   !--- congestion occurs here. Nov 13 12:19:34.640: IP: s=7.0.0.200 (FastEthernet0/1),
d=6.6.6.6, Len 1482, rcvd 4 Nov 13 12:19:34.856: IP: s=7.0.0.200 (FastEthernet0/1), d=6.6.6.6,
Len 1482, rcvd 4 Nov 13 12:19:35.068: IP: s=7.0.0.1 (FastEthernet0/1), d=6.6.6.6, Len 1482, rcvd
4 Nov 13 12:19:35.280: IP: s=7.0.0.200 (FastEthernet0/1), d=6.6.6.6, Len 1482, rcvd 4 Nov 13
12:19:35.496: IP: s=7.0.0.200 (FastEthernet0/1), d=6.6.6.6, Len 1482, rcvd 4 Nov 13
12:19:35.708: IP: s=7.0.0.1 (FastEthernet0/1), d=6.6.6.6, Len 1482, rcvd 4 Nov 13 12:19:35.920:
```

```
IP: s=7.0.0.200 (FastEthernet0/1), d=6.6.6.6, Len 1482, rcvd 4 Nov 13 12:19:36.136: IP:
s=7.0.0.200 (FastEthernet0/1), d=6.6.6.6, Len 1482, rcvd 4 Nov 13 12:19:36.348: IP: s=7.0.0.1
(FastEthernet0/1), d=6.6.6.6, Len 1482, rcvd 4 Nov 13 12:19:36.560: IP: s=7.0.0.200
(FastEthernet0/1), d=6.6.6.6, Len 1482, rcvd 4 Nov 13 12:19:36.776: IP: s=7.0.0.200
(FastEthernet0/1), d=6.6.6.6, Len 1482, rcvd 4 Nov 13 12:19:36.988: IP: s=7.0.0.1
(FastEthernet0/1), d=6.6.6.6, Len 1482, rcvd 4 Nov 13 12:19:37.200: IP: s=7.0.0.200
(FastEthernet0/1), d=6.6.6.6, Len 1482, rcvd 4 Nov 13 12:19:37.416: IP: s=7.0.0.200
(FastEthernet0/1), d=6.6.6.6, Len 1482, rcvd 4 Nov 13 12:19:37.628: IP: s=7.0.0.1
(FastEthernet0/1), d=6.6.6.6, Len 1482, rcvd 4 Nov 13 12:19:37.840: IP: s=7.0.0.201
(FastEthernet0/1), d=6.6.6.6, Len 1482, rcvd 4 Nov 13 12:19:38.056: IP: s=7.0.0.201
(FastEthernet0/1), d=6.6.6.6, Len 1482, rcvd 4 Nov 13 12:19:38.268: IP: s=7.0.0.1
(FastEthernet0/1), d=6.6.6.6, Len 1482, rcvd 4 Nov 13 12:19:38.480: IP: s=7.0.0.201
(FastEthernet0/1), d=6.6.6.6, Len 1482, rcvd 4 Nov 13 12:19:38.696: IP: s=7.0.0.201
(FastEthernet0/1), d=6.6.6.6, Len 1482, rcvd 4 Nov 13 12:19:38.908: IP: s=7.0.0.1
(FastEthernet0/1), d=6.6.6.6, Len 1482, rcvd 4 Nov 13 12:19:39.136: IP: s=7.0.0.201
(FastEthernet0/1), d=6.6.6.6, Len 1482, rcvd 4 Nov 13 12:19:39.348: IP: s=7.0.0.201
(FastEthernet0/1), d=6.6.6.6, Len 1482, rcvd 4 Nov 13 12:19:39.560: IP: s=7.0.0.1
(FastEthernet0/1), d=6.6.6.6, Len 1482, rcvd 4 Nov 13 12:19:39.776: IP: s=7.0.0.201
(FastEthernet0/1), d=6.6.6.6, Len 1482, rcvd 4 Nov 13 12:19:39.988: IP: s=7.0.0.201
(FastEthernet0/1), d=6.6.6.6, Len 1482, rcvd 4 Nov 13 12:19:40.200: IP: s=7.0.0.201
(FastEthernet0/1), d=6.6.6.6, Len 1482, rcvd 4 Nov 13 12:19:40.416: IP: s=7.0.0.201
(FastEthernet0/1), d=6.6.6.6, Len 1482, rcvd 4
```

Puisque les tailles des paquets pour les deux flux sont identiques, selon la formule du temps de planification, nous devrions voir deux paquets de notre petite classe envoyés pour chaque paquet de notre grande classe. C'est exactement ce que nous voyons dans les débogages ci-dessus.

Essai B

Pour notre deuxième test, remplissons les classes de la manière suivante :

- Petite classe : nous avons configuré le paramètre de bande passante à 32 kbits/s. Dix paquets de 500 octets de 7.0.0.200 sont générés, suivis de dix paquets de 1 500 octets de 7.0.0.201.
- Grande classe : nous avons configuré le paramètre de bande passante à 16 kbits/s. La classe stocke un flux de paquets de 1 500 octets provenant de 7.0.0.1.

Le générateur de trafic envoie une rafale de trafic à 100 Mbits/s au routeur 2 dans l'ordre suivant :

1. Dix paquets de 1 500 octets de 7.0.0.1.
2. Dix paquets de 500 octets de 7.0.0.200.
3. Dix paquets de 1 500 octets de 7.0.0.201.

FIFO est configuré dans chaque classe.

Vérification du poids du flux

L'étape suivante consiste à vérifier le poids appliqué aux flux classifiés :

```
alcazaba#show queue ATM 4/0.102
   Interface ATM4/0.102 VC 0/102
   Queueing strategy: weighted fair
   Total output drops per VC: 0
   Output queue: 23/512/64/0 (size/max total/threshold/drops)
      Conversations  2/3/16 (active/max active/max total)
      Reserved Conversations 2/2 (allocated/max allocated)
```

```
  (depth/weight/total drops/no-buffer drops/interleaves) 15/128/0/0/0
     Conversation 25, linktype: ip, length: 494
     source: 7.0.0.200, destination: 6.6.6.6, id: 0x0000, ttl: 63, prot: 255

  (depth/weight/total drops/no-buffer drops/interleaves) 8/256/0/0/0
     Conversation 26, linktype: ip, length: 1494
     source: 7.0.0.1, destination: 6.6.6.6, id: 0x0000, ttl: 63, prot: 255

   alcazaba#show queue ATM 4/0.102
   Interface ATM4/0.102 VC 0/102
   Queueing strategy: weighted fair
   Total output drops per VC: 0
   Output queue: 13/512/64/0 (size/max total/threshold/drops)
        Conversations  2/3/16 (active/max active/max total)
        Reserved Conversations 2/2 (allocated/max allocated)

  (depth/weight/total drops/no-buffer drops/interleaves) 8/128/0/0/0
     Conversation 25, linktype: ip, length: 1494
     source: 7.0.0.201, destination: 6.6.6.6, id: 0x0000, ttl: 63, prot: 255

  (depth/weight/total drops/no-buffer drops/interleaves) 5/256/0/0/0
     Conversation 26, linktype: ip, length: 1494
     source: 7.0.0.1, destination: 6.6.6.6, id: 0x0000, ttl: 63,
```

Comme vous pouvez le voir dans le résultat ci-dessus, les flux de 7.0.0.200 et 7.0.0.201 ont reçu le même poids (128). Ce poids correspond à la moitié du poids attribué au débit du 7.0.0.1. Cela correspond au fait que la petite classe a une bande passante deux fois plus grande que la grande classe.

Vérification de la distribution de bande passante

Nous pouvons produire les débogages suivants à partir du périphérique de l'évier :

```
Nov 14 06:52:01.761: IP: s=7.0.0.1 (FastEthernet0/1), d=6.6.6.6, Len 1482, rcvd 4
   Nov 14 06:52:01.973: IP: s=7.0.0.1 (FastEthernet0/1), d=6.6.6.6, Len 1482, rcvd 4

   !--- Congestion occurs here. Nov 14 06:52:02.049: IP: s=7.0.0.200 (FastEthernet0/1),
d=6.6.6.6, Len 482, rcvd 4 Nov 14 06:52:02.121: IP: s=7.0.0.200 (FastEthernet0/1), d=6.6.6.6,
Len 482, rcvd 4 Nov 14 06:52:02.193: IP: s=7.0.0.200 (FastEthernet0/1), d=6.6.6.6, Len 482, rcvd
4 Nov 14 06:52:02.269: IP: s=7.0.0.200 (FastEthernet0/1), d=6.6.6.6, Len 482, rcvd 4 Nov 14
06:52:02.341: IP: s=7.0.0.200 (FastEthernet0/1), d=6.6.6.6, Len 482, rcvd 4 Nov 14 06:52:02.413:
IP: s=7.0.0.200 (FastEthernet0/1), d=6.6.6.6, Len 482, rcvd 4 Nov 14 06:52:02.629: IP: s=7.0.0.1
(FastEthernet0/1), d=6.6.6.6, Len 1482, rcvd 4 Nov 14 06:52:02.701: IP: s=7.0.0.200
(FastEthernet0/1), d=6.6.6.6, Len 482, rcvd 4 Nov 14 06:52:02.773: IP: s=7.0.0.200
(FastEthernet0/1), d=6.6.6.6, Len 482, rcvd 4 Nov 14 06:52:02.849: IP: s=7.0.0.200
(FastEthernet0/1), d=6.6.6.6, Len 482, rcvd 4 Nov 14 06:52:02.921: IP: s=7.0.0.200
(FastEthernet0/1), d=6.6.6.6, Len 482, rcvd 4 Nov 14 06:52:03.149: IP: s=7.0.0.1
(FastEthernet0/1), d=6.6.6.6, Len 1482, rcvd 4 Nov 14 06:52:03.361: IP: s=7.0.0.201
(FastEthernet0/1), d=6.6.6.6, Len 1482, rcvd 4 Nov 14 06:52:03.572: IP: s=7.0.0.201
(FastEthernet0/1), d=6.6.6.6, Len 1482, rcvd 4 Nov 14 06:52:03.788: IP: s=7.0.0.1
(FastEthernet0/1), d=6.6.6.6, Len 1482, rcvd 4 Nov 14 06:52:04.000: IP: s=7.0.0.201
(FastEthernet0/1), d=6.6.6.6, Len 1482, rcvd 4 Nov 14 06:52:04.212: IP: s=7.0.0.201
(FastEthernet0/1), d=6.6.6.6, Len 1482, rcvd 4 Nov 14 06:52:04.428: IP: s=7.0.0.1
(FastEthernet0/1), d=6.6.6.6, Len 1482, rcvd 4 Nov 14 06:52:04.640: IP: s=7.0.0.201
(FastEthernet0/1), d=6.6.6.6, Len 1482, rcvd 4 Nov 14 06:52:04.852: IP: s=7.0.0.201
(FastEthernet0/1), d=6.6.6.6, Len 1482, rcvd 4 Nov 14 06:52:05.068: IP: s=7.0.0.1
(FastEthernet0/1), d=6.6.6.6, Len 1482, rcvd 4 Nov 14 06:52:05.280: IP: s=7.0.0.201
(FastEthernet0/1), d=6.6.6.6, Len 1482, rcvd 4 Nov 14 06:52:05.492: IP: s=7.0.0.201
(FastEthernet0/1), d=6.6.6.6, Len 1482, rcvd 4 Nov 14 06:52:05.708: IP: s=7.0.0.1
(FastEthernet0/1), d=6.6.6.6, Len 1482, rcvd 4 Nov 14 06:52:05.920: IP: s=7.0.0.201
(FastEthernet0/1), d=6.6.6.6, Len 1482, rcvd 4 Nov 14 06:52:06.132: IP: s=7.0.0.201
```

```
(FastEthernet0/1), d=6.6.6.6, Len 1482, rcvd 4 Nov 14 06:52:06.348: IP: s=7.0.0.1
(FastEthernet0/1), d=6.6.6.6, Len 1482, rcvd 4 Nov 14 06:52:06.560: IP: s=7.0.0.1
(FastEthernet0/1), d=6.6.6.6, Len 1482, rcvd 4
```

Dans ce scénario, les flux de notre petite classe n'ont pas la même taille de paquet. Par conséquent, la distribution des paquets n'est pas aussi triviale que pour le test A ci-dessus.

Horaires de planification

Examinons de plus près les horaires de planification de chaque paquet. Le temps de planification des paquets est calculé à l'aide de la formule suivante :

```
scheduling tail_time= sub_queue_tail_time + pktsize *
     weight
```

Pour différentes tailles de paquets, le temps de planification utilise la formule suivante :

```
500 bytes (small class): scheduling tail_time = x + 494 * 128
     = x + 63232
     1500 bytes (small class): scheduling tail_time = x + 1494 *
     128 = x + 191232
     1500 bytes (big class): scheduling tail_time = x + 1494 *
     256 = x + 382464
```

Àpartir de ces formules, nous pouvons voir que six paquets de 500 octets de notre petite classe sont transmis pour chaque paquet de 1500 octets de notre grande classe (voir la sortie de débogage ci-dessus).

Nous pouvons également voir que deux paquets de 1500 octets de notre petite classe sont envoyés pour un paquet de 1500 octets de notre grande classe (voir la sortie de débogage ci-dessus).

Àpartir de nos tests ci-dessus, nous pouvons conclure ce qui suit :

- La taille de la sonnerie de transmission (limite d'anneau de transmission) détermine la vitesse à laquelle le mécanisme de mise en file d'attente commence à fonctionner. Nous pouvons voir l'impact avec l'augmentation de la requête ping RTT lorsque la limite de l'anneau de transmission augmente. Par conséquent, si vous mettez en oeuvre CBWFQ ou Low Latency Queueing [LLQ]), envisagez de réduire la limite de sonnerie de transmission.
- CBWFQ permet un partage équitable de la bande passante d'interface entre différentes classes.

Informations connexes

- Mise en file d'attente pondérée par classe par circuit virtuel (CBWFQ par circuit virtuel) sur les routeurs Cisco 7200, 3600 et 2600
- Mise en file d'attente pondérée par classe par circuit virtuel sur les plates-formes RSP
- Présentation de la mise en file d'attente pondérée (WFQ) sur des interfaces ATM
- Prise en charge de la technologie de classe de service IP à ATM

- Support technologique ATM
- Support et documentation techniques - Cisco Systems