De betekenis van op klasse gebaseerde Weighted Fair Queuing op ATM

Inhoud

Inleiding

Dit document biedt een inleiding tot traffic shaping met behulp van class-Based Waing (CBWFQ)-technologie.

Weighted Fair Quinging (WFQ) stelt snelheidslinks, zoals seriële links, in staat om een eerlijke behandeling te bieden voor alle soorten verkeer. Het classificeert het verkeer in verschillende stromen (ook bekend als gesprekken) op basis van laag drie en laag vier informatie, zoals IP adressen en TCP poorten. Dit gebeurt zonder dat u toegangslijsten hoeft te definiëren. Dit betekent dat laagbandbreedteverkeer effectief prioriteit heeft boven hoogbandbreedteverkeer omdat het hoge bandbreedteverkeer de transmissiemedia in verhouding tot zijn toegewezen gewicht deelt. WFQ heeft echter bepaalde beperkingen:

- Het is niet schaalbaar indien het stroombedrag aanzienlijk stijgt.
- Native WFQ is niet beschikbaar op snelle interfaces zoals ATM-interfaces.

CBWFQ biedt een oplossing voor deze beperkingen. In tegenstelling tot standaard WFQ staat CBWFQ u toe om verkeersklassen te definiëren en parameters, zoals bandbreedte en wachtrij-limieten, op deze klassen toe te passen. De bandbreedte die u aan een klasse toevoegt wordt gebruikt om het "gewicht" van die klasse te berekenen. Het gewicht van elk pakje dat aan de klassecriteria voldoet, wordt ook hieruit berekend. De WFQ wordt toegepast op de klassen (die meerdere stromen kunnen omvatten) in plaats van de stromen zelf.

Klik op de volgende koppelingen voor meer informatie over het configureren van CBWFQ:

Op klasse gebaseerde Weighted Fair Queuing (Per-VC CBWFQ) op Cisco 7200, 3600 en 2600 routers.

Op klasse gebaseerde Weighted Fair Queuing op RSP-gebaseerde platforms

Voordat u begint

Conventies

Zie de Cisco Technical Tips Convention voor meer informatie over documentconventies.

Voorwaarden

Er zijn geen specifieke voorwaarden van toepassing op dit document.

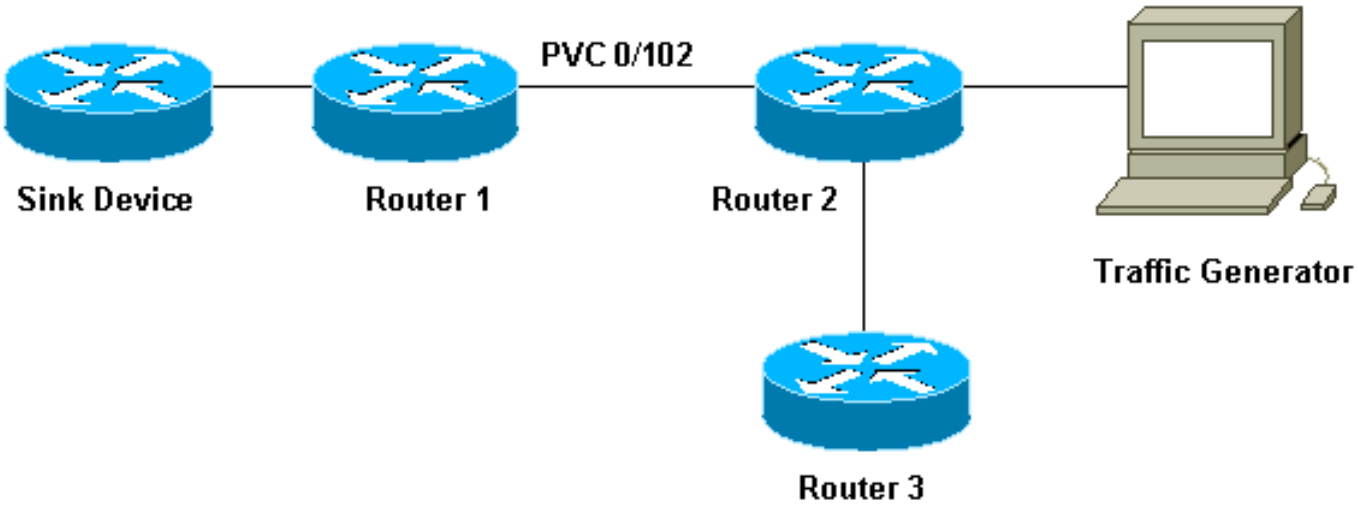
Gebruikte componenten

Dit document is niet beperkt tot specifieke software- en hardware-versies.

De informatie in dit document is gebaseerd op apparaten in een specifieke laboratoriumomgeving. Alle apparaten die in dit document worden beschreven, hadden een opgeschoonde (standaard)configuratie. Als u in een levend netwerk werkt, zorg er dan voor dat u de potentiële impact van om het even welke opdracht begrijpt alvorens het te gebruiken.

Netwerkdiagram

Laten we de volgende instellingen gebruiken om te illustreren hoe WFQ werkt:



In de instellingen die we hier gebruiken, kunnen pakketten worden opgeslagen in een van de volgende twee wachtrijen:

- De hardware eerst in first out (FIFO) wachtrij op de poortadapter en de netwerkmodule.
- De rij in de Cisco IOS® Software (op het router input/output [I/O] geheugen) waar Quality of Service (QoS) functies zoals CBWFQ kunnen worden toegepast.

De FIFO-wachtrij op de poortadapter slaat de pakketten op voordat deze in cellen worden gesegmenteerd voor verzending. Wanneer deze rij vol is, geven de poortadapter of de netwerkmodule signalen aan de IOS software die de rij verstopt is. Dit mechanisme heet "tegendruk". Na het ontvangen van dit signaal stopt de router met het verzenden van pakketten naar de interface-FIFO-wachtrij en slaat u de pakketten in de IOS-software op totdat de wachtrij opnieuw wordt onverstopt. Wanneer de pakketten in IOS worden opgeslagen, kan het systeem QoS eigenschappen zoals CBWFQ toepassen.

De transmissielimiet instellen

Eén probleem met dit wachtend mechanisme is dat, hoe groter de FIFO-wachtrij op de interface, hoe langer de vertraging voordat pakketten aan het einde van deze wachtrij worden verzonden, kan worden uitgesteld. Dit kan ernstige prestatieproblemen veroorzaken voor vertragingsgevoelig verkeer zoals spraakverkeer.

Met de opdracht permanent virtueel circuit (PVC) **tx-ring** kunt u de grootte van de FIFO-wachtrij beperken.

```
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
```

De limiet (x) die u kunt specificeren is een aantal pakketten (Cisco 2600 en 3600 routers) of een hoeveelheid deeltjes (Cisco 7200 en 7500 routers).

Het verminderen van de grootte van de stuurring heeft twee voordelen:

- De tijd dat pakketten in de FIFO-wachtrij worden geplaatst, wordt verkort voordat deze wordt

gesegmenteerd.
- Het versnelt het gebruik van QoS in de IOS-software.

Invloed van de transmissielimiet

Laten we het effect van de verzendring bekijken met behulp van de instellingen die in ons netwerkdiagram hierboven worden getoond. We gaan uit van het volgende:

- De verkeersgenerator zendt verkeer (1500 byte-pakketten) naar het gootsteenapparaat en dit verkeer overbelast PVC 0/102 tussen router1 en router2.
- Router3 probeert router1 te pingelen.
- CBWFQ is ingeschakeld op router 2.

Laten we nu naar twee configuraties kijken die verschillende radiogolven gebruiken om de impact te zien die dit heeft.

Voorbeeld

In dit voorbeeld, hebben we de zendring ingesteld op drie (**TX-ring-limit=3**). Dit is wat we zien wanneer we router1 van router3 ping.

```
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
```

Voorbeeld B

In dit voorbeeld, hebben we de zendring ingesteld op 40 (**TX-ring-limit=40**). Dit is wat we zien als we dezelfde ping gebruiken als in Voorbeeld 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
```

Zoals u hier kunt zien, hoe groter de limiet van de verzendring, hoe groter de 'ping round-trip' tijd (RTT). We kunnen hieruit concluderen dat een grote radiogrens kan leiden tot aanzienlijke vertragingen bij de transmissie.

Hoe CBWFQ werkt

Nu we de impact van de grootte van de hardware FIFO-wachtrij hebben gezien, zullen we precies zien hoe CBWFQ werkt.

Native WFQ wijst een gewicht aan elk gesprek toe, en plant dan de zendtijd voor elk pakket van de verschillende stromen. Het gewicht is een functie van de IP-voorrang van elke stroom en de planningstijd is afhankelijk van de pakketgrootte. Klik hier voor meer informatie over WFQ.

CBWFQ kent een gewicht toe aan elke geconfigureerde klasse in plaats van elke stroom. Dit gewicht is evenredig met de bandbreedte die voor elke klasse is ingesteld. Om precies te zijn, is het gewicht een functie van de interface bandbreedte gedeeld door de class bandbreedte. Hoe groter de bandbreedte parameter, hoe kleiner het gewicht.

We kunnen de pakketplanningstijd berekenen met de volgende formule:

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

Totale interfacebandbreedte-afdeling

Laten we kijken hoe de router de totale interfacebandbreedte tussen de verschillende klassen verdeelt. Om de klassen te onderhouden gebruikt de router kalender wachtrijen. Elk van deze kalenderwachtrijen slaat pakketten op die bij hetzelfde plan_tail_time moeten worden verzonden. De router stelt deze kalender dan één voor één samen. Laten we naar dit proces kijken:

1. Als er sprake is van stremmingen op de poortadapter wanneer een pakket arriveert op de uitvoerinterface, veroorzaakt dit een wachtrij voor IOS (CBWFQ in dit geval).
2. De router berekent een planningstijd voor dit aankomende pakket en slaat het op in de kalender die met deze planningstijd overeenkomt. Er kan slechts één pakket per klasse worden opgeslagen in een bepaalde kalender.
3. Wanneer het tijd is om de kalenderrij te bedienen waarin het pakket is opgeslagen, leegt IOS deze rij en stuurt de pakketten naar de FIFO-wachtrij op de poortadapter zelf. De grootte van deze FIFO-wachtrij wordt bepaald door de hierboven beschreven verzendring.
4. Als de FIFO-wachtrij te klein is om alle pakketten in de onderhouden kalender-wachtrij te kunnen plaatsen, herstelt de router de pakketten die niet voor de volgende planningstijd (overeenkomstig hun gewicht) kunnen worden opgeslagen en plaatst ze deze in de betreffende kalender-wachtrij.
5. Wanneer dit alles wordt gedaan, behandelt de poortadapter de pakketten in zijn FIFO rij en stuurt de cellen op de draad en de IOS beweegt naar de volgende kalender.Dankzij dit mechanisme ontvangt elke klasse statistisch een deel van de interface bandbreedte overeenkomend met de parameters die ervoor zijn ingesteld.

Mechanisme voor kalender-wachtrij versus grootte van doorgifte

Laten we kijken naar de relatie tussen het kalender-rijmechanisme en de grootte van de verzendring. Een kleine verzendring maakt het mogelijk om QoS sneller te starten en beperkt de latentie voor pakketten die wachten om te worden verzonden (wat belangrijk is voor vertragingsgevoelig verkeer zoals stem). Als de doorvoersnelheid echter te klein is, kan deze voor bepaalde klassen lager zijn. Dit komt doordat veel pakketten mogelijk moeten worden aangepast als de verzendring ze niet kan bevatten.

Helaas is er geen ideale waarde voor de grootte van de verzendring en de enige manier om de beste waarde te vinden is door te experimenteren.

Bandbreedteschatting

We kunnen het concept van het delen van bandbreedte bekijken met behulp van de instellingen die in ons netwerkdiagram worden getoond, hierboven. De pakketgenerator produceert verschillende stromen en stuurt deze naar het gootsteenapparaat. De totale hoeveelheid verkeer die door deze stromen wordt weergegeven, is genoeg om het PVC te overladen. We hebben CBWFQ op Router2 geïmplementeerd. Dit is hoe onze configuratie eruit ziet:

```
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
```

In ons voorbeeld is Router2 een Cisco 7200 router. Dit is belangrijk omdat de limiet van de verzendring wordt uitgedrukt in deeltjes en niet in pakketten. Packets worden in de FIFO-wachtrij van de poortadapter geplaatst zodra er een gratis deeltje beschikbaar is, zelfs als er meer dan één deeltje nodig is om het pakket op te slaan.

Wat is een Deeltaart?

In plaats van één stukje aangrenzend geheugen toe te wijzen voor een buffer, deelt het bufferen van deeltjes aan elkaar verwante (verspreide) geheugenstukken toe, noemde zij deeltjes, en koppelt ze vervolgens aan elkaar om één logische pakketbuffer te vormen. Dit heet een deeltjesbuffer. In zo'n schema kan een pakje dan over meerdere deeltjes worden verspreid.

In de 7200 router die we hier gebruiken, is de deeltjesgrootte 512 bytes.

We kunnen controleren of Cisco 7200 routers deeltjes gebruiken door de opdracht **showbuffers** te gebruiken:

```
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

De klassen "Klein" en "Groot" die we voor deze test gebruiken, zijn als volgt ingevuld:

- Kleine klasse - we hebben de bandbreedteparameters ingesteld op 32 kbps. Deze class slaat tien pakketten van 1500 bytes van 7.0.0.200 op, gevolgd door tien pakketten van 1500 bytes van 7.0.201
- Big class - we hebben de bandbreedte parameter op 16 kbps ingesteld. Deze class slaat één stroom van tien pakketten met 1500 bytes van 7.0.0.1 op.

De verkeersgenerator verstuurt een golf van verkeer voor het gootsteenapparaat met 100 Mbps naar Router2 in de volgende volgorde:

1. Tien pakketten van 7.0.0.1.
2. Tien pakketten van 7.0.200.
3. Tien pakketten van 7.0.201.

Het stroomgewicht controleren

Laten we kijken naar het gewicht dat wordt toegepast op de verschillende stromen. Om dit te doen, kunnen we de opdracht van de **showwachtrij van ATM x/y.z** gebruiken.

```
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
```

Wanneer alle pakketten van 7.0.0.200 uit de router in de wachtrij zijn geplaatst, kunnen we het volgende zien:

```
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
```

Zoals u hier kunt zien, hebben de stromen van 7.0.200 en 7.0.0.201 hetzelfde gewicht (128). Dit gewicht is de helft van het gewicht dat is toegewezen aan de stroom van 7.0.0.1 (256). Dit komt overeen met het feit dat onze kleine bandbreedte twee keer zo groot is als onze grote klasse.

De bandbreedtedistributie controleren

Hoe kunnen we de bandbreedteverdeling tussen de verschillende stromen controleren? De FIFO in de wachtrij wordt in elke klasse gebruikt. Onze kleine klasse is gevuld met tien pakketten van de eerste stroom en tien pakketten van de tweede stroom. De eerste stroom wordt bij 32 kbps uit de kleine klasse verwijderd. Zodra ze verzonden zijn worden ook de tien pakketten van de andere stroom verzonden. Intussen worden pakketten van onze grote klasse verwijderd op 16 kbps.

We kunnen zien dat, aangezien de verkeersgenerator een barst op 100 Mbps stuurt, het PVC zal worden overbelast. Aangezien er echter geen verkeer op het PVC is wanneer de test wordt gestart en aangezien de pakketten van 7.0.0.1 als eerste de router hebben bereikt, worden sommige pakketten van 7.0.0.1 verzonden voordat CBWFQ start vanwege congestie (met andere woorden voordat de verzendring volledig is).

Aangezien de deeltjesgrootte 512 bytes is en de grootte van de verzendring drie deeltjes is, kunnen we zien dat twee pakketten van 7.0.0.1 worden verzonden voordat er opstopping wordt. De ene wordt onmiddellijk op de draad verzonden en de tweede wordt opgeslagen in de drie deeltjes die de FIFO-wachtrij van de poortadapter vormen.

We kunnen de onderstaande specificaties op het gootsteenapparaat zien (dit is eenvoudigweg een router):

```
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
```

Aangezien de pakketgrootte voor beide stromen gelijk is, op basis van de planningstijdformule, moeten we twee pakketten van onze kleine klasse zien die voor elk pakket van onze grote klasse worden verzonden. Dit is precies wat we zien in de uitwerpselen hierboven.

Test B

Voor onze tweede test laten we de lessen op de volgende manier bevolken:

- Small class - we hebben de bandbreedte-parameter ingesteld op 32 kbps. Tien pakketten van 500 bytes van 7.0.0.200 worden gegenereerd, gevolgd door tien pakketten van 1500 bytes van 7.0.201.
- Big class - we hebben de bandbreedte parameter op 16 kbps ingesteld. De class slaat één stroom van 1500 bytes pakketten op die van 7.0.0.1 komen.

De verkeersgenerator verstuurt een doorbraak van verkeer met 100 Mbps naar Router2 in de volgende volgorde:

1. Tien pakketten van 1500 bytes van 7.0.0.1.
2. Tien pakketten van 500 bytes van 7.0.0.200.
3. Tien 1500 byte-pakketten van 7.0.201.

FIFO wordt in elke klasse ingesteld.

Het stroomgewicht controleren

De volgende stap is het op de gerubriceerde stromen toegepaste gewicht te verifiëren:

```
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,
```

Zoals u kunt zien in de bovenstaande output, hebben de stromen van 7.0.0.200 en 7.0.0.201 hetzelfde gewicht (128) ontvangen. Dit gewicht is de helft van de grootte van het gewicht dat is toegewezen aan de stroom van 7.0.0.1. Dit komt overeen met het feit dat kleine klasse twee keer zo groot is als onze grote klasse.

De bandbreedtedistributie controleren

We kunnen de volgende debugs maken van het gootsteenapparaat:

```
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
```

In dit scenario, hebben de stromen in onze kleine klasse niet de zelfde pakketgrootte. Daarom is de pakketdistributie niet zo triviaal als voor Test A, hierboven.

Schedulingstijden

Laten we de planningtijden voor elk pakje nader bekijken. De planningstijd voor pakketten wordt berekend met de volgende formule:

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

Voor verschillende pakketformaten gebruikt de planningtijd de volgende formule:

```
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
```

Uit deze formules kunnen we zien dat zes pakketten van 500 bytes van onze kleine klasse worden verzonden voor elk pakket van 1500 bytes van onze grote klasse (weergegeven in de debug uitvoer hierboven).

We kunnen ook zien dat twee pakketten van 1500 bytes van onze kleine klasse worden verzonden voor één pakket van 1500 bytes van onze grote klasse (weergegeven in de debug uitvoer hierboven).

Op basis van de bovenstaande testen kunnen we het volgende concluderen:

- De grootte van de zendring (TX-ring-limit) bepaalt hoe snel het wachtrijen mechanisme begint te werken. We kunnen het effect zien met de verhoging van de ping RTT als de limiet voor de verzendring toeneemt. Als u CBWFQ of Low Latency Queueing [LLQ] implementeert, moet u daarom overwegen de limiet voor de verzendring te verlagen.
- CBWFQ maakt eerlijk delen van de interface-bandbreedte tussen de verschillende klassen mogelijk.

Gerelateerde informatie

- Op klasse gebaseerde Weighted Fair Queuing (Per-VC CBWFQ) op Cisco 7200, 3600 en 2600 routers
- Op klasse 1-VC gebaseerde Weighted Fair Queuing op RSP-gebaseerde platforms
- De betekenis van Weighted Fair Queuing op ATM
- Ondersteuning van IP-naar-ATM serviceklasse
- Ondersteuning van ATM-technologie
- Technische ondersteuning en documentatie – Cisco Systems