了解 ATM 上基于类的加权公平排队

目录

简介

本文档介绍使用基于类的加权公平队列(CBWFQ)技术的流量队列。

加权公平队列(WFQ)使慢速链路（如串行链路）能够为所有类型的流量提供公平处理。它根据第3层和第4层信息（如IP地址和TCP端口）将流量分类为不同的流（也称为会话）。它无需您定义访问列表即可执行此操作。这意味着低带宽流量实际上比高带宽流量具有优先级，因为高带宽流量与其分配的权重成正比地共享传输介质。但是，WFQ有某些限制：

- 如果流量显着增加，则无法扩展。
- 本地WFQ在高速接口（如ATM接口）上不可用。

CBWFQ提供了解决这些限制的解决方案。与标准WFQ不同，CBWFQ允许您定义流量类并将带宽和队列限制等参数应用到这些类。分配给某个类的带宽用于计算该类的“权重”。此外，还会根据此计算与类标准匹配的每个数据包的权重。WFQ应用于类（可以包含多个流），而不是流本身。

有关配置CBWFQ的详细信息，请点击以下链接：

Cisco 7200、3600和2600路由器上基于每VC类的加权公平队列（每VC CBWFQ）。

基于RSP的平台上基于每VC类的加权公平队列。

开始使用前

规则

有关文档规则的详细信息，请参阅 Cisco 技术提示规则。

先决条件

本文档没有任何特定的前提条件。

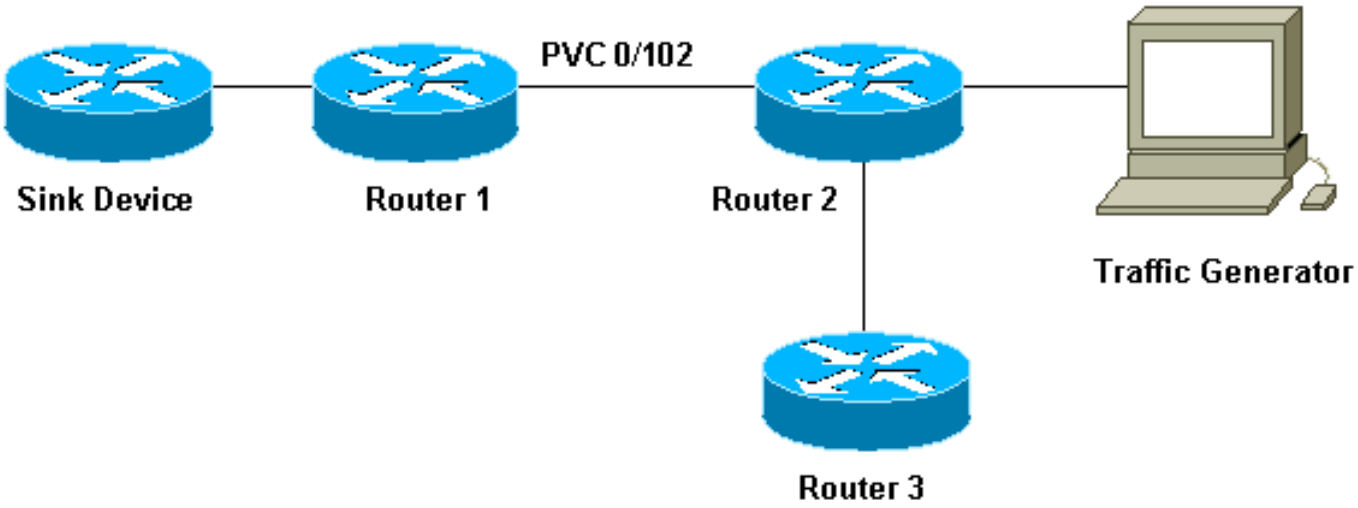
使用的组件

本文档不限于特定的软件和硬件版本。

本文档中的信息都是基于特定实验室环境中的设备创建的。本文档中使用的所有设备最初均采用原始（默认）配置。如果您是在真实网络上操作，请确保您在使用任何命令前已经了解其潜在影响。

网络图

为了说明WFQ如何工作，我们使用以下设置：



在此处使用的设置中，数据包可以存储在以下两个队列之一：

- 端口适配器和网络模块上的硬件先进先出(FIFO)队列。
- Cisco IOS®软件（在路由器输入/输出[I/O]内存上）中的队列，其中可以应用服务质量(QoS)功能（如CBWFQ）。

端口适配器上的FIFO队列在数据包被分段为信元以供传输之前存储这些数据包。当此队列已满时，端口适配器或网络模块会向IOS软件发出队列拥塞的信号。这种机制称为背压。收到此信号后

，路由器停止向接口FIFO队列发送数据包，并将数据包存储在IOS软件中，直到队列再次不拥塞。当数据包存储在IOS中时，系统可以应用QoS功能，如CBWFQ。

设置传输环路限制

此排队机制的一个问题是，接口上的FIFO队列越大，此队列末尾的数据包传输前的延迟就越长。这可能导致延迟敏感型流量（如语音流量）出现严重的性能问题。

永久虚电路(PVC)tx-ring-limit**命令**使您能够减小FIFO队列的大小。

```
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
```

您可以在此指定的限制(x)是数据包数（对于Cisco 2600和3600路由器）或粒子数（对于Cisco 7200和7500路由器）。

减小传输环的大小有两个优点：

- 它减少了数据包在分段之前在FIFO队列中等待的时间。
- 它加快了IOS软件中QoS的使用。

传输环路限制的影响

让我们使用上面网络图中显示的设置来查看传输环限制的影响。我们可以假设：

- 流量生成器正在向接收设备发送流量（1500字节数据包），此流量使路由器1和路由器2之间的PVC 0/102过载。
- Router3尝试ping router1。
- 路由器2上启用了CBWFQ。

现在，让我们看看使用不同传输环限制的两种配置，以了解其影响。

示例 A

在本例中，我们将传输环设置为三(**TX-ring-limit=3**)。 从router3 ping router1时，我们会看到以下内容。

```
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
```

示例 B

在本例中，我们将传输环设置为40(**TX-ring-limit=40**)。 以下是我们使用与示例A中相同的ping时看到的内容：

```
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
```

如图所示，传输环限制越大，ping往返时间(RTT)就越大。 由此可以推断，较大的传输环限制可能导致传输中的显着延迟。

CBWFQ 如何工作

既然我们已经了解了硬件FIFO队列大小的影响，让我们来看看CBWFQ的具体工作方式。

本地WFQ为每个会话分配权重，然后为不同流的每个数据包安排传输时间。权重是每个流的IP优先级的函数，而调度时间取决于数据包大小。单击此处了解有关WFQ的详细信息。

CBWFQ为每个已配置的类（而不是每个流）分配权重。此权重与为每个类配置的带宽成比例。更精确地说，权重是接口带宽除以类带宽的函数。因此，带宽参数越大，权重越小。

我们可以使用以下公式计算数据包调度时间：

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

总接口带宽划分

让我们看看路由器如何划分不同类之间的接口总带宽。为了为类提供服务，路由器使用日历队列。每个日历队列都存储必须在同一scheduling_tail_time上传输的数据包。然后，路由器会逐个为这些日历队列提供服务。让我们来了解一下此过程：

1. 如果数据包到达输出接口时端口适配器上发生拥塞，这会导致IOS（本例中为CBWFQ）中的队列。
2. 路由器计算到达的数据包的调度时间，并将其存储在与此调度时间对应的日历队列中。每个类只能存储一个数据包在特定日历队列中。
3. 当需要为存储数据包的日历队列提供服务时，IOS会清空此队列，并将数据包发送到端口适配器自身的FIFO队列。此FIFO队列的大小由上述传输环限制确定。

4. 如果FIFO队列太小，无法容纳在所服务的日历队列中的所有数据包，则路由器重新安排不能存储到下一个调度时间（与其权重相对应）的数据包，并将它们放入相应的日历队列。
5. 完成所有这些后，端口适配器将处理其FIFO队列中的数据包，并发送线路上的信元，IOS将移至下一个日历队列。由于这种机制，每个类都会统计地接收与为其配置的参数对应的部分接口带宽。

日历队列机制与传输环大小

我们来了解一下日历队列机制与传输环大小之间的关系。小的传输环使QoS能够更快地启动并减少等待传输的数据包的延迟(这对延迟敏感型流量（如语音）非常重要)。 但是，如果它太小，则可能导致某些类的吞吐量较低。这是因为，如果传输环无法容纳大量数据包，可能必须重新安排大量数据包。

遗憾的是，对于传输环大小，没有理想的值，而找到最佳值的唯一方法就是实验。

带宽共享

我们可以使用上面网络图中显示的设置来了解带宽共享的概念。数据包生成器生成不同的流并将其发送到接收设备。这些流量代表的总流量足以使PVC过载。我们已在Router2上实施CBWFQ。以下是我们的配置：

```
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
```

在本例中，Router2是Cisco 7200路由器。这很重要，因为传输环限制以粒子而非数据包表示。当空闲粒子可用时，即使需要多个粒子来存储数据包，数据包也会在端口适配器FIFO队列中排队。

什么是微粒？

粒子缓冲不是为缓冲区分配一个连续的内存，而是分配不连续（分散的）内存片段，称为粒子，然后将它们链接在一起以形成一个逻辑数据包缓冲区。这称为粒子缓冲。在这种方案中，分组随后可以跨多个粒子分布。

在我们使用的7200路由器中，粒度为512字节。

我们可以使用**show buffers**命令验证Cisco 7200路由器是否使用**粒子**:

```
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
```

测试 A

我们用于此测试的“小”和“大”类按以下方式填充：

- 小类 — 我们已将带宽参数配置为32 kbps。此类存储10个1500字节的数据包（从7.0.0.200开始），后跟1500字节的10个数据包（从7.0.0.201开始）
- 大类 — 我们已将带宽参数配置为16 kbps。此类存储来自7.0.0.1的10个1500字节数据包的流。

流量生成器以100 Mbps的速率将发往接收设备的突发流量发送到Router2，顺序如下：

1. 来自7.0.0.1的10个数据包。
2. 来自7.0.0.200的10个数据包。
3. 来自7.0.0.201的10个数据包。

验证流权重

让我们看看应用于不同流的权重。为此，我们可以使用**show queue ATM x/y.z**命令。

```
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
```

当来自7.0.0.200的所有数据包排队出路由器时，我们可以看到以下内容：

```
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
```

如图所示，从7.0.0.200和7.0.0.201的流具有相同的重量(128)。 此重量是分配给流量的重量(从7.0.0.1(256))的一半。 这与我们的小类带宽是大类带宽的两倍这一事实相对应。

验证带宽分配

那么，如何验证不同流之间的带宽分布呢？FIFO排队方法用于每个类。我们的小类包含来自第一个流的十个数据包和来自第二个流的十个数据包。第一个流以32 kbps的速率从小类中删除。发送后，来自另一流的十个数据包也会发送。同时，来自我们大类的数据包以16 kbps的速度被删除。

我们可以看到，由于流量生成器以100 Mbps的速率发送突发，PVC将过载。但是，由于测试开始时PVC上没有流量，并且由于来自7.0.0.1的数据包是第一个到达路由器的数据包，因此，由于拥塞（换句话说，在传输环已满之前），在CBWFQ启动之前，将发送来自7.0.0.1的一些数据包。

由于粒度为512字节，传输环大小为3个粒子，因此我们可以看到，在发生拥塞之前，会发送来自7.0.0.1的两个数据包。一个立即发送到线上，第二个存储在形成端口适配器FIFO队列的三个粒子中。

我们可以在接收设备（仅是路由器）上看到以下调试：

```
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
```

由于两个流的数据包大小相同，因此根据调度时间公式，我们应该看到来自小类的两个数据包针对来自大类的每个数据包发送。这正是我们在上述调试中看到的。

测试 B

对于第二个测试，我们按以下方式填充类：

- 小类 — 我们已将带宽参数配置为32 kbps。从7.0.0.200生成10个500字节的数据包，然后是7.0.0.201生成1500字节的10个数据包。
- 大类 — 我们已将带宽参数配置为16 kbps。该类存储来自7.0.0.1的1500字节数据包的一个流。

流量生成器按以下顺序将突发流量以100 Mbps的速率发送到Router2:

1. 10个来自7.0.0.1的1500字节数据包。
2. 来自7.0.0.200的10个500字节数据包。
3. 10个来自7.0.0.201的1500字节数据包。

FIFO在每个类中配置。

验证流权重

下一步是验证应用于分类流的权重：

```
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,
```

如上面的输出所示，来自7.0.0.200和7.0.0.201的流量已收到相同的权重(128)。 此权重是分配给流量的权重的一半大小（从7.0.0.1开始）。这与小类的带宽是大类的两倍这一事实相对应。

验证带宽分配

我们可以从接收设备生成以下调试：

```
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
```

在此场景中，小类中的流的数据包大小不同。因此，数据包分布不像上面的测试A那样简单。

调度时间

让我们详细了解一下每个数据包的调度时间。数据包的调度时间使用以下公式计算：

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

对于不同的数据包大小，调度时间使用以下公式：

```
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
```

从这些公式中，我们可以看到，小类中的6个500字节数据包是为大类中每个1500字节的数据包传输

的（如上面的调试输出所示）。

我们还可以看到，小类中两个1500字节的数据包是为大类中一个1500字节的数据包发送的（如上面的调试输出所示）。

通过上述测试，我们可以得出以下结论：

- 传输环的大小（TX环限制）决定了排队机制开始工作的速度。当传输环限制增加时，我们可以看到ping RTT的增加对网络的影响。因此，如果实施CBWFQ或低延迟排队[LLQ])，请考虑减少传输环限制。
- CBWFQ允许不同类之间公平共享接口带宽。

相关信息

- Cisco 7200、3600和2600路由器上基于每VC类的加权公平队列（每VC CBWFQ）
- 基于RSP平台上基于每VC类的加权公平队列
- 了解 ATM 上的加权公平排队
- IP到ATM服务类技术支持
- ATM技术支持
- 技术支持和文档 - Cisco Systems