ATM でのクラスベース重み付け均等化キューイングについて

内容

概要

この文書では、Class-Based Weighted Fair Queuing (CBWFQ; クラスベース均等化キューイング) テクノロジーを使用したトラフィック キューイングの概要を説明します。

Weighted Fair Queuing (WFQ; 均等化キューイング) を使用すると、シリアル リンクなどの低速リンクは、すべてのタイプのトラフィックを公平に扱うことができます。WFQ は、IP アドレスや TCP ポートなどのレイヤ 3 およびレイヤ 4 の情報に基づいて、トラフィックをさまざまな (カンバセーションとも呼ばれる) フローに分類します。これには、アクセスリストを定義する必要はありません。このことは、高帯域幅トラフィックは、割り当てられた重みに比例する伝送メディアを共有しているため、事実上、低帯域幅トラフィックは高帯域幅トラフィックよりも優先度が高いことを意味します。ただし、WFQ にはいくつかの制限があります。

- フローの量がかなり多くなると、拡張性がなくなります。
- ATM インターフェイスなどの高速インターフェイスでは、ネイティブ WFQ は使用できません。

CBWFQ は、これらの制限に対する解決策となります。標準的な WFQ とは異なり、CBWFQ を使用すると、トラフィック クラスを定義し、これらのクラスに対して bandwidth や queue-limit などのパラメータを適用することができます。クラスに割り当てた帯域幅は、そのクラスの「重み」を計算するために使用されます。また、この値からクラスの基準を満たす各パケットの重みも計算されます。WFQ は、フロー自体ではなく、（複数のフローを含むことができる）クラスに適用されます。

CBWFQ の設定の詳細は、次のリンクを参照してください。

Cisco 7200、3600、および 2600 ルータでの、VC 単位、クラスベース均等化キューイング（Per-VC CBWFQ）

RSP ベースのプラットフォームでの、VC 単位、クラスベース均等化キューイング

はじめに

表記法

ドキュメント表記の詳細は、『シスコ テクニカル ティップスの表記法』を参照してください。

前提条件

このドキュメントに関しては個別の前提条件はありません。

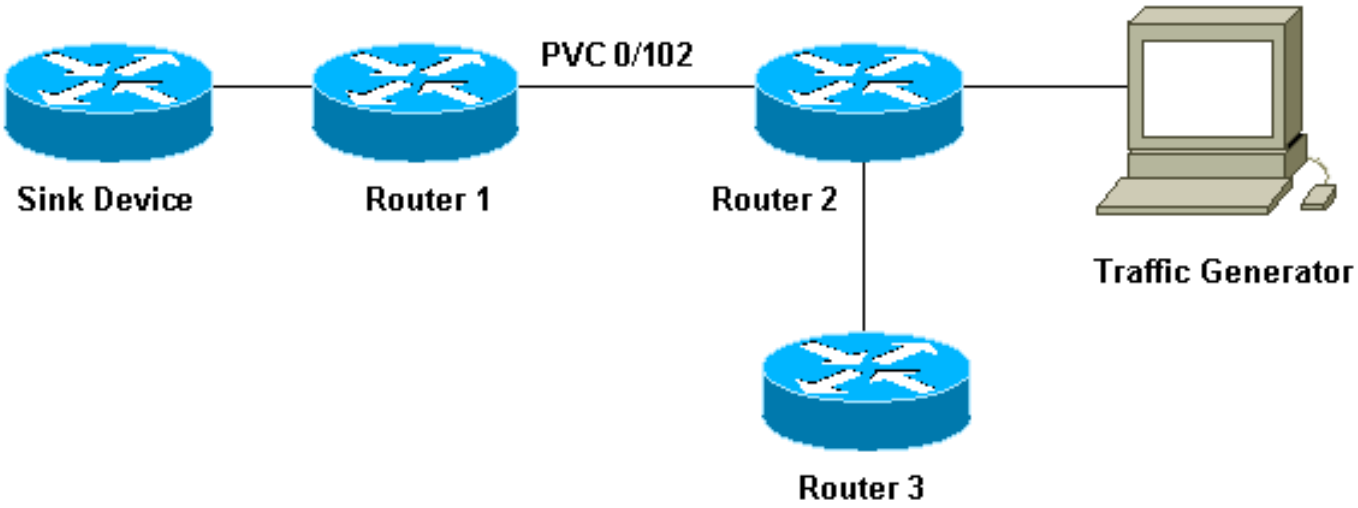
使用するコンポーネント

このドキュメントの内容は、特定のソフトウェアやハードウェアのバージョンに限定されるものではありません。

このマニュアルの情報は、特定のラボ環境に置かれたデバイスに基づいて作成されました。このドキュメントで使用するすべてのデバイスは、初期（デフォルト）設定の状態から起動しています。実稼動中のネットワークで作業をしている場合、実際にコマンドを使用する前に、その潜在的な影響について理解しておく必要があります。

ネットワーク図

WFQ の動作を詳しく説明するために、次の設定を使用します。



この設定では、パケットは次の 2 つのキューのいずれかに格納することができます。

- ポート アダプタおよびネットワーク モジュールの、ハードウェア First In First Out (FIFO; 先入れ先出し) キュー。
- CBWFQなどのQuality of Service(QoS)機能を適用できるCisco IOS®ソフトウェア(ルータの入出力(I/O)メモリ上)のキュー。

ポート アダプタの FIFO キューには、伝送用のセルにセグメント化される前のパケットが格納されます。このキューがいっぱいである場合、ポート アダプタまたはネットワーク モジュールは、IOS ソフトウェアに対して、キューに輻輳が生じているという信号を出します。このメカニズムは、バックプレッシャと呼ばれます。この信号を受信すると、ルータはこのインターフェイスの FIFO キューへのパケットの送信を停止し、キューの輻輳状態が緩和されるまでこのパケットを IOS ソフトウェアに保存します。パケットが IOS に格納されている場合は、システムは CBWFQ などの QoS 機能を適用できます。

送信リングの制限の設定

このキューイング メカニズムに関連する問題の 1 つは、インターフェイス上の FIFO キューが大きいほど、このキューの最後にあるパケットが送信されるまでの遅延時間が長くなることです。これにより、音声トラフィックなどの遅延の影響を受けやすいトラフィックには、重大なパフォーマンス問題が生じます。

Permanent Virtual Circuit (PVC; 相手先固定接続) の tx-ring-limit コマンドを使用すると、FIFO キューのサイズを小さくすることができます。

```
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
```

このコマンドで指定できる制限 (x) は、パケットの数 (Cisco 2600 および 3600 ルータの場合) 、またはパーティクルの数量 (Cisco 7200 および 7500 ルータの場合) です。

送信リングのサイズを小さくすることには、次の 2 つの利点があります。

- パケットがセグメント化されるまでの FIFO キューでの待ち時間が短くなります。

- IOS ソフトウェアでの QoS の使用が促進されます。

送信リングの制限の影響

上記のネットワーク ダイアグラムに示す設定を使用して、送信リングの制限の影響について調べます。次のような動作を仮定します。

- トラフィック ジェネレータからシンク デバイスにトラフィック（1500 バイトのパケット）が送られ、このトラフィックによりルータ 1 とルータ 2 との間の PVC 0/102 で過負荷状態が発生しています。
- ルータ 3 からルータ 1 に ping が送られます。
- router2 では CBWFQ が有効になっています。

ここで、異なる送信リングの制限を使用している 2 つの設定を調べ、その影響を見てみましょう。

例 A

この例では、送信リングを3(**TX-ring-limit=3**)に設定しています。 router3 から router1 へ ping を送信した場合の出力を、次に示します。

```
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
```

例 B

この例では、送信リングを40(**TX-ring-limit=40**)に設定しています。 例 A と同じ ping を実行したときの結果を次に示します。

```
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
```

上記の出力からわかるように、送信リングの制限が大きくなれば、ping の Round-Trip Time (RTT; ラウンドトリップ時間) も長くなります。 このことから、送信リングの制限が大きいことにより、転送に著しい遅延が生じる場合があることが分かります。

CBWFQ の機能

ハードウェア FIFO キュー のサイズの影響は確認できたので、CBWFQ の動作の仕組みを正確に調べます。

ネイティブの WFQ は、各カンバセーションに重みを割り当ててから、さまざまなフローの各パケットの送信時間をスケジュールします。重みは各フローの IP 優先順位の関数で、スケジュール時間はパケット サイズに依存します。WFQ の詳細については、ここをクリックしてください。

CBWFQ では、各フローの代わりに、設定された各クラスに重みを割り当てます。この重みは、各クラスに設定された帯域幅に比例します。より正確には、重みは、インターフェイス帯域幅をクラス帯域幅で除算した関数になります。そのため、bandwidth パラメータが大きくなれば、重みが小さくなります。

次の式を使用すると、パケットのスケジュール時間を計算できます。

scheduling tail_time= queue_tail_time + pktsize * weight

全インターフェイス帯域幅分割

ここでは、ルータが合計インターフェイス帯域幅をさまざまなクラス間でどのように分割するかを調べます。クラス に応対するため、ルータはカレンダ キューを使用します。これらのカレンダ キューはそれぞれ、同じ scheduling_tail_time に送信する必要があるパケットを格納します。続いてルータは、 1 度に 1 つずつこれらのカレンダ キューに応対します。次に、このプロセスについて説明します。

1. パケットが出力インターフェイスに到達した時点でポート アダプタで輻輳が発生した場合、これにより IOS (この場合は CBWFQ) でキューイングが生じます。
2. ルータは、この着信パケットのスケジュール時間を計算し、このスケジュール時間に対応するカレンダ キューに、着信パケットを格納します。特定のカレンダ キューには、1 クラスにつき 1 つのパケットのみ格納することができます。
3. パケットが格納されているカレンダ キューに応対する時間になれば、IOS はこのキューを空にして、ポート アダプタ自体の FIFO キューにパケットを送信します。FIFO キューのサイズは、上記の送信リングの制限によって決まります。
4. FIFO キューが小さ過ぎるため、応対すべきカレンダ キューに格納されているすべてのパケットを格納できない場合は、ルータは、格納できないパケットを次のスケジュール時間に向けて (重みに応じて) 再スケジュールし、一致するカレンダ キューにそれらを配置します。
5. この作業がすべて完了すると、ポート アダプタは FIFO キューにあるパケットを処理し、セルをワイヤーで送出します。また IOS は次のカレンダ キューに移動します。このメカニズムにより、各クラスは、クラスに対して設定されているパラメータに応じたインターフェイス帯域幅の一部を、統計に基づいて受け取ります。

カレンダ キュー メカニズムと送信リング サイズ

カレンダ キュー メカニズムと送信リング サイズとの間にある関係について調べます。送信リングのサイズが小さいと、QoS をより迅速に開始し、送信を待機するパケットの遅延を短くすることができます (音声など、遅延の影響を受けやすいトラフィックにとって重要です)。 ただし、サイズが小さ過ぎると、一部のクラスのスループットが低下する原因になることがあります。これは、送信リングがパケットに対応できないと、数多くのパケットを再スケジュールしなければならない場合があるためです。

残念なことに、送信リング サイズの理想値は存在せず、最適値を見つけるには実験してみるしかありません。

帯域幅の共有

上記のネットワーク ダイアグラムに示す設定を使用した、帯域幅の共有の概念を調べます。パケット ジェネレータはさまざまなフローを生成し、それらを受信装置に送信します。これらのフローによるトラフィックの合計量は、PVC を過負荷状態にするのに十分です。Router2にCBWFQを実装しました。設定は次のようになります。

```
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
```

この例では、router2 は Cisco 7200 ルータです。このことが重要であるのは、送信リングの制限はパケットではなく、パーティクルで表されているためです。パケットの格納に複数のパーティクルが必要な場合であっても、空きパーティクルが使用可能になるとすぐ、パケットはポート アダプタ FIFO キューにキューイングされます。

パーティクルとは何か

パーティクル バッファリングでは、連続したメモリの一区画がバッファに割り当てられるのではなく、複数のパーティクルと呼ばれる不連続 (分散した状態) なメモリ区画が割り当てられます。そしてこれらがつなぎ合わされて、論理的な 1 つのパケット バッファが形成されます。これはパーティクル バッファと呼ばれます。このような方法では、1 つのパケットが複数のパーティクルに分けられます。

ここで使用している 7200 ルータでは、パーティクル サイズは 512 バイトになっています。

show buffers コマンドを使用すると、Cisco 7200 ルータがパーティクルを使用しているかどうかを調べることができます。

```
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
```

テスト A

このテストで使用している「スモール」および「ビッグ」のクラスは、次のようになります。

- スモール クラス ： bandwidth パラメータの設定は 32 kbps です。このクラスは、7.0.0.200 から 1500 バイトのパケットを 10 個格納し、後に 7.0.0.201 からの 1500 バイトのパケット 10 個が続きます。
- ビッグ クラス ： bandwidth パラメータの設定は 16 kbps です。このクラスは、7.0.0.1 からの 1500 バイトのパケット 10 個からなる 1 フローを格納します。

トラフィック ジェネレータは、次の順序で、受信装置を送信先としたトラフィックのバーストを、100 Mbps で router2 に送信します。

1. 7.0.0.1 からのパケット 10 個。
2. 7.0.0.200 からのパケット 10 個。
3. 7.0.0.201 からのパケット 10 個。

フローの重みの検証

さまざまなフローに適用される重みを調べます。これには、show queue ATM x/y.z コマンドを使用します。

```
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
```

7.0.0.200 からのすべてのパケットがルータの外にキューイングされた場合は、次のような出力になります。

```
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
```

上記の出力からわかるように、7.0.0.200 と 7.0.0.201 からのフローは、同じ重み（128）を持っています。この重みは、7.0.0.1 からのフローに割り当てられた重み（256）の半分のサイズです。このことは、スモール クラスの帯域幅は、ビッグ クラスのサイズの 2 倍であるという事実と一致します。

帯域幅配分の検証

さまざまなフロー間の帯域幅分配を検証する方法を考えてみます。各クラスでは、FIFO キューイング方法が使用されています。スモール クラスは、最初のフローからの 10 個のパケットと、2 番目のフローからの 10 個のパケットで満たされています。最初のフローは、32 kbps でスモール クラスから送信されます。パケットが送信されるとすぐに、もう 1 つのフローからの 10 個のパケットも送信されます。その間、ビッグ クラスからのパケットは 16 kbps で送信されます。

トラフィック ジェネレータは 100 Mbps でバーストを送信しているため、PVC が過負荷状態になることが確認できます。ただし、テスト開始時点に PVC にはトラフィックがなく、7.0.0.1 からのパケットがルータに到達する最初のパケットであるため、輻輳が原因で CBWFQ が開始する前（つまり送信リングがいっぱいになる前）に、7.0.0.1 からの一部のパケットが送信されます。

パーティクルのサイズが 512 バイトで、送信リング サイズはパーティクル 3 つであるため、輻輳が発生する前に 7.0.0.1 からの 2 つのパケットが送信されることが確認できます。1 つはただちにワイヤーに送出され、2 番目は、ポート アダプタ FIFO キューを形成する 3 つのパーティクルに格納されます。

受信装置（単なるルータ）でのデバッグは、次のようになることが確認できます。

```
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
```

両方のフローのパケット サイズは同じであるため、スケジュール時間の式に基づいて、ビッグ クラスから 1 パケットが送信されるたびに、スモール クラスから 2 つのパケットが送信されることが確認できるはずです。このことは、確かに上記のデバッグで確認できます。

テスト B

第 2 のテストでは、クラスを次のようにします。

- スモール クラス ： bandwidth パラメータの設定は 32 kbps です。7.0.0.200 から 500 バイトのパケットが 10 個生成され、後に 7.0.0.201 からの 1500 バイトのパケット 10 個が続きます。
- ビッグ クラス ： bandwidth パラメータの設定は 16 kbps です。このクラスは、7.0.0.1 からの 1500 バイトのパケットを 1 フロー格納します。

トラフィック ジェネレータは、次の順序で、トラフィックのバーストを 100 Mbps で router2 に送信します。

1. 7.0.0.1 から 1500 バイトのパケット 10 個。
2. 7.0.0.200 から 500 バイトのパケット 10 個。
3. 7.0.0.201 から 1500 バイトのパケット 10 個。

各クラスで FIFO が設定されています。

フローの重みの検証

次のステップでは、分類されたフローに適用される重みを検証します。

```
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,
```

上記の出力からわかるように、7.0.0.200 と 7.0.0.201 からのフローは、同じ重み（128）を受信しています。 この重みは、7.0.0.1からのフローに割り当てられた重みの半分のサイズです。これは、小さいクラスが大きなクラスの2倍のサイズの帯域幅を持つことに対応します。

帯域幅配分の検証

受信装置からのデバッグは、次のようになります。

```
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
```

このシナリオでは、スモール クラスのフローは同じパケット サイズを持っていません。そのためパケット分配は、上記のテスト A のように単純にはなりません。

スケジュール時間

各パケットのスケジュール時間を詳しく調べます。パケットのスケジュール時間は、次の式を使用して計算します。

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

パケット サイズが異なる場合、スケジュール時間は次の式を使用します。

```
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
```

これらの式から、（上記のデバッグ出力に示すように）ビッグ クラスからの 1500 バイトの 1 パケットに対して、スモール クラスから 500 バイトのパケット 6 個が送信されることが確認できます。

また、（上記のデバッグ出力に示すように）ビッグ クラスからの 1500 バイトの 1 つのパケットに対して、スモール クラスから 1500 バイトのパケット 2 つが送信されることも確認できます。

上記のテストから、次の結論が得られます。

- 送信リングのサイズ（TX-ring-limit）は、キューイング メカニズムがどれだけ迅速に動作を開始するかを決定します。送信リングの制限が大きくなった場合、ping RTT の増加に伴う影響を確認できます。したがって、CBWFQ または 低遅延キューイング（LLQ）を実装する場合は、送信リングの制限を小さくすることを考慮します。
- CBWFQ により、さまざまなクラス間でインターフェイス帯域幅を公平に共有できます。

関連情報

- Cisco 7200、3600、および2600ルータでのVC単位クラスベース重み付け均等化キューイング（VC単位CBWFQ）
- RSPベースプラットフォームでのVC単位クラスベース均等化キューイング
- ATM での均等化キューイングについて
- IP to ATM Class of Service（CoS；サービスクラス）テクノロジーのサポート
- ATM テクノロジーに関するサポート
- テクニカル サポートとドキュメント – Cisco Systems