ATM에서 클래스 기반 가중 공정 대기열 이해

목차

소개

이 문서에서는 클래스 기반 CBWFQ(Weighted Fair Queuing) 기술을 사용하여 트래픽 대기열을 소개합니다.

WFQ(Weighted Fair Queuing)는 직렬 링크와 같은 저속 링크를 통해 모든 유형의 트래픽에 대해 공평한 처리를 제공합니다. IP 주소 및 TCP 포트와 같은 레이어 3 및 레이어 4 정보를 기반으로 트래픽을 서로 다른 플로우(대화라고도 함)로 분류합니다. 액세스 목록을 정의할 필요 없이 이 작업을 수행합니다. 즉, 고대역폭 트래픽이 할당된 가중치에 비례하여 전송 미디어를 공유하기 때문에 저대역폭 트래픽이 고대역폭 트래픽보다 효과적으로 우선순위를 갖게 됩니다. 그러나 WFQ에는 몇 가지 제한 사항이 있습니다.

- 플로우 양이 상당히 증가하면 확장성이 없습니다.
- 네이티브 WFQ는 ATM 인터페이스와 같은 고속 인터페이스에서 사용할 수 없습니다.

CBWFQ는 이러한 제한 사항에 대한 솔루션을 제공합니다. 표준 WFQ와 달리 CBWFQ에서는 트래

픽 클래스를 정의하고 이러한 클래스에 대역폭 및 대기열 제한 등의 매개변수를 적용할 수 있습니다. 클래스에 할당하는 대역폭은 해당 클래스의 "가중치"를 계산하는 데 사용됩니다. 클래스 기준과 일치하는 각 패킷의 가중치도 여기에서 계산됩니다. WFQ는 흐름 자체가 아닌 클래스(여러 흐름을 포함할 수 있음)에 적용됩니다.

CBWFQ 구성에 대한 자세한 내용을 보려면 다음 링크를 클릭하십시오.

Cisco 7200, 3600 및 2600 라우터의 VC 클래스 기반 가중 공정 대기열(CBWFQ당)입니다.

RSP 기반 플랫폼의 VC 클래스 기반 가중 공정 대기열

시작하기 전에

표기 규칙

문서 규칙에 대한 자세한 내용은 Cisco 기술 팁 표기 규칙을 참조하십시오.

사전 요구 사항

이 문서에 대한 특정 요건이 없습니다.

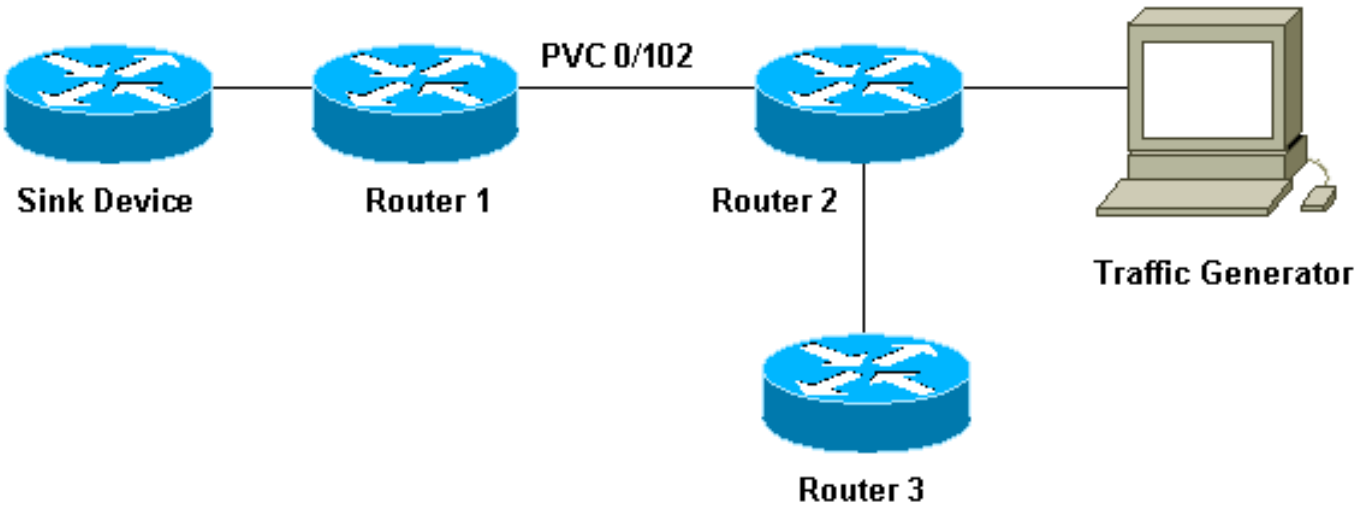
사용되는 구성 요소

이 문서는 특정 소프트웨어 및 하드웨어 버전으로 한정되지 않습니다.

이 문서의 정보는 특정 랩 환경의 디바이스를 토대로 작성되었습니다. 이 문서에 사용된 모든 디바이스는 초기화된(기본) 컨피그레이션으로 시작되었습니다. 라이브 네트워크에서 작업하는 경우, 사용하기 전에 모든 명령의 잠재적인 영향을 이해해야 합니다.

네트워크 다이어그램

WFQ의 작동 방식을 설명하기 위해 다음 설정을 사용하겠습니다.



여기서 사용하는 설정에서 패킷은 다음 두 큐 중 하나에 저장할 수 있습니다.

- 포트 어댑터 및 네트워크 모듈의 FIFO(First In First Out) 대기열에서 하드웨어의 첫 번째입니다 .
- CBWFQ와 같은 QoS(Quality of Service) 기능을 적용할 수 있는 Cisco IOS® Software(라우터 입력/출력[I/O] 메모리) 대기열

포트 어댑터의 FIFO 대기열은 패킷을 전송하기 위해 셀로 분할하기 전에 패킷을 저장합니다. 이 큐가 가득 차면 포트 어댑터 또는 네트워크 모듈이 IOS 소프트웨어에 대기열이 혼잡함을 알립니다. 이 메커니즘을 백압력이라고 합니다. 이 신호를 수신하면 라우터는 인터페이스 FIFO 대기열로 패킷을 전송하는 것을 중지하고 대기열이 다시 혼잡하지 않을 때까지 IOS 소프트웨어에 패킷을 저장합니다. 패킷이 IOS에 저장되면 시스템은 CBWFQ와 같은 QoS 기능을 적용할 수 있습니다.

전송 링 제한 설정

이 대기열 메커니즘의 한 가지 문제는 인터페이스에서 FIFO 대기열이 클수록 이 대기열 끝에 있는 패킷이 전송되기 전에 지연이 길어진다는 것입니다. 이로 인해 음성 트래픽과 같이 지연에 민감한 트래픽에 심각한 성능 문제가 발생할 수 있습니다.

PVC(Permanent Virtual Circuit) **tx-ring-limit** 명령을 사용하면 FIFO 대기열의 크기를 줄일 수 있습니다.

```
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
```

여기서 지정할 수 있는 제한(x)은 패킷 수(Cisco 2600 및 3600 라우터의 경우) 또는 입자 수량(Cisco 7200 및 7500 라우터의 경우)입니다.

전송 링의 크기를 줄이면 두 가지 이점이 있습니다.

- 분할되기 전에 FIFO 대기열에서 기다리는 패킷의 양을 줄입니다.
- IOS 소프트웨어에서 QoS 사용을 가속화합니다.

전송 링 제한의 영향

위 네트워크 다이어그램에 표시된 설정을 사용하여 전송 링 한도의 영향을 살펴보겠습니다. 다음과 같이 가정합니다.

- 트래픽 생성기가 트래픽(1500바이트 패킷)을 싱크 디바이스로 보내고 있으며, 이 트래픽은 router1과 router2 간에 PVC 0/102를 오버로드하고 있습니다.
- Router3에서 router1에 ping을 시도합니다.
- CBWFQ는 라우터 2에서 활성화됩니다.

이제 서로 다른 전송 링 제한을 사용하는 두 가지 컨피그레이션을 살펴보겠습니다. 이 컨피그레이션이 미치는 영향을 살펴보겠습니다.

예 A

이 예에서는 전송 링을 3으로 설정했습니다(**TX-ring-limit=3**). router3에서 router1을 ping할 때 표시되는 내용은 다음과 같습니다.

```
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
```

예 B

이 예에서는 전송 링을 40(**TX-ring-limit=40**)으로 설정했습니다. 예제 A와 동일한 ping을 사용할 때 표시되는 내용은 다음과 같습니다.

```
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
```

여기서 볼 수 있듯이 전송 링 한도가 클수록 ping 왕복 시간(RTT)이 큽니다. 따라서 전송 링 한도가 커지면 전송 지연이 크게 발생할 수 있다는 것을 알 수 있습니다.

CBWFQ 작동 방식

이제 하드웨어 FIFO 대기열의 크기가 미치는 영향을 살펴보았습니다. 이제 CBWFQ의 작동 방식을 정확히 살펴보겠습니다.

네이티브 WFQ는 각 대화에 가중치를 할당한 다음 서로 다른 흐름의 각 패킷에 대해 전송 시간을 예약합니다. 가중치는 각 흐름의 IP 우선 순위의 기능이며, 예약 시간은 패킷 크기에 따라 달라집니다. WFQ에 대한 자세한 내용을 보려면 여기를 클릭하십시오.

CBWFQ는 구성된 각 클래스에 각 흐름 대신 가중치를 할당합니다. 이 가중치는 각 클래스에 대해 구성된 대역폭에 비례합니다. 더 정확하게 가중치는 인터페이스 대역폭을 클래스 대역폭으로 나눈 기능입니다. 따라서 대역폭 매개변수가 클수록 가중치가 줄어듭니다.

다음 공식을 사용하여 패킷 예약 시간을 계산할 수 있습니다.

scheduling tail_time= queue_tail_time + pktsize * weight

총 인터페이스 대역폭 사업부

라우터가 서로 다른 클래스 간에 총 인터페이스 대역폭을 어떻게 분할하는지 살펴보겠습니다. 라우터는 클래스를 서비스하기 위해 일정 대기열을 사용합니다. 이러한 각 달력 대기열은 동일한 scheduling_tail_time에서 전송해야 하는 패킷을 저장합니다. 그런 다음 라우터는 이러한 일정 대기열을 한 번에 하나씩 서비스합니다. 이 프로세스를 살펴보겠습니다.

1. 패킷이 출력 인터페이스에 도착할 때 포트 어댑터에서 혼잡이 발생하면 IOS에서 큐잉이 발생합니다(이 경우 CBWFQ).
2. 라우터는 이 도착하는 패킷의 예약 시간을 계산하고 이 예약 시간에 해당하는 일정 대기열에 저장합니다. 특정 일정 대기열에는 클래스당 하나의 패킷만 저장할 수 있습니다.
3. 패킷이 저장된 달력 대기열을 서비스할 때 IOS는 이 대기열을 비우고 포트 어댑터 자체의 FIFO 대기열로 패킷을 전송합니다. 이 FIFO 대기열의 크기는 위에 설명된 전송 링 제한에 따라 결정됩니다.
4. FIFO 대기열이 너무 작아서 서비스된 달력 대기열에 있는 모든 패킷에 맞지 않으면 라우터는 다음 예약 시간(가중치에 해당)에 저장할 수 없는 패킷을 다시 예약하고 해당 패킷을 일정 대기열에 넣습니다.
5. 이 모든 작업이 완료되면 포트 어댑터는 FIFO 대기열의 패킷을 처리하고 전선의 셀을 전송하며 IOS가 다음 일정 대기열로 이동합니다.이 메커니즘으로 인해 각 클래스는 통계적으로 인터페이스 대역폭의 일부를 해당 클래스에 대해 구성된 매개변수에 따라 받습니다.

달력 대기열 메커니즘 대 전송 링 크기

달력 대기열 메커니즘과 전송 링 크기 간의 관계를 살펴보겠습니다. 작은 전송 링을 사용하면 QoS가 더 신속하게 시작되고 전송 대기 중인 패킷의 레이턴시를 줄일 수 있습니다(음성 등 지연에 민감한 트래픽의 경우 중요). 그러나 너무 작으면 특정 클래스의 처리량이 저하될 수 있습니다. 이는 전송 링이 패킷을 수용할 수 없을 경우 많은 패킷이 다시 예약해야 하기 때문입니다.

안타깝게도 전송 링 크기에 이상적인 값이 없으며, 가장 좋은 값을 찾는 유일한 방법은 실험을 통한 것입니다.

대역폭 공유

위의 네트워크 다이어그램에 표시된 설정을 사용하여 대역폭 공유 개념을 살펴볼 수 있습니다. 패킷 생성기는 서로 다른 흐름을 생성하여 싱크 디바이스로 전송합니다. 이러한 플로우로 표시되는 총 트래픽 양은 PVC를 오버로드하기에 충분합니다. Router2에 CBWFQ를 구현했습니다. 이 컨피그레이션은 다음과 같습니다.

```
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
```

이 예에서 Router2는 Cisco 7200 라우터입니다. 이는 전송 링 한도가 패킷이 아니라 입자로 표현되기 때문에 중요합니다. 패킷을 저장하는 데 둘 이상의 파티클이 필요한 경우에도 사용 가능한 파티클이 있는 즉시 패킷이 포트 어댑터 FIFO 대기열에서 대기됩니다.

입자란?

입자 버퍼링은 버퍼에 대해 하나의 인접 메모리를 할당하는 대신 입자라고 하는 비연속(산산) 메모리 조각을 할당한 다음 이를 연결하여 하나의 논리적 패킷 버퍼를 형성합니다. 이것을 입자 버퍼라고 합니다. 이러한 체계에서 패킷은 여러 입자에 걸쳐 분산될 수 있습니다.

여기서 사용하는 7200 라우터에서 입자 크기는 512바이트입니다.

show buffers 명령을 사용하여 Cisco 7200 라우터에서 입자를 사용하는지 확인할 수 있습니다.

```
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
```

테스트 A

이 테스트에 사용하는 "Small" 및 "Big" 클래스는 다음과 같은 방법으로 채워집니다.

- 소규모 클래스 - 대역폭 매개변수를 32kbps로 구성했습니다. 이 클래스는 7.0.0.200에서 1500바이트의 10개의 패킷을 저장하고, 그 다음에 7.0.0.201에서 1500바이트의 10개의 패킷을 저장합니다.
- 빅 클래스 - 대역폭 매개변수를 16kbps로 구성했습니다. 이 클래스는 7.0.0.1에서 1500바이트 패킷 10개의 흐름을 저장합니다.

트래픽 생성기는 100Mbps의 싱크 디바이스로 향하는 트래픽의 버스트를 다음 순서로 Router2로 전송합니다.

1. 7.0.0.1에서 10개 패킷
2. 7.0.0.200에서 10개 패킷
3. 7.0.0.201에서 10개 패킷

플로우 중량 확인

다른 흐름에 적용되는 무게를 살펴보겠습니다. 이를 위해 **show queue ATM x/y.z** 명령을 사용할 수 있습니다.

```
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
```

7.0.0.200의 모든 패킷이 라우터에서 대기된 경우 다음을 확인할 수 있습니다.

```
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
```

여기서 볼 수 있듯이 7.0.0.200과 7.0.0.201의 흐름은 동일한 무게(128)를 가집니다. 이 가중치는 7.0.0.1(256)에서 흐름에 할당된 가중치의 절반 크기입니다. 이것은 우리의 작은 클래스 대역폭이 우리의 빅 클래스 크기의 두 배라는 사실에 해당합니다.

대역폭 분배 확인

그러면 서로 다른 플로우 간의 대역폭 분포를 어떻게 확인할 수 있을까요? FIFO 대기열 처리 방법은 각 클래스에 사용됩니다. 우리의 작은 클래스는 첫 번째 흐름에서 10개의 패킷으로 채워지고 두 번째 흐름에서 10개의 패킷으로 채워진다. 첫 번째 흐름은 32kbps로 소규모 클래스에서 제거됩니다. 전송되면 다른 흐름에서 보낸 10개의 패킷도 전송됩니다. 그 사이에, 우리 빅클래스의 패킷은 16kbps로 제거됩니다.

트래픽 생성기가 100Mbps의 버스트를 전송하므로 PVC는 오버로드됩니다. 그러나 테스트가 시작될 때 PVC에 트래픽이 없으므로 7.0.0.1의 패킷이 라우터에 처음 도달하기 때문에 혼잡(즉, 전송 링이 가득 차기 전에) CBWFQ가 시작되기 전에 7.0.0.1의 일부 패킷이 전송됩니다.

입자 크기는 512바이트이고 전송 링 크기는 3개의 입자이므로 혼잡이 발생하기 전에 7.0.0.1에서 보낸 2개의 패킷이 전송되는 것을 확인할 수 있습니다. 하나는 즉시 전선에 전송되고 다른 하나는 포트 어댑터 FIFO 대기열을 형성하는 세 개의 입자에 저장됩니다.

싱크 디바이스(단순히 라우터인)에서 아래의 디버그를 확인할 수 있습니다.

```
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
```

스케줄링 시간 공식을 기준으로 두 흐름의 패킷 크기가 동일하므로, Cisco의 소규모 클래스에서 오는 두 패킷이 빅클래스로부터 각 패킷에 대해 전송되는 것을 확인해야 합니다. 이것이 바로 위의 디버그에서 볼 수 있는 것입니다.

테스트 B

두 번째 테스트에서는 다음과 같이 클래스를 채우도록 하겠습니다.

- Small class - 대역폭 매개변수를 32kbps로 구성했습니다. 7.0.0.200에서 500바이트의 10개의 패킷이 생성되고, 7.0.0.201에서 1500바이트의 10개의 패킷이 생성됩니다.
- 빅 클래스 - 대역폭 매개변수를 16kbps로 구성했습니다. 이 클래스는 7.0.0.1에서 오는 1500바이트 패킷의 한 흐름을 저장합니다.

트래픽 생성기는 다음 순서로 100Mbps의 트래픽 버스트를 Router2로 전송합니다.

1. 7.0.0.1에서 1500바이트 패킷 10개
2. 7.0.0.200에서 500바이트 패킷 10개
3. 7.0.0.201에서 1500바이트 패킷 10개

FIFO는 각 클래스에 구성됩니다.

플로우 중량 확인

다음 단계는 분류된 흐름에 적용되는 가중치를 확인하는 것입니다.

```
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,
```

위 출력에서 볼 수 있듯이 7.0.0.200 및 7.0.0.201의 플로우는 동일한 가중치(128)를 받았습니다. 이 가중치는 7.0.0.1에서 흐름에 할당된 가중치의 절반 크기입니다. 이것은 작은 클래스가 큰 클래스의 두 배 크기의 대역폭을 가지고 있다는 사실에 해당합니다.

대역폭 분배 확인

싱크 디바이스에서 다음 디버그를 생성할 수 있습니다.

```
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
```

이 시나리오에서 소규모 클래스의 플로우는 패킷 크기가 다릅니다. 따라서 패킷 분배는 위의 테스트 A에 비해 간단한 것이 아닙니다.

예약 시간

각 패킷의 예약 시간을 자세히 살펴보겠습니다. 패킷의 예약 시간은 다음 공식을 사용하여 계산됩니다.

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

여러 패킷 크기의 경우 예약 시간은 다음 공식을 사용합니다.

```
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
```

이러한 공식을 통해, 위의 디버그 출력에서 보여주는 것처럼, 빅 클래스에서 1500바이트의 각 패킷에 대해 소규모 클래스에서 500바이트의 6개의 패킷이 전송되는 것을 확인할 수 있습니다.

또한 Small 클래스에서 1500바이트의 2개의 패킷이 빅 클래스에서 1500바이트의 1패킷에 대해 전송된다는 것을 확인할 수 있습니다(위의 디버그 출력에 표시됨).

위의 테스트에서 다음과 같은 결과를 얻을 수 있습니다.

- 전송 링(TX-ring-limit)의 크기에 따라 큐 메커니즘의 작동 속도가 결정됩니다. 전송 링 제한이 증가할 때 ping RTT가 증가하면 그 영향을 확인할 수 있습니다. 따라서 CBWFQ 또는 LLQ(Low Latency Queuing)를 구현하는 경우 전송 링 제한을 줄이는 것을 고려하십시오.
- CBWFQ를 사용하면 서로 다른 클래스 간에 인터페이스 대역폭을 균등하게 공유할 수 있습니다.

관련 정보

- Cisco 7200, 3600 및 2600 라우터의 VC 클래스 기반 가중 공정 대기열(CBWFQ당)

- RSP 기반 플랫폼의 VC 클래스 기반 가중 공정 대기열
- ATM에서 가중치 공정 대기열 이해
- IP-ATM Class of Service 기술 지원
- ATM 기술 지원
- 기술 지원 및 문서 – Cisco Systems