Présentation de la mise en file d'attente pondérée (WFQ) sur des interfaces ATM

Contenu

Introduction

Ce document présente la mise en file d'attente du trafic qui utilise la technologie WFQ (Weighted Fair Queuing).

WFQ a été introduit afin d'activer les liaisons à vitesse lente, telles que les liaisons série, pour fournir un traitement équitable pour tous les types de trafic. Pour ce faire, WFQ 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 besoin de 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.

Mais 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.

La mise en file d'attente pondérée basée sur les classes (CBWFQ) offre une solution à ces limitations.

Contrairement à la WFQ standard, CBWFQ vous permet de définir des classes de trafic. Vous pouvez également leur appliquer des paramètres, tels que la bande passante et les limites de file d'attente. 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 ensuite appliqué aux classes, qui peuvent inclure plusieurs flux, plutôt que les flux eux-mêmes.

Pour plus d'informations sur la configuration de CBWFQ, reportez-vous aux documents suivants :

- Mise en file d'attente pondérée par classe et par circuit virtuel (CBWFQ) sur les routeurs Cisco 7200, 3600 et 2600
- Mise en file d'attente pondérée basée sur les classes par circuit virtuel sur les plates-formes basées sur un processeur de commutation routage (RSP)

Les interfaces ATM ne prennent pas en charge la WFQ native basée sur le flux configurée directement sur une interface avec la commande **fair-queue**. Mais avec le logiciel qui prend en charge CBWFQ, vous pouvez configurer WFQ basé sur le flux dans la classe par défaut, comme illustré dans cet exemple :

```
policy-map test
 class class-default
  fair-queue
!
interface ATMx/y.z point-to-point
 ip address a.b.c.d M.M.M.M
 pvc A/B
  service-policy output test
```

Conditions préalables

Conditions requises

Aucune spécification déterminée n'est requise pour ce document.

Components Used

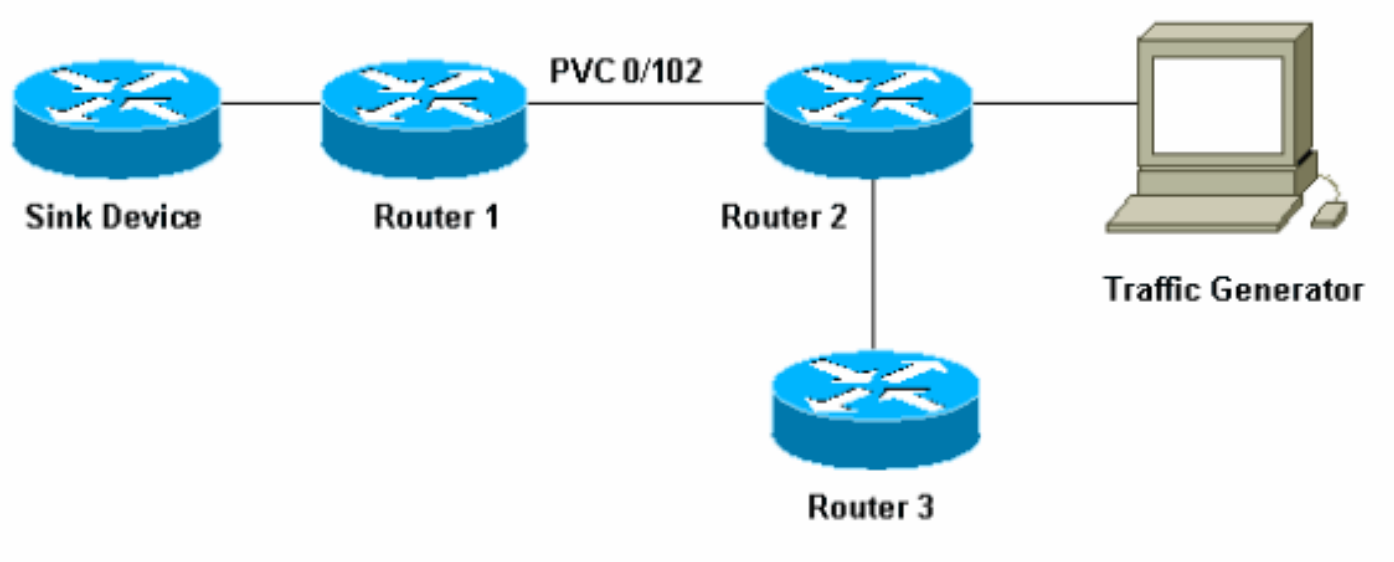
Ce document n'est pas limité à des versions de matériel et de logiciel spécifiques.

Conventions

Pour plus d'informations sur les conventions utilisées dans ce document, reportez-vous à Conventions relatives aux conseils techniques Cisco.

Diagramme du réseau

Utilisez cette configuration afin d'illustrer le fonctionnement de WFQ :



Dans cette configuration, les paquets peuvent être stockés dans l'une de ces deux files d'attente :

- La file d'attente FIFO (1er entré dans la file d'attente) du matériel sur l'adaptateur de port et le module réseau
- File d'attente dans le logiciel Cisco IOS®, sur la mémoire d'entrée/sortie [E/S] du routeur, où les fonctionnalités de qualité de service (QoS) telles que CBWFQ peuvent être appliquées

La file d'attente FIFO sur l'adaptateur de port stocke les paquets avant qu'ils ne soient segmentés en cellules pour transmission. Lorsque cette file d'attente est pleine, l'adaptateur de port 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 s'arrête pour 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 la QoS.

Comment définir la limite de la 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 *<taille>* 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 qu'il ne soit segmenté.
- Il accélère l'utilisation de la QoS dans le logiciel IOS.

Impact de la limite de sonnerie de transmission

Examinez l'impact de la limite de l'anneau de transmission qui utilise la configuration indiquée dans le schéma de réseau précédent. Supposez que :

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

Examinez deux configurations qui utilisent des limites de cycle de transmission différentes afin de voir l'impact que cela a.

Exemple A

Dans cet exemple, vous définissez la sonnerie de transmission sur trois (tx-ring-limit=3). Voici ce que vous voyez lorsque vous envoyez 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

router2#show queue atm 4/0.102
   Interface ATM4/0.102 VC 0/102
   Queuing strategy: weighted fair
   Total output drops per VC: 1505772
   Output queue: 65/512/64/1505772 (size/max total/threshold/drops)
    Conversations  2/3/16 (active/max active/max total)
    Reserved Conversations 0/0 (allocated/max allocated)

(depth/weight/discards/tail drops/interleaves) 1/32384/0/0/0
   Conversation 2, linktype: ip, length: 58
   source: 8.0.0.1, destination: 6.6.6.6, id: 0x2DA1, ttl: 254, prot: 1
   !--- ping (depth/weight/discards/tail drops/interleaves) 64/32384/1505776/0/0
   Conversation 15, linktype: ip, length: 1494
   source: 7.0.0.1, destination: 6.6.6.6, id: 0x0000, ttl: 63, prot: 255
   !--- This is traffic from the traffic generator.
```

Exemple B

Dans cet exemple, vous définissez la sonnerie de transmission sur 40 (tx-ring-limit=40). Voici ce que vous voyez lorsque vous utilisez 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 ms
```

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. Vous pouvez en déduire qu'une grande limite de sonnerie de transmission peut entraîner des retards importants dans la transmission.

Comment calculer le poids

Dans la sortie **show queue atm** dans l'exemple A, vous voyez qu'un poids est attribué à chaque conversation. Examinez ceci plus en détail :

```
router2#show queue ATM 4/0.102
   Interface ATM4/0.102 VC 0/102
   Queuing strategy: weighted fair
   Total output drops per VC: 1505772
   Output queue: 65/512/64/1505772 (size/max total/threshold/drops)
    Conversations  2/3/16 (active/max active/max total)
    Reserved Conversations 0/0 (allocated/max allocated)

(depth/weight/discards/tail drops/interleaves) 1/32384/0/0/0
   Conversation 2, linktype: ip, length: 58
   source: 8.0.0.1, destination: 6.6.6.6, id: 0x2DA1, ttl: 254, prot: 1

(depth/weight/discards/tail drops/interleaves) 64/32384/1505776/0/0
   Conversation 15, linktype: ip, length: 1494
   source: 7.0.0.1, destination: 6.6.6.6, id: 0x0000, ttl: 63, prot: 255
```

Lorsque vous utilisez WFQ, vous pouvez calculer le poids de chaque conversation à l'aide de cette formule :

- **weight=32384/(priority+1)** - pour le logiciel Cisco IOS Version 12.0(5)T et ultérieure.
- **weight=4096/(priority+1)** - pour les versions du logiciel Cisco IOS antérieures à 12.0(5)T.

Comment calculer l'heure de planification

Vous pouvez maintenant utiliser ces pondérations afin de calculer le temps de planification de chaque paquet, lorsque le paquet est transféré de la file d'attente IOS vers la file d'attente FIFO de l'adaptateur de port ou du module réseau.

Vous pouvez calculer le temps de planification de sortie à l'aide de cette formule, où **queue_tail_time** est l'heure de planification actuelle :

durée de planification de sortie = queue_tail_time + pktsize*weight

Fonctionnement de WFQ

Cette section explique comment fonctionne WFQ. Le principe de WFQ est que les paquets dont le poids est faible, ou les petits paquets, doivent obtenir la priorité lorsqu'ils sont envoyés.

Créez un flux comprenant dix paquets de grande taille et quatre paquets de petite taille (de 82 octets) qui utilise un générateur de trafic afin de vérifier ceci.

Dans cet exemple, le routeur 2 est un routeur Cisco 7200 avec un PA-A3 (adaptateur de port ATM). Ceci est important car la taille de la file d'attente FIFO de sortie sur l'adaptateur de port est exprimée en particules et non en paquets. Voir Qu'est-ce qu'une particule ? pour plus d'informations.

Qu'est-ce qu'une particule ?

Au lieu d'allouer une partie de mémoire contiguë pour un tampon, la mise en mémoire tampon des particules alloue des morceaux de mémoire discontinus (dispersés), appelés particules, puis les relie ensemble afin de former un 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, la taille des particules est de 512 octets.

Utilisez la commande **show buffers** afin de vérifier si les routeurs Cisco 7200 utilisent des particules :

```
router#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
ATM2/0 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

Voici quelques tests pour illustrer la fonctionnalité WFQ. Dans ce premier test, déterminez si la bande passante peut être partagée entre différentes conversations.

Au cours de ce test, vous avez fait en sorte que le générateur de trafic envoie le trafic assez rapidement pour surcharger le circuit virtuel permanent 0/102 entre le routeur 1 et le routeur 2. Exécutez une requête ping du routeur 3 au routeur 1 sur le même circuit virtuel permanent :

```
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:
......... (WFQ is enabled here)!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!.[break]
Success rate is 98 percent (604/613), round-trip min/avg/max = 164/190/232 ms
```

Comme vous pouvez le voir dans le résultat affiché, jusqu'à ce que WFQ soit activé sur l'interface, le trafic empêche les autres trafics de traverser et affame la ligne. Dès que WFQ est activé, la requête ping aboutit.

Vous pouvez voir à partir de cela que, avec l'utilisation de WFQ, la bande passante peut être partagée entre différentes conversations sans que l'une bloque les autres.

Essai B

Voici *comment* la bande passante est partagée.

Le flux que le générateur de trafic envoie est une rafale composée de dix paquets volumineux, suivie de quatre paquets plus petits de 82 octets. Vous envoyez ceci à 100 Mbits/s au routeur 2. Lorsque vous envoyez la rafale, la file d’attente de sortie sur l’interface ATM du routeur 2 est vide. Le routeur 2 envoie ces paquets via un circuit virtuel permanent de 10 Ko (il s’agit d’un circuit virtuel permanent très lent) afin de s’assurer que la congestion se produit sur la file d’attente de sortie.

Effectuez ce test en plusieurs étapes afin de simplifier ce processus :

Étape 1

Le trafic important comprend dix paquets de 482 octets. Puisque les particules sur PA-A3 sont de 512 octets, chaque paquet, grand ou petit, doit prendre une particule lorsqu'il est stocké dans la file d'attente de sortie de l'adaptateur de port. Le routeur a une limite de sonnerie de transmission de trois (tx-ring-limit=3). Voici un exemple de ce que vous pouvez voir sur le périphérique de l'évier :

```
   .Nov  7 15:39:13.776: IP: s=7.0.0.1 (FastEthernet0/1), d=6.6.6.6, len 482, rcvd 4
   .Nov  7 15:39:13.776: IP: s=7.0.0.1 (FastEthernet0/1), d=6.6.6.6, Len 482, unknown protocol
   .Nov  7 15:39:14.252: IP: s=7.0.0.1 (FastEthernet0/1), d=6.6.6.6, Len 482, rcvd 4
   .Nov  7 15:39:14.252: IP: s=7.0.0.1 (FastEthernet0/1), d=6.6.6.6, Len 482, unknown protocol
   .Nov  7 15:39:14.732: IP: s=7.0.0.1 (FastEthernet0/1), d=6.6.6.6, Len 482, rcvd 4
   .Nov  7 15:39:14.732: IP: s=7.0.0.1 (FastEthernet0/1), d=6.6.6.6, Len 482, unknown protocol
   .Nov  7 15:39:15.208: IP: s=7.0.0.1 (FastEthernet0/1), d=6.6.6.6, Len 482, rcvd 4
   .Nov  7 15:39:15.208: IP: s=7.0.0.1 (FastEthernet0/1), d=6.6.6.6, Len 482, unknown protocol

   !--- Congestion occurs at this point. .Nov 7 15:39:15.512: IP: s=7.0.0.200 (FastEthernet0/1),
d=6.6.6.6, Len 82, rcvd 4 .Nov 7 15:39:15.516: IP: s=7.0.0.200 (FastEthernet0/1), d=6.6.6.6, Len
82, unknown protocol .Nov 7 15:39:15.644: IP: s=7.0.0.200 (FastEthernet0/1), d=6.6.6.6, Len 82,
rcvd 4 .Nov 7 15:39:15.644: IP: s=7.0.0.200 (FastEthernet0/1), d=6.6.6.6, Len 82, unknown
protocol .Nov 7 15:39:15.776: IP: s=7.0.0.200 (FastEthernet0/1), d=6.6.6.6, Len 82, rcvd 4 .Nov
7 15:39:15.776: IP: s=7.0.0.200 (FastEthernet0/1), d=6.6.6.6, Len 82, unknown protocol .Nov 7
```

```
15:39:15.904: IP: s=7.0.0.200 (FastEthernet0/1), d=6.6.6.6, Len 82, rcvd 4 .Nov 7 15:39:15.904:
IP: s=7.0.0.200 (FastEthernet0/1), d=6.6.6.6, Len 82, unknown protocol .Nov 7 15:39:16.384: IP:
s=7.0.0.1 (FastEthernet0/1), d=6.6.6.6, Len 482, rcvd 4 .Nov 7 15:39:16.384: IP: s=7.0.0.1
(FastEthernet0/1), d=6.6.6.6, Len 482, unknown protocol .Nov 7 15:39:16.860: IP: s=7.0.0.1
(FastEthernet0/1), d=6.6.6.6, Len 482, rcvd 4 .Nov 7 15:39:16.860: IP: s=7.0.0.1
(FastEthernet0/1), d=6.6.6.6, Len 482, unknown protocol .Nov 7 15:39:17.340: IP: s=7.0.0.1
(FastEthernet0/1), d=6.6.6.6, Len 482, rcvd 4 .Nov 7 15:39:17.340: IP: s=7.0.0.1
(FastEthernet0/1), d=6.6.6.6, Len 482, unknown protocol .Nov 7 15:39:17.816: IP: s=7.0.0.1
(FastEthernet0/1), d=6.6.6.6, Len 482, rcvd 4 .Nov 7 15:39:17.820: IP: s=7.0.0.1
(FastEthernet0/1), d=6.6.6.6, Len 482, unknown protocol .Nov 7 15:39:18.296: IP: s=7.0.0.1
(FastEthernet0/1), d=6.6.6.6, Len 482, rcvd 4 .Nov 7 15:39:18.296: IP: s=7.0.0.1
(FastEthernet0/1), d=6.6.6.6, Len 482, unknown protocol .Nov 7 15:39:18.776: IP: s=7.0.0.1
(FastEthernet0/1), d=6.6.6.6, Len 482, rcvd 4 .Nov 7 15:39:18.776: IP: s=7.0.0.1
(FastEthernet0/1), d=6.6.6.6, Len 482, unknown protocol
```

Vous pouvez voir les quatre paquets de 482 octets envoyés avant les paquets de 82 octets, qui devraient normalement obtenir la priorité. C'est pourquoi cela arrive.

Comme la rafale est principalement composée de dix paquets de 482 octets, ceux-ci atteignent le routeur en premier, suivi des paquets de 82 octets. Puisque les paquets de 482 octets arrivent à un moment où il n'y a pas de congestion, car il n'y a pas de trafic, un paquet est immédiatement mis en file d'attente vers la segmentation et le réassemblage de l'adaptateur de port (SAR) pour être tronqué dans les cellules et envoyé sur le câble. En d'autres termes, la sonnerie de transmission est toujours vide.

Vous pouvez calculer que le temps nécessaire pour envoyer un paquet de 482 octets est supérieur au temps nécessaire pour le générateur de trafic afin d'envoyer la rafale totale. Vous pouvez donc supposer que, lorsque le premier paquet de 482 octets est mis en file d'attente sur l'adaptateur de port, plus de paquets de 482 octets de la rafale sont déjà présents dans le routeur. Par conséquent, plus de paquets de 482 octets peuvent être mis en file d'attente sur l'anneau de transmission. Trois paquets supplémentaires de 482 octets sont mis en file d'attente avec l'utilisation des trois particules libres présentes.

Remarque : Les paquets sont mis en file d'attente dans l'anneau de transmission dès qu'il y a une particule libre, même s'ils ont besoin de plus d'une particule pour être stockés.

Àce stade, il y a des embouteillages, puisque les trois particules sont pleines. Par conséquent, la mise en file d’attente démarre dans IOS. Lorsque les quatre paquets de 82 octets atteignent enfin le routeur, il y a congestion. Ces quatre paquets sont mis en file d'attente et WFQ est utilisé sur les deux flux. Regardez la file d'attente ATM qui utilise la commande **show queue ATM** afin de voir ceci :

```
router2#show queue ATM 4/0.102 vc 0/102
   Interface ATM4/0.102 VC 0/102
   Queuing strategy: weighted fair
   Total output drops per VC: 0
   Output queue: 10/512/64/0 (size/max total/threshold/drops)
     Conversations  2/4/16 (active/max active/max total)
     Reserved Conversations 0/0 (allocated/max allocated)

(depth/weight/total drops/no-buffer drops/interleaves) 4/32384/0/0/0
   Conversation 6, linktype: ip, length: 82
   source: 7.0.0.200, destination: 6.6.6.6, id: 0x0000, ttl: 63, prot: 255

(depth/weight/total drops/no-buffer drops/interleaves) 6/32384/0/0/0
   Conversation 15, linktype: ip, length: 482
   source: 7.0.0.1, destination: 6.6.6.6, id: 0x0000, ttl: 63, prot: 255
```

Vous pouvez voir dans les débogages que les quatre premiers paquets de 482 octets sont suivis

des paquets de 82 octets. Ces petits paquets sortent du routeur avant les grands paquets. Cela indique que, dès que la congestion se produit, les petits paquets obtiennent la priorité sur les grands paquets.

Utilisez les formules de poids et de temps de planification indiquées dans la section Calcul du poids afin de vérifier ceci.

Étape 2

Si vous augmentez la limite de l'anneau de transmission à cinq et que les grands paquets sont de 482 octets, alors, conformément à la sortie précédente, vous devriez voir six paquets de 482 octets avant qu'un encombrement ne se produise, suivis de quatre paquets de 82 octets, puis quatre autres paquets de 482 octets :

```
.Nov  7 15:49:57.365: IP: s=7.0.0.1 (FastEthernet0/1), d=6.6.6.6, Len 482, rcvd 4
.Nov  7 15:49:57.365: IP: s=7.0.0.1 (FastEthernet0/1), d=6.6.6.6, Len 482, unknown protocol
.Nov  7 15:49:57.841: IP: s=7.0.0.1 (FastEthernet0/1), d=6.6.6.6, Len 482, rcvd 4
.Nov  7 15:49:57.845: IP: s=7.0.0.1 (FastEthernet0/1), d=6.6.6.6, Len 482, unknown protocol
.Nov  7 15:49:58.321: IP: s=7.0.0.1 (FastEthernet0/1), d=6.6.6.6, Len 482, rcvd 4
.Nov  7 15:49:58.321: IP: s=7.0.0.1 (FastEthernet0/1), d=6.6.6.6, Len 482, unknown protocol
.Nov  7 15:49:58.797: IP: s=7.0.0.1 (FastEthernet0/1), d=6.6.6.6, Len 482, rcvd 4
.Nov  7 15:49:58.801: IP: s=7.0.0.1 (FastEthernet0/1), d=6.6.6.6, Len 482, unknown protocol
.Nov  7 15:49:59.277: IP: s=7.0.0.1 (FastEthernet0/1), d=6.6.6.6, Len 482, rcvd 4
.Nov  7 15:49:59.277: IP: s=7.0.0.1 (FastEthernet0/1), d=6.6.6.6, Len 482, unknown protocol
.Nov  7 15:49:59.757: IP: s=7.0.0.1 (FastEthernet0/1), d=6.6.6.6, Len 482, rcvd 4
.Nov  7 15:49:59.757: IP: s=7.0.0.1 (FastEthernet0/1), d=6.6.6.6, Len 482, unknown protocol
.Nov  7 15:49:59.973: IP: s=7.0.0.200 (FastEthernet0/1), d=6.6.6.6, Len 82, rcvd 4
.Nov  7 15:49:59.973: IP: s=7.0.0.200 (FastEthernet0/1), d=6.6.6.6, Len 82, unknown protocol
.Nov  7 15:50:00.105: IP: s=7.0.0.200 (FastEthernet0/1), d=6.6.6.6, Len 82, rcvd 4
.Nov  7 15:50:00.105: IP: s=7.0.0.200 (FastEthernet0/1), d=6.6.6.6, Len 82, unknown protocol
.Nov  7 15:50:00.232: IP: s=7.0.0.200 (FastEthernet0/1), d=6.6.6.6, Len 82, rcvd 4
.Nov  7 15:50:00.232: IP: s=7.0.0.200 (FastEthernet0/1), d=6.6.6.6, Len 82, unknown protocol
.Nov  7 15:50:00.364: IP: s=7.0.0.200 (FastEthernet0/1), d=6.6.6.6, Len 82, rcvd 4
.Nov  7 15:50:00.364: IP: s=7.0.0.200 (FastEthernet0/1), d=6.6.6.6, Len 82, unknown protocol
.Nov  7 15:50:00.840: IP: s=7.0.0.1 (FastEthernet0/1), d=6.6.6.6, Len 482, rcvd 4
.Nov  7 15:50:00.844: IP: s=7.0.0.1 (FastEthernet0/1), d=6.6.6.6, Len 482, unknown protocol
.Nov  7 15:50:01.320: IP: s=7.0.0.1 (FastEthernet0/1), d=6.6.6.6, Len 482, rcvd 4
.Nov  7 15:50:01.320: IP: s=7.0.0.1 (FastEthernet0/1), d=6.6.6.6, Len 482, unknown protocol
.Nov  7 15:50:01.796: IP: s=7.0.0.1 (FastEthernet0/1), d=6.6.6.6, Len 482, rcvd 4
.Nov  7 15:50:01.800: IP: s=7.0.0.1 (FastEthernet0/1), d=6.6.6.6, Len 482, unknown protocol
.Nov  7 15:50:02.276: IP: s=7.0.0.1 (FastEthernet0/1), d=6.6.6.6, Len 482, rcvd 4
.Nov  7 15:50:02.276: IP: s=7.0.0.1 (FastEthernet0/1), d=6.6.6.6, Len 482, unknown protocol
```

Comme vous pouvez le voir ici, c'est en effet ce qui se passe.

Étape 3

La taille de la particule est de 512 octets. Par conséquent, si l'anneau de transmission est exprimé en particules, et que vous utilisez des paquets légèrement plus grands que la taille des particules, chacun prend deux particules. Ceci est illustré par l'utilisation de paquets de 582 octets et d'un anneau de transmission de trois. Avec ces paramètres, vous devriez voir trois paquets de 582 octets. L'une est envoyée sans trafic sur l'interface ATM, ce qui laisse trois particules libres. Par conséquent, deux paquets supplémentaires peuvent être mis en file d'attente, suivis de quatre paquets de 82 octets, puis de sept paquets de 582 octets :

```
.Nov  7 15:51:34.604: IP: s=7.0.0.1 (FastEthernet0/1), d=6.6.6.6, Len 582, rcvd 4
```

```
.Nov  7 15:51:34.604: IP: s=7.0.0.1 (FastEthernet0/1), d=6.6.6.6, Len 582, unknown protocol
.Nov  7 15:51:35.168: IP: s=7.0.0.1 (FastEthernet0/1), d=6.6.6.6, Len 582, rcvd 4
.Nov  7 15:51:35.168: IP: s=7.0.0.1 (FastEthernet0/1), d=6.6.6.6, Len 582, unknown protocol
.Nov  7 15:51:35.732: IP: s=7.0.0.1 (FastEthernet0/1), d=6.6.6.6, Len 582, rcvd 4
.Nov  7 15:51:35.736: IP: s=7.0.0.1 (FastEthernet0/1), d=6.6.6.6, Len 582, unknown protocol
.Nov  7 15:51:35.864: IP: s=7.0.0.200 (FastEthernet0/1), d=6.6.6.6, Len 82, rcvd 4
.Nov  7 15:51:35.864: IP: s=7.0.0.200 (FastEthernet0/1), d=6.6.6.6, Len 82, unknown protocol
.Nov  7 15:51:35.996: IP: s=7.0.0.200 (FastEthernet0/1), d=6.6.6.6, Len 82, rcvd 4
.Nov  7 15:51:35.996: IP: s=7.0.0.200 (FastEthernet0/1), d=6.6.6.6, Len 82, unknown protocol
.Nov  7 15:51:36.124: IP: s=7.0.0.200 (FastEthernet0/1), d=6.6.6.6, Len 82, rcvd 4
.Nov  7 15:51:36.124: IP: s=7.0.0.200 (FastEthernet0/1), d=6.6.6.6, Len 82, unknown protocol
.Nov  7 15:51:36.256: IP: s=7.0.0.200 (FastEthernet0/1), d=6.6.6.6, Len 82, rcvd 4
.Nov  7 15:51:36.256: IP: s=7.0.0.200 (FastEthernet0/1), d=6.6.6.6, Len 82, unknown protocol
.Nov  7 15:51:36.820: IP: s=7.0.0.1 (FastEthernet0/1), d=6.6.6.6, Len 582, rcvd 4
.Nov  7 15:51:36.820: IP: s=7.0.0.1 (FastEthernet0/1), d=6.6.6.6, Len 582, unknown protocol
.Nov  7 15:51:37.384: IP: s=7.0.0.1 (FastEthernet0/1), d=6.6.6.6, Len 582, rcvd 4
.Nov  7 15:51:37.388: IP: s=7.0.0.1 (FastEthernet0/1), d=6.6.6.6, Len 582, unknown protocol
.Nov  7 15:51:37.952: IP: s=7.0.0.1 (FastEthernet0/1), d=6.6.6.6, Len 582, rcvd 4
.Nov  7 15:51:37.952: IP: s=7.0.0.1 (FastEthernet0/1), d=6.6.6.6, Len 582, unknown protocol
.Nov  7 15:51:38.604: IP: s=7.0.0.1 (FastEthernet0/1), d=6.6.6.6, Len 582, rcvd 4
.Nov  7 15:51:38.604: IP: s=7.0.0.1 (FastEthernet0/1), d=6.6.6.6, Len 582, unknown protocol
.Nov  7 15:51:39.168: IP: s=7.0.0.1 (FastEthernet0/1), d=6.6.6.6, Len 582, rcvd 4
.Nov  7 15:51:39.168: IP: s=7.0.0.1 (FastEthernet0/1), d=6.6.6.6, Len 582, unknown protocol
.Nov  7 15:51:39.732: IP: s=7.0.0.1 (FastEthernet0/1), d=6.6.6.6, Len 582, rcvd 4
.Nov  7 15:51:39.736: IP: s=7.0.0.1 (FastEthernet0/1), d=6.6.6.6, Len 582, unknown protocol
.Nov  7 15:51:40.300: IP: s=7.0.0.1 (FastEthernet0/1), d=6.6.6.6, Len 582, rcvd 4
.Nov  7 15:51:40.300: IP: s=7.0.0.1 (FastEthernet0/1), d=6.6.6.6, Len 582, unknown protocol
```

Étape 4

Prenez une taille de paquet de 1482 (trois particules) et définissez un anneau de transmission de cinq. Si l'anneau de transmission est défini en particules, vous voyez quelque chose de similaire :

- Un paquet transmis immédiatement
- Un paquet qui prend trois des cinq particules
- Un paquet mis en file d'attente car deux particules sont libres

```
.Nov  8 07:22:41.200: IP: s=7.0.0.1 (FastEthernet0/1), d=6.6.6.6, Len 1482, rcvd 4
.Nov  8 07:22:41.200: IP: s=7.0.0.1 (FastEthernet0/1), d=6.6.6.6, Len 1482, unknown protocol
.Nov  8 07:22:42.592: IP: s=7.0.0.1 (FastEthernet0/1), d=6.6.6.6, Len 1482, rcvd 4
.Nov  8 07:22:42.592: IP: s=7.0.0.1 (FastEthernet0/1), d=6.6.6.6, Len 1482, unknown protocol
.Nov  8 07:22:43.984: IP: s=7.0.0.1 (FastEthernet0/1), d=6.6.6.6, Len 1482, rcvd 4
.Nov  8 07:22:43.984: IP: s=7.0.0.1 (FastEthernet0/1), d=6.6.6.6, Len 1482, unknown protocol
.Nov  8 07:22:44.112: IP: s=7.0.0.200 (FastEthernet0/1), d=6.6.6.6, Len 82, rcvd 4
.Nov  8 07:22:44.112: IP: s=7.0.0.200 (FastEthernet0/1), d=6.6.6.6, Len 82, unknown protocol
.Nov  8 07:22:44.332: IP: s=7.0.0.200 (FastEthernet0/1), d=6.6.6.6, Len 82, rcvd 4
.Nov  8 07:22:44.332: IP: s=7.0.0.200 (FastEthernet0/1), d=6.6.6.6, Len 82, unknown protocol
.Nov  8 07:22:44.460: IP: s=7.0.0.200 (FastEthernet0/1), d=6.6.6.6, Len 82, rcvd 4
.Nov  8 07:22:44.460: IP: s=7.0.0.200 (FastEthernet0/1), d=6.6.6.6, Len 82, unknown protocol
.Nov  8 07:22:44.591: IP: s=7.0.0.200 (FastEthernet0/1), d=6.6.6.6, Len 82, rcvd 4
.Nov  8 07:22:44.591: IP: s=7.0.0.200 (FastEthernet0/1), d=6.6.6.6, Len 82, unknown protocol
.Nov  8 07:22:45.983: IP: s=7.0.0.1 (FastEthernet0/1), d=6.6.6.6, Len 1482, rcvd 4
.Nov  8 07:22:45.983: IP: s=7.0.0.1 (FastEthernet0/1), d=6.6.6.6, Len 1482, unknown protocol
.Nov  8 07:22:47.371: IP: s=7.0.0.1 (FastEthernet0/1), d=6.6.6.6, Len 1482, rcvd 4
.Nov  8 07:22:47.375: IP: s=7.0.0.1 (FastEthernet0/1), d=6.6.6.6, Len 1482, unknown protocol
.Nov  8 07:22:48.763: IP: s=7.0.0.1 (FastEthernet0/1), d=6.6.6.6, Len 1482, rcvd 4
.Nov  8 07:22:48.767: IP: s=7.0.0.1 (FastEthernet0/1), d=6.6.6.6, Len 1482, unknown protocol
.Nov  8 07:22:50.155: IP: s=7.0.0.1 (FastEthernet0/1), d=6.6.6.6, Len 1482, rcvd 4
.Nov  8 07:22:50.155: IP: s=7.0.0.1 (FastEthernet0/1), d=6.6.6.6, Len 1482, unknown protocol
.Nov  8 07:22:51.547: IP: s=7.0.0.1 (FastEthernet0/1), d=6.6.6.6, Len 1482, rcvd 4
.Nov  8 07:22:51.547: IP: s=7.0.0.1 (FastEthernet0/1), d=6.6.6.6, Len 1482, unknown protocol
```

```
.Nov  8 07:22:53.027: IP: s=7.0.0.1 (FastEthernet0/1), d=6.6.6.6, Len 1482, rcvd 4
.Nov  8 07:22:53.027: IP: s=7.0.0.1 (FastEthernet0/1), d=6.6.6.6, Len 1482, unknown protocol
.Nov  8 07:22:54.415: IP: s=7.0.0.1 (FastEthernet0/1), d=6.6.6.6, Len 1482, rcvd 4
.Nov  8 07:22:54.419: IP: s=7.0.0.1 (FastEthernet0/1), d=6.6.6.6, Len 1482, unknown protocol
```

Résumé

Àpartir des tests effectués, vous pouvez conclure que :

- Sur les circuits virtuels permanents lents sans WFQ, le trafic en masse affecte le petit trafic, comme les requêtes ping qui sont bloquées jusqu'à ce que WFQ soit activé.
- La taille de l'anneau de transmission (limite de sonnerie tx) détermine la vitesse à laquelle le mécanisme de mise en file d'attente commence à faire son travail. Vous pouvez voir l'impact de ceci avec l'augmentation de la requête ping RTT lorsque la limite de l'anneau de transmission augmente. Par conséquent, si WFQ ou LLQ doit être implémenté, il est logique de réduire la limite de l'anneau de transmission.
- WFQ qui utilise CBWFQ donne en effet la priorité aux petits trafics sur le trafic en masse.

Informations connexes

- Pages d'assistance technique ATM
- Présentation de la gestion des encombrements
- Support et documentation techniques - Cisco Systems