ATM での均等化キューイングについて

目次

概要

このドキュメントでは、重み付け均等化キューイング（WFQ）テクノロジーを使用するトラフィック キューイングの概要について説明します。

WFQ はすべてのトラフィック の 種類に公正な判定を提供することをシリアルリンクのような低速リンクが、可能にするために導入されました。 これを実現するために、WFQ では、IP アドレスや TCP ポートなどのレイヤ 3 およびレイヤ 4 の情報をベースとして、トラフィックをさまざまなフローに分類します（会話とも呼ばれます）。 それはあなたの要件なしでアクセス リストを定義するためにこれをします。 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。

しかし、WFQ に一定の制限があります:

- 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。

Class-based weighted fair queuing（CBWFQ; クラスベース重み付け均等化キューイング）では、これらの制限に対する解決策が提供されています。

標準の WFQ とは異なり、CBWFQ ではトラフィック クラスを定義できます。 またパラメータを、それらへの帯域幅および queue-limit のような、適用できます。 クラスに割り当てる帯域幅はそのクラスのウエイトを計算するために使われます。 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 は複数のフローを含むことができるクラスにフロー自身よりもむしろそれから適用されます。

CBWFQ を設定する方法に関する詳細についてはこれらの文書を参照して下さい:

- VC 単位のクラス ベース、均等化キューイング（VC 単位の CBWFQ） on Cisco 7200、3600、および 2600 人のルータ
- RSP ベースのプラットフォームでの、VC 単位、クラスベースの重み付け均等化キューイング

ATM インターフェイスは **fair-queue コマンド**でインターフェイスで直接設定されるネイティブ フロー ベース WFQ をサポートしません。 しかし、ソフトウエアで CBWFQ をサポートする、この例に示すようにデフォルト クラス内のフロー ベース WFQ を、設定できます:

```
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
```

前提条件

要件

このドキュメントに関する固有の要件はありません。

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。

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。

ネットワーク図

WFQ がどのようにはたらくか説明するためにこのセットアップを使用して下さい:

このセットアップでは、パケットはこれら二つのキューの 1 つで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）キュー
- Cisco IOS ® ソフトウェアのキュー、CBWFQ のような Quality of Service（QoS）機能が適

用しますルータ 入出力[I/O]メモリで、
ポート アダプタにある FIFO キューには、パケットが送信用にセルにセグメント化される前に格納されます。 このキューがいっぱいになると、ポート アダプタまたはネットワーク モジュールから IOS ソフトウェアに対して、キューが輻輳しているという信号が出されます。 このメカニズムは、バックプレッシャと呼ばれます。 この場合の受信でインターフェイス FIFOキューにパケットを送信するために、ルータは停止し、IOS software でキューが再度輻輳状態でなくなるまでパケットを蓄えます。 パケットが IOS に保存されているときには、システムは QOS を適用できます。

送信リング限界を設定 する方法

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 <size> service-policy output test
```

ここに規定できる *<size>* いくつかのパケット、Cisco 2600 および 3600 ルータのために、または Cisco 7200 および 7500 ルータのためのパーティクルの数量、です。

送信リングのサイズのリダクションに 2 つの利点があります:

- それは FIFOキューでセグメント化される前にパケット時間数の待っています減らします。
- 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

前のネットワークダイアグラムで表示されるセットアップを使用する送信リング限界の影響を検知 して下さい。 これらを仮定して下さい:

- トラフィックジェネレータはシンク デバイスにトラフィック (1500 のバイトパケット) を送信 し、このトラフィックは router1 と router2 間の PVC 0/102 を過剰にします。
- ルータ 3 からルータ 1 に ping が送られます。
- ルータ 2 では WFQ が有効にされています。

2 コンフィギュレーションを検知 して下さいこれが持っている影響ことを見るために異なる送信リング限界を使用する。

例 A

この例では、3 に送信リングを設定 しました (TX-ring-limit=3)。 これは router3 から router1 を ping するとき見るものです:

```
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 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.
```

例 B

この例では、40 に送信リングを設定 しました（TX-ring-limit=40）。 これは同じ PING 次例 A を使用するとき見るものです:

```
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 ms
```

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。 大きい送信リング限界が伝達のかなりの遅延の原因となる場合があることをこれから推論できます。

ウエイトを計算する方法

例 A で出力される **show queue ATM** ではウエイトが各メッセージ交換に割り当てられることがわかります。 これをより詳しく検知 して下さい:

```
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
```

WFQ を使用するとき、この数式の使用の各メッセージ交換のウエイトを計算できます:

- **weight=32384/(precedence+1)** - Cisco IOS ソフトウエア Release 12.0(5)T と それ以降のための…。
- 重み = 4096 / (優先順位 + 1) - Cisco IOS ソフトウェア リリース 12.0(5)T 以前の場合。

スケジューリングタイムを計算する方法

パケットが IOS キューからポート アダプタまたはネットワークモジュール FIFOキューへの転送されるとき各パケットのスケジューリングタイムを計算するためにこれらのウエイトを使用するようになることができます。

queue_tail_time が現在のスケジューリングタイムであるこの数式の使用の出力スケジューリング時間を計算できます:

出力スケジューリング時間 = queue_tail_time + パケットサイズ * 重み

WFQ の動作の仕組み

このセクションは WFQ がどのようにはたらくか説明します。 WFQ の原則は送信 されるとき小さい重量のパケット、か小さいパケットが、優先順位を得る必要があることです。

フローを作成すれば 10 のパケット大きいパケットから成り立つその (82 バイト) の 4 つのより小さいパケットはこれを確認するためにトラフィックジェネレータを使用します。

この例では、router2 は PA-A3 （ ATM ポートアダプタ) の Cisco 7200 ルータです。 これはポート アダプタの出力 FIFOキューのサイズがパーティクルとないパケットに表現されるので重要です。 パーティクルはである何参照して下さいか。 詳細についてはセクション。

パーティクルとは何か

バッファのための連続するメモリの一つのアロケーションの代りに、パーティクル バッファリングはパーティクルと呼ばれるメモリの隣接しない (分散させた) ピースを割り当て次に 1 つの論理的なパケット バッファを形成するためにリンクします。 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。

7200 ルータでは、パーティクルサイズは 512 バイトです。

Cisco 7200 ルータ 使用粒子かどうか確かめるために **show buffers コマンドを**使用して下さい:

```
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
```

テスト A

これらはいくつかのテスト WFQ 機能性を説明するためにです。 この最初テストでは、帯域幅が異なるメッセージ交換の間で共有することができるかどうか外観の。

このテストでは router1 と router2 間の PVC 0/102 を過剰にするには、トラフィックジェネレータ送信トラフィックを十分にすぐに作りました。 router3 から同じ PVC を渡る router1 に PING を行って下さい:

```
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
```

示されている出力からわかるように WFQ がインターフェイスで有効に なるまで、トラフィックは他のトラフィックが防ぎ、行を行くことを飢えさせる。 WFQ が有効にされると、直ちに ping は成功します。

、WFQ の使用と、帯域幅が他をブロックする 1 つなしで異なるメッセージ交換の間で共有することができるこれから見ることができます。

テスト B

これは帯域幅がどのように共有されるかです。

トラフィックジェネレータが送信 する フローは 82 バイトの 4 つのより小さいパケットによって 10 の大きいパケットで、続かれて構成されるバーストです。 router2 に 100 Mbps でこれを送信します。 バーストを送信 するとき、router2 ATMインターフェイスの出力キューは空です。 Router2 は 10 KB PVC によって輻輳が出力キューに発生するようにするためにこれらのパケットを (これは非常に遅い PVC です) 送信します。

このプロセスを簡素化するために複数のステージのこのテストを行なって下さい:

第 1 段階

大きなトラフィックには、482 バイトのパケットが 10 個含まれています。 PA-A3 のパーティクルは 512 バイトであるため、各パケットは、大きなものでも小さなものでも、ポート アダプタの出力キューに格納されるときには 1 つのパーティクルを使用します。 ルータの送信リングの制限は 3 です (tx-ring-limit=3) 。 これはシンク デバイスで次のように表示できるものをの例です:

```
   .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
```

普通優先順位を得る必要がある 82 バイト パケットの前に送信 される 482 バイトの 4 つのパケットを表示できます。 こういうわけでこれは起こります。

まず、このバーストは 482 バイトのパケット 10 個で構成されているため、ルータに最初に到達し、その後に 82 バイトのパケットが続きます。 輻輳がない時 482-byte パケットが着くので、トラフィックがないので、1 パケットはセルに chunked、送出されるポート アダプタ

Segmentation And Reassembly（SAR）にすぐにワイヤー キューに入ります。 すなわち、送信リングはまだ空のままです。

総バーストを送信 するためにことを必要な時間が必要なトラフィックジェネレータの時間より大きい 1 482-byte パケットを送信 するために計算できます。 従って最初の 482-byte パケットがポート アダプタにキューに入るとき、バーストのより多くの 482-byte パケットは既にルータにあっていると仮定できます。 よって、その他の 482 バイトのパケットは、送信リングにキューされます。 482 バイトの 3 つのより多くのパケットは現在の 3 つのフリーパーティクルのそこの使用とキューに入ります。

注: パケットの格納に複数のパーティクルが必要とされている場合でも、空きパーティクルが 1 つでもあれば、直ちに送信リングにパケットがキューされます。

この時点で 3 つのパーティクルがいっぱいになったため、輻輳状態になります。 したがって、IOS でのキューイングが始まります。 4 つの 82 バイトのパケットがようやくルータに到達した時点では、輻輳が発生しています。 これらの 4 つのパケットはキューイングされて、この 2 つのフローに対して WFQ が使用されます。 ATM キューを検知 して下さいこれを見るために **show queue atm コマンドを**使用する:

```
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
```

482 バイトの最初の 4 つのパケットが 82 バイト パケットに先行していることがデバッグでわかります。 これらの小さなパケットは、大きなパケットより先にルータから送信されます。 このことは、輻輳が発生すると、直ちに小さなパケットに対して大きなパケットより高い優先順位が与えられることを意味しています。

与えられるこれを確認するために重量 セクションの計算でウエイトおよびスケジューリングタイム数式を使用して下さい。

ステージ 2

5 への送信リング限界をおよび高めれば大きいパケットは前の出力へ輻輳が発生する前に調和のそれから 482 バイト、82 バイトの 4 つのパケットに先行している 482 バイトの 6 つのパケットを見るはずですそして 482 バイトのもう 4 つのパケットです:

```
.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
```

ここを見てわかるように、これは全く何が起こるかです。

ステージ 3

パーティクル サイズは 512 バイトです。 従って送信リングがパーティクルに表現されたら、およびあなた大きいパケットをわずかに使用して下さいパーティクルサイズが、各自 2 つのパーティクルを奪取 するより。 これは 582 バイトのパケットおよび 3 の送信リングの使用によって説明されます。 これらのパラメータによって、582 バイトの 3 つのパケットを見るはずです。 1 つは ATMインターフェイスのトラフィック無しで、それ空けておきます 3 つのパーティクルを送信されます。 したがって、その他の 2 つのパケットがキューイングされて、その後に 82 バイトのパケット 4 つが続き、さらにその後に 582 バイトのパケット 7 つが続きます。

```
.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
```

ステージ 4

1482 のパケットサイズを (3 つのパーティクル) 奪取 し、5 の送信リングを定義して下さい。 送信リングがパーティクルで定義される場合、類似した何かを見ます:

- すぐに送信される 1 パケット

- 1 パケット 5 つのパーティクルの 3 つを奪取 する
- 2 つのパーティクルが自由であるのでキューに入る 1 パケット

```
.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
```

要約

遂行されるテストからこれらを完了できます:

- WFQ のない遅い PVC で、バルク トラフィックは WFQ が有効に なるまで停止する ping のような小さいトラフィックに、影響を与えます。
- 送信リング（tx-ring-limit）のサイズはすぐにキューイング機構がジョブをどのようにし始めるか判別します。 送信リング限界が増加するとき PING RTT の増加とこれの影響を表示できます。 したがって、WFQ または LLQ を実装する必要がある場合は、送信リングの制限を小さくする方が合理的です。
- CBWFQ を使用する WFQ は全くバルク トラフィック上の小さいトラフィックに優先順位をつけます。

関連情報

- ATM テクノロジーに関するサポート ページ
- 輻輳管理の概要
- テクニカルサポートとドキュメント - Cisco Systems