QoS 階層型スケジューリング

この章では、スケジューリングの章で説明したように、コマンドとそのセマンティクスをさまざまな方法で組み合わせて、より複雑な結果を達成する方法を説明します。

複雑なスケジューリング階層を設定するには、階層型ポリシーマップと論理インターフェイスに適用されたポリシーマップの2つの異なるアプローチがあります。ここでは、どちらの方法で同じ結果が得られるかを示し、各アプローチの相対的な利点を説明します。

- 階層型スケジュールについて （1 ページ）
- 階層型スケジューリングの操作 （7 ページ）
- 優先度の伝播 （13 ページ）
- リーフ スケジュールにおける bandwidth コマンド （19 ページ）
- Bandwidth コマンドはローカルでのみ有効 （24 ページ）
- 論理インターフェイスに適用されたポリシーマップ （29 ページ）
- 階層型ポリシーマップ （41 ページ）
- 確認 （50 ページ）

階層型スケジュールについて

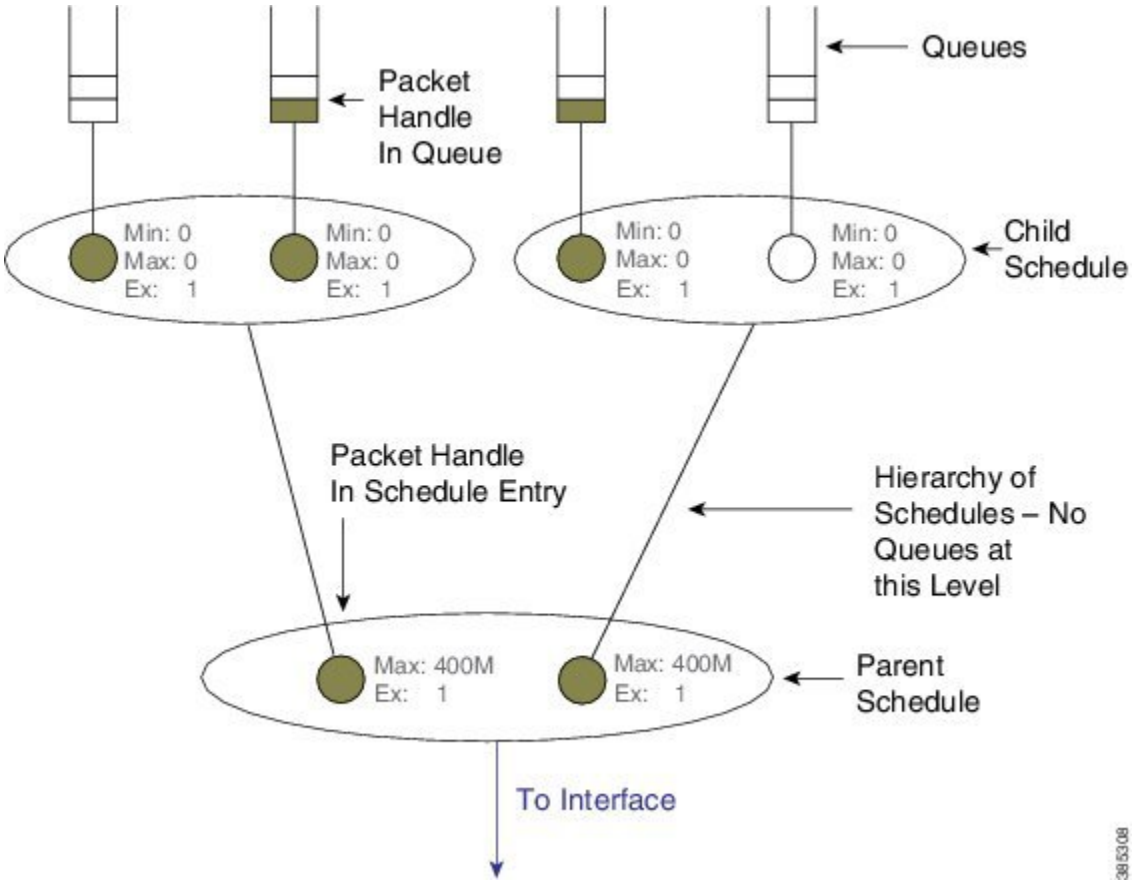
定義

スケジュールの役割と、そのエントリの子がどのように処理されるかに関する情報（パケット処理、クラス キューなど）をスケジュール エントリにどのように格納するかを理解していることを前提としています（スケジュールの章の「定義」を参照）。ここでは、その議論を基にしていきます。

ここで示した内容と前の章の根本的な違いは、子スケジュールにあり、キューまたは別のスケジュールである可能性があります。階層型スケジューリングにより、複数のレイヤで帯域幅を共有し、複雑な構造を構築できます。

次の図は、基本的な階層型スケジューリング構造を示しています。

図 *1:* 階層型スケジューリングの定義



図から最初に気付くことは、キューの階層ではなく、スケジュールの階層を実装していることです。これは、キューが階層のリーフ レイヤのみに存在し、パケット処理（パケット表記ビークル）がキューからキューに移動することはないことを意味します。代わりに、1 つのパケット処理が親スケジュール エントリ（パケットが送信を待機していることを前提とした場合）にロードされます。

スケジューリング階層を詳述する場合、スケジュールを親または子（または実際には孫）として記述します。これらの記述は相対的なものです。親スケジュールは、階層のルートに一段近い（インタフェースに近い）スケジュールです。 スケジュールの子は、スケジュールまたはキューのいずれかになります。スケジュールをリーフ スケジュールまたは非リーフ スケジュールと呼ぶこともあります。 リーフ スケジュール には、子としてのキューのみがあります。非リーフ スケジュールには、子供として少なくとも 1 つのスケジュールがあります。

図を見ると、親スケジュール（非リーフ）のスケジュール エントリには、スケジュール エントリごとに2つのパラメータしかしかないことがわかります。最小帯域幅パラメータはリーフ スケジュールでのみサポートされ、非リーフ スケジュールではサポートされません。

スケジューリングの決定：ルートからリーフへ

次の一連の図は、階層内のスケジュールがどのように協調して機能し、さらにインターフェイスを介して送信する次のパケットを選択する際にローカルな決定を下すかを示しています。

ローカルに格納されているパケットの中で、親スケジュールはまず、インターフェイスに転送する最も適格なパケットを選択します。関連付けられたパケット処理を送信した後、独自のスケジュールエントリに空きスポットが生成されます。そのエントリの子からのパケット処理は存在しません。

子が別のスケジュールである場合、親は *pop*（「最も適格なパケットを選び、そのパケット処理を私に送ってください」と伝えるメッセージ）を送信します。子スケジュールは、各エントリの設定を確認し、次に送信するパケットを決定し、そのパケット処理を親スケジュールに転送します。これで子には、そのスケジュール エントリのフリー スポットが生成されます。子がキューであるため、決定は必要ありません。キューの先頭にあるパケット処理は、（子）スケジュール エントリにロードされます。

図 *2:* スケジューリングの決定：ルートからリーフへ、ステップ *1* ～ *2*

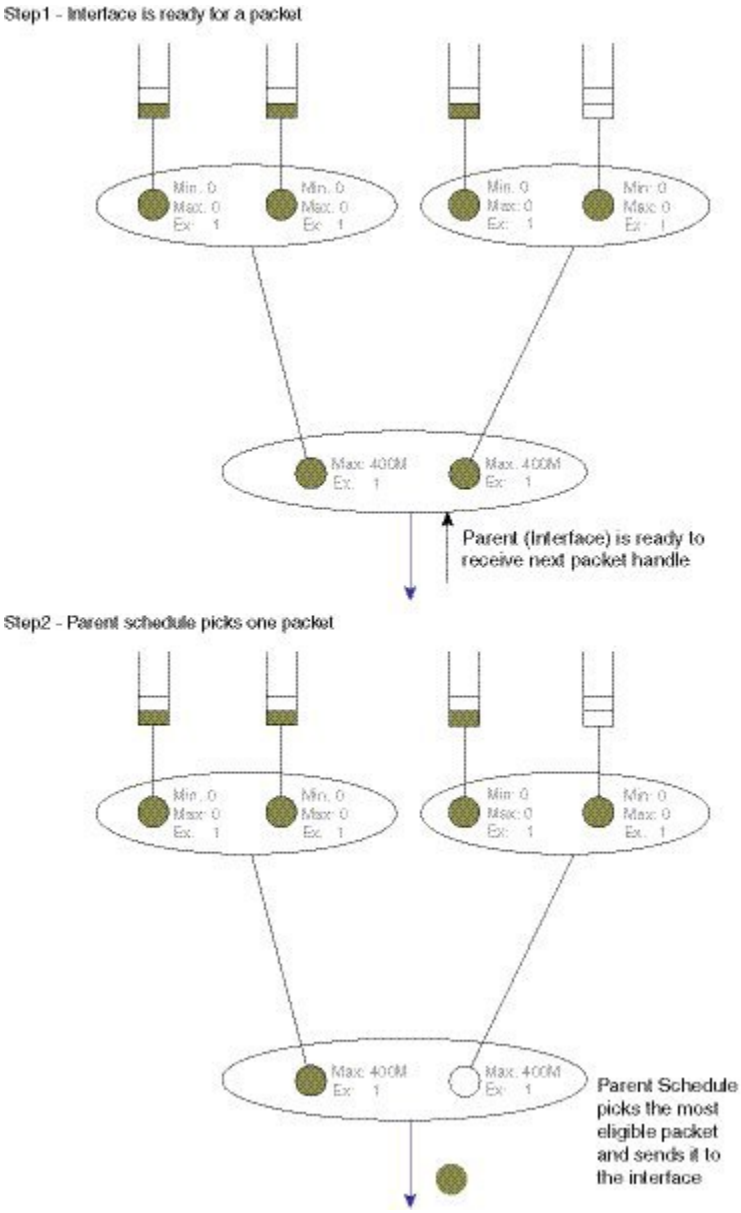


図 *3 :* スケジューリングの決定：ルートからリーフへ、ステップ *3* ～ *4*

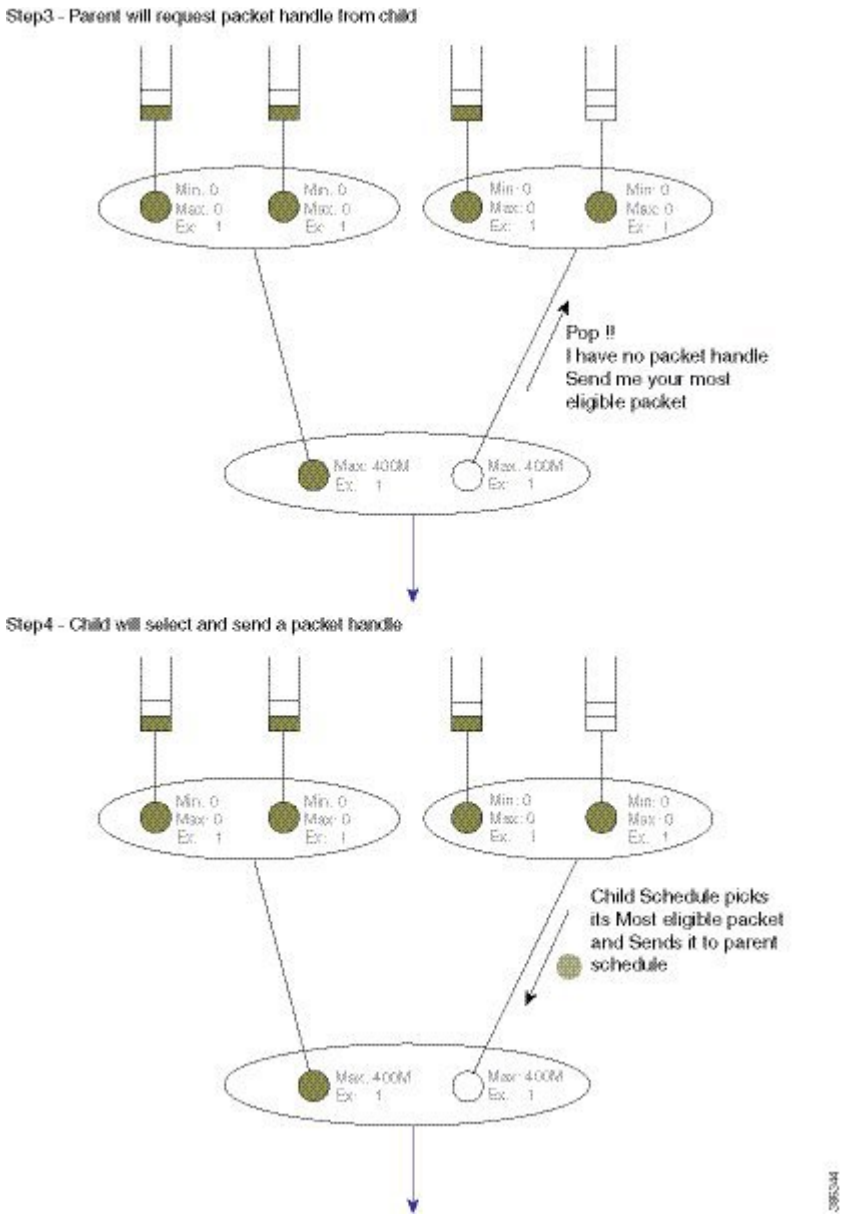
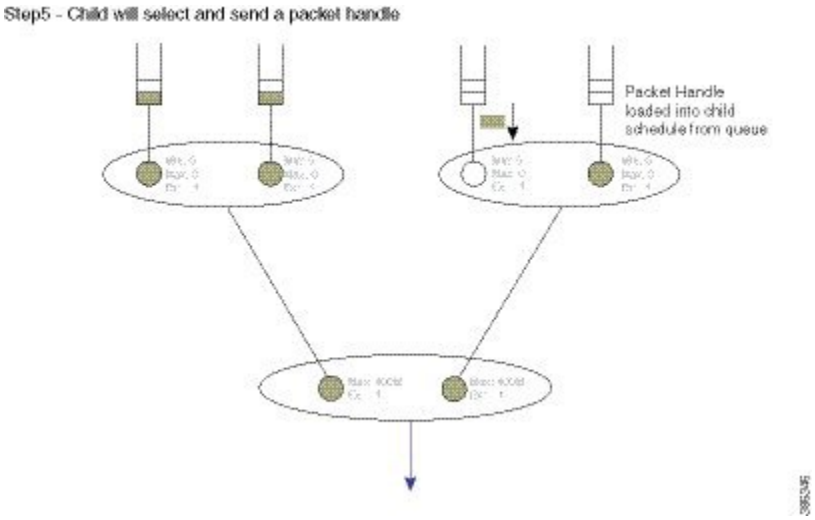


図 *4:* スケジューリングの決定：ルートからリーフへ、ステップ *5*

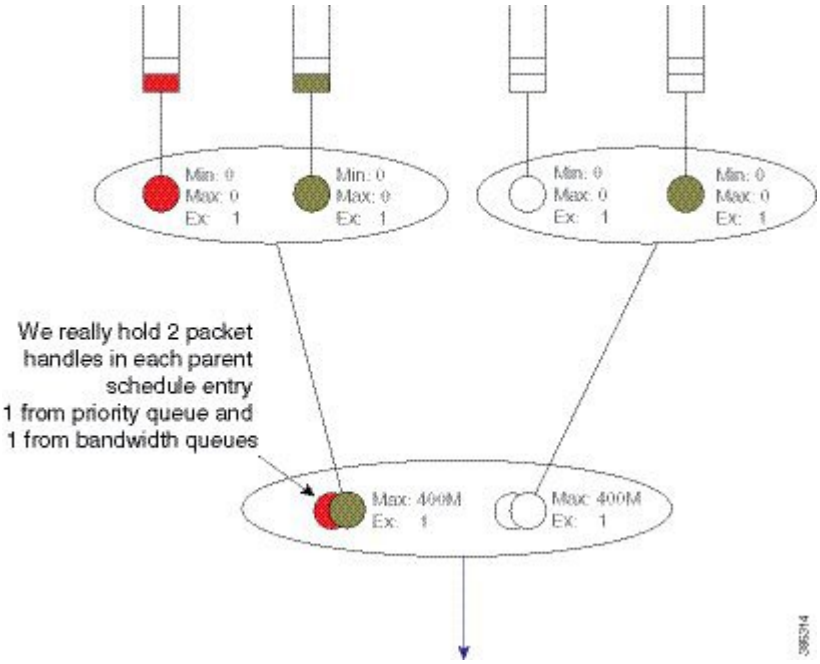


優先度の伝搬の概念

これまでの説明では、帯域幅キューのみを扱っており、やや単純でありました。実際には、各子では、親スケジュールがプライオリティ（キュー）パケットの処理 および帯域幅（キュー）パケットの処理を保持できます（この機能をレーンの通過と呼ぶ）。パケットの処理が子スケジュールから親に送信されると、それがプライオリティクラスまたは帯域幅クラスのどちらから発生したのかを示し、さらに優先度レベルも示します（この動作を優先度の伝播と呼ぶ）。

優先度の伝播については、この章の後半で説明します。ここでは、階層的なスケジューリングのルールを理解する上で概念のみを紹介します。

図 *5:* 親スケジュールはプライオリティおよび帯域幅の処理（レーンの通過）を保持できます。



この階層では、プライオリティサービスは親スケジュールエントリの設定を必要としません。（親）スケジュールエントリには、最大ウェイトと超過ウェイトの2つのパラメータしかありません。

階層型スケジューリングの操作

スケジューリングの章では、物理インタフェースに適用されたフラットポリシーのスケジューリング決定がどのように行われるかについて説明します。ここでは、リーフスケジュール（子としてキューのみを持つスケジュールに対処するシナリオ）のスケジューリングルールについて説明します。それらの規則はまだ有効です。

親/子のインタラクションのルールを含めるために、ここでその説明をさらに掘り下げていきます。

- プライオリティトラフィックは、親スケジュールで設定されたMax（**shape** コマンド）に対してカウントされます。
- プライオリティトラフィックは、親で設定されたEx（**bandwidth remaining ratio** コマンド）によって変更されることはありません。
- 親スケジュールのプライオリティパケットは、常に帯域幅パケットの前にスケジュールされます。
- 優先度は、親で設定されたシェープレートに比例してスケジュールされます。この点はトピックを全体的にカバーするために説明していますが、優先の対象となる負荷がインターフェイスをオーバーサブスクライブできない限り、あまり重要な要素ではありません。

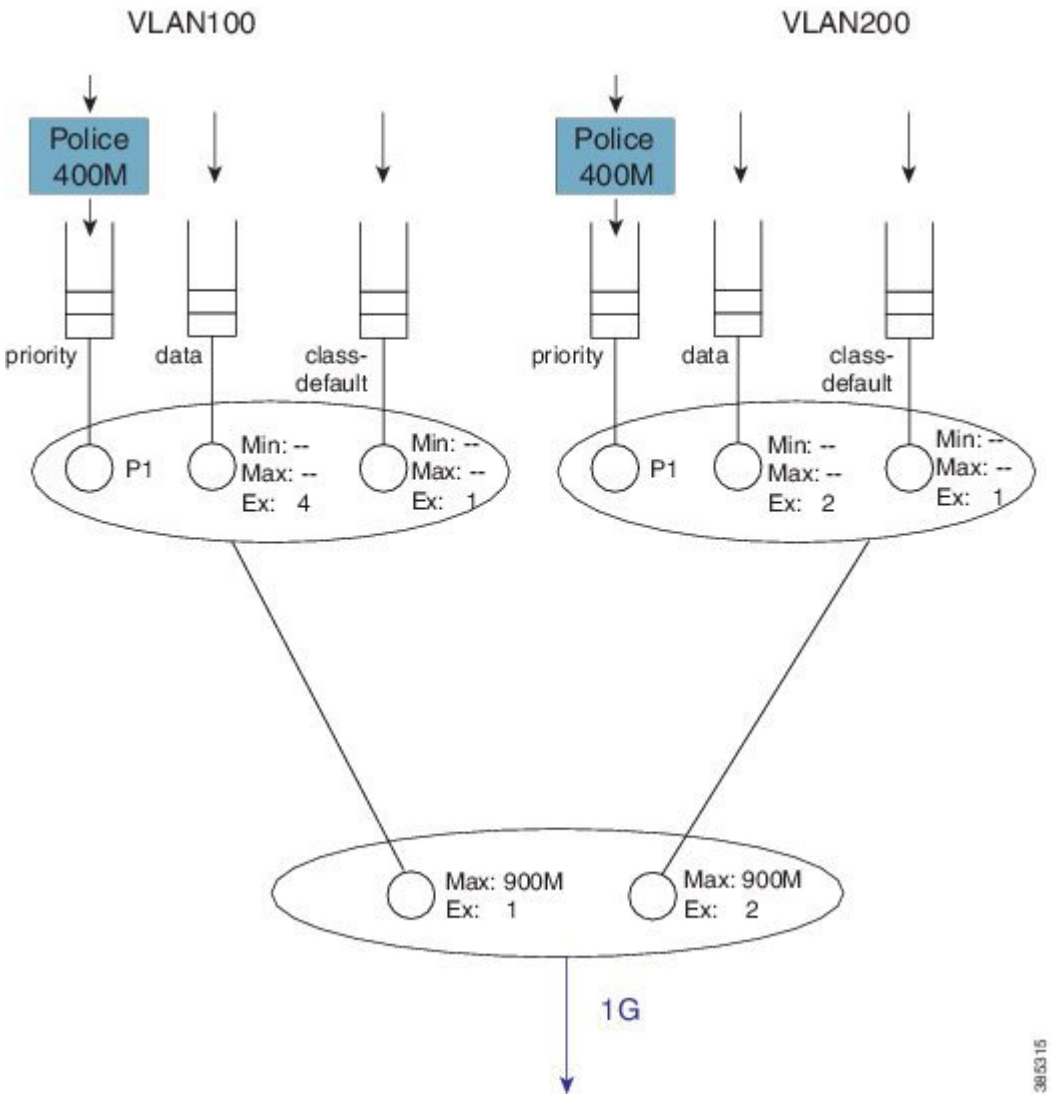
- 優先度の伝播では、親は、パケットがプライオリティ キューから来たことを認識しますが、それが P1（プライオリティ レベル 1）または P2（プライオリティ レベル 2）であるかどうかは認識できません。
- **bandwidth** または **bandwidth remaining** コマンドで設定されたキューからのトラフィックは、親側でも同等に処理されます（最小帯域幅の伝播なし）。これ以降の章では、帯域幅キューからのトラフィックを帯域幅トラフィックと呼びます。.
- 親側での超過ウェイト設定は、プライオリティ トラフィックによって消費されない物理帯域幅をめぐって競合する複数の子からの帯域幅トラフィック間の均等性を制御します。

これらのルールを理解するには、次の設定例を見てみましょう。後で、設定がデータ パス設定にどのようにマッピングされるかを詳しく説明します。現時点では、図と図に示されているスケジュール エントリから動作を充分に理解できます。

```
policy-map child100
  class priority
    priority
    police cir 400m
  class data
    bandwidth remaining ratio 4
!
policy-map parent100
  class class-default
    shape average 900m
    service-policy child100
!
policy-map child200
  class priority
    priority
    police cir 400m
  class data
    bandwidth remaining ratio 2
!
policy-map parent200
  class class-default
    shape average 900m
    bandwidth remaining ratio 2
    service-policy child200
!
int g1/0/4.100
  encaps dot1q 100
  service-policy out parent100
!
int g1/0/4.200
  encaps dot1q 200
  service-policy out parent200
```

次の図は、以前の設定に関連付けられているスケジューリング階層を示しています。前に説明したように、親ポリシーの **shape** コマンドは Max パラメータを設定し、**bandwidth remaining ratio** コマンドはスケジュール エントリに Ex パラメータを設定します（ルール 1 および 2）。明示的に設定されていない場合、後者はデフォルトの 1 になります：

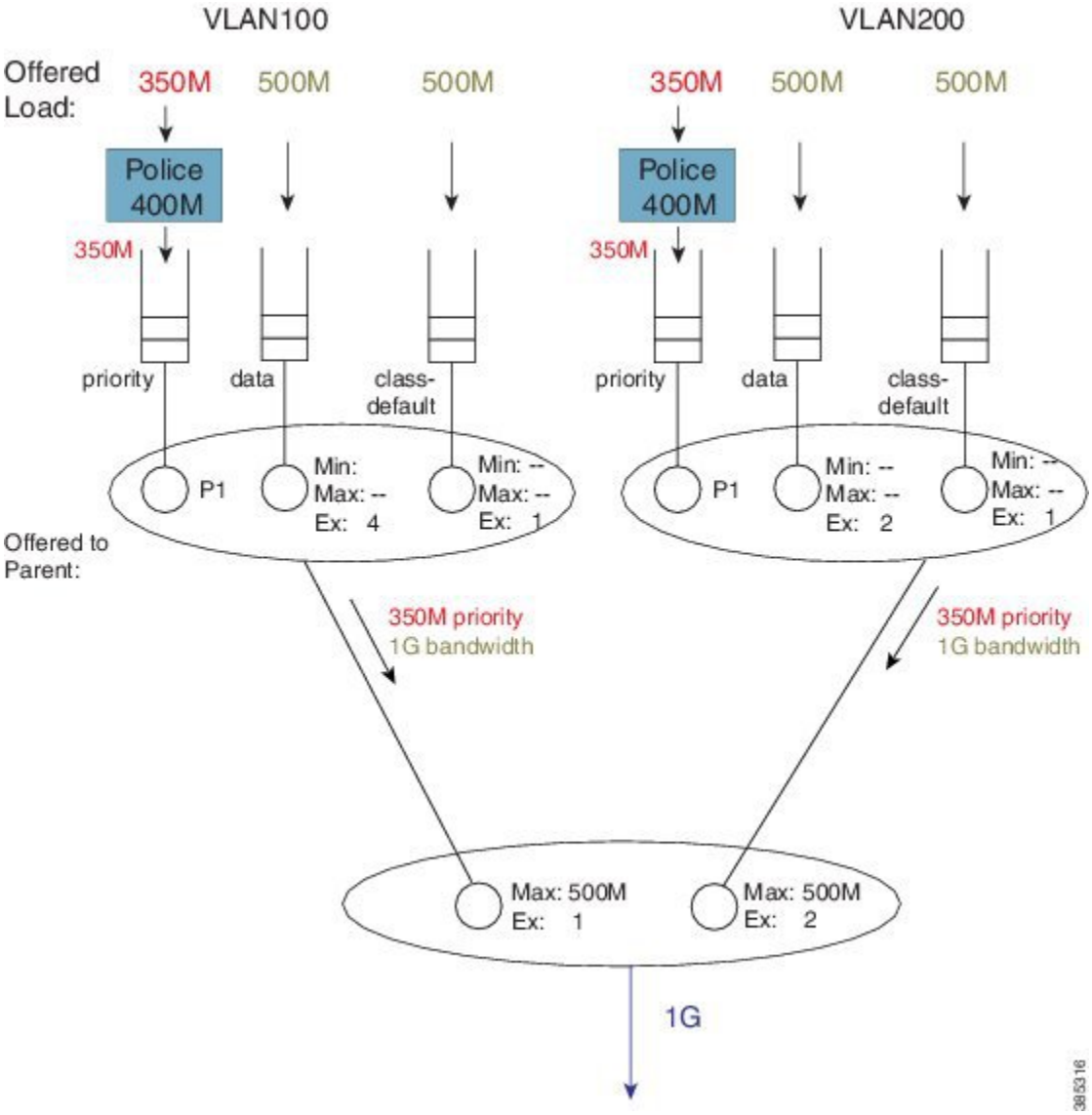
図 *6:* スケジュール階層の例：プライオリティ負荷全体を転送



それでは、供給負荷に対して予想されるスループットを見てみましょう。

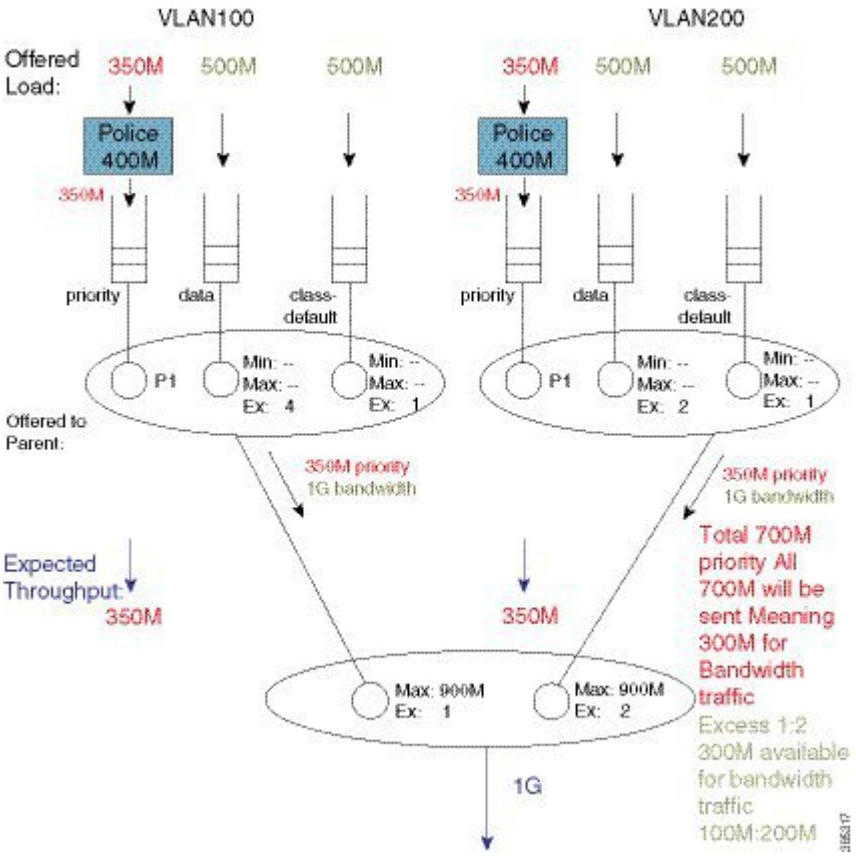
（注） 次の例では、オーバーヘッドアカウンティングを無視しています。これらは、非主要な詳細点とは無関係に、予想されるスループットを計算する方法を説明することのみを目的としています。

図 7: 各子スケジュールから親に提供されるものを計算する



予想されるスループットを計算する際の最初のステップは、クラスごとに供給される負荷を目測することです。次のステップでは、それらを集約して、親に供給されるプライオリティクラスと帯域幅クラスからの合計負荷を観察します。

図 *8*：帯域幅キューの残存帯域幅の計算



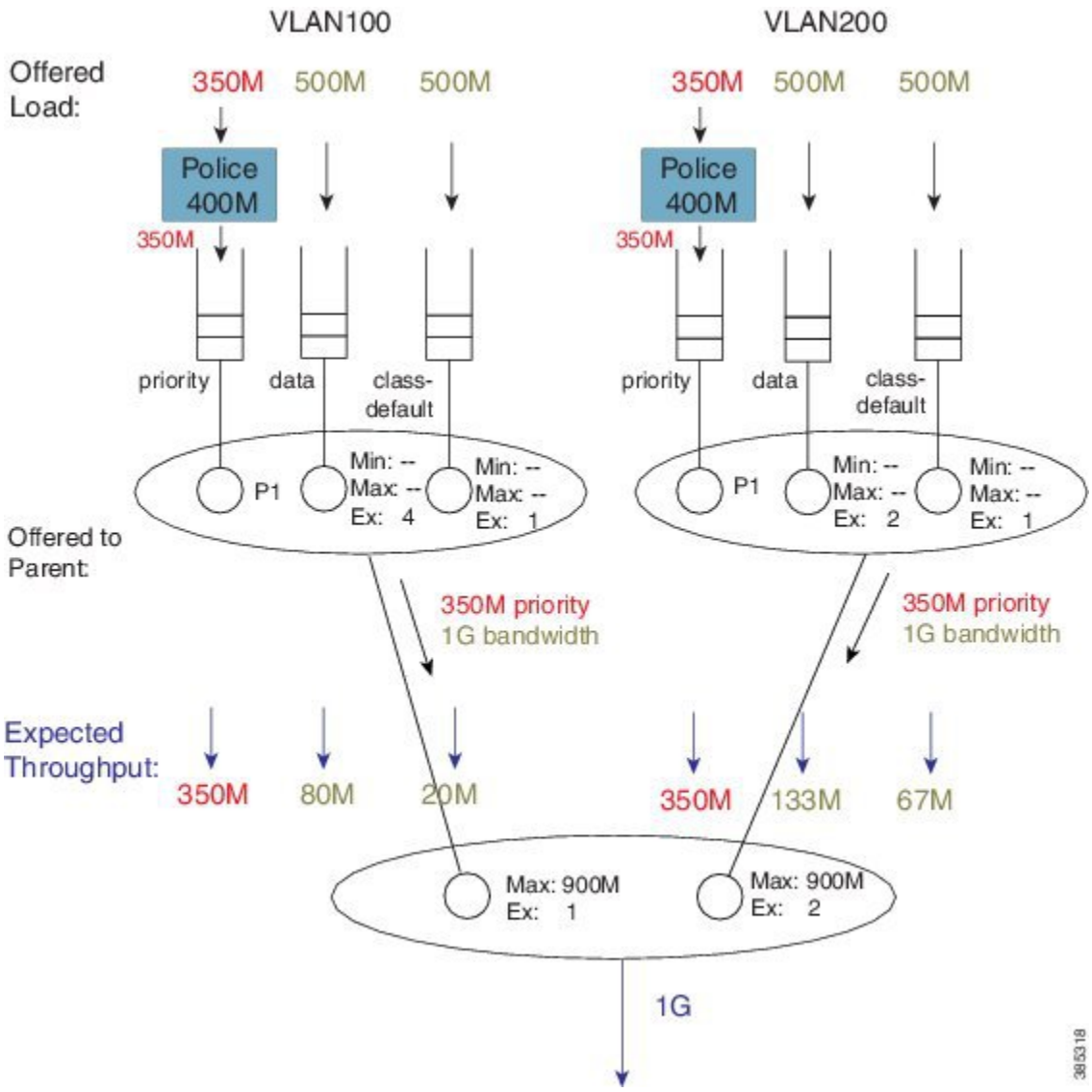
各子スケジュールは、350 Mbps のプライオリティ トラフィックを親に提供しています。インターフェイスには 1 Gbps の利用可能な帯域幅があるため、700 Mbps のプライオリティ負荷全体が転送されます。

ルール 3 に従って、帯域幅トラフィックの前にプライオリティ トラフィックをスケジュールします。各親スケジュール エントリの最大値（レート）が、そのエントリの子スケジュールから供給されるプライオリティ負荷を超えているため、350 Mbps のトラフィック全体が転送されます。

スケジュールされたプライオリティ負荷（350 mbps + 350 mbps のトラフィック）を使用すると、帯域幅（キュー）トラフィックの（残存）帯域幅を計算できるようになります（プライオリティ負荷によって消費された 700 Mbps、総帯域幅 300 Mbps または 1 Gbps）。

親スケジュールは Ex 設定を使用して 300 Mbps の（残存）帯域幅を割り当てます。 VLAN 100 と VLAN 200 の Ex 値が 1 と 2 の場合、それぞれ帯域幅は 1：2 で共有されます。VLAN 100 は 100 Mbps を受信し、VLAN 200 は 200 Mbps の帯域幅トラフィック スループットを受信します。

図 *9*: 子スケジュールの超過ウェイトに基づく帯域幅の共有



この 100 Mbps がどのように割り当てられるかを計算するために、VLAN 100 の帯域幅キューのスケジュール エントリ（スケジュール内）を調べることができるようになりました。

最低帯域保証が設定されていないため（**bandwidth** コマンドは親スケジュールではサポートされていない）、共有はすべて子スケジュールのスケジュール済み Ex 値に依存します。設定（クラス データに 4、class-default に 1）に基づいて、100 Mbps が 4：1 で共有されます（クラス データは 80 Mbps を受信し、class-default は 20 Mbps を受信します）。

VLAN200 で同じアプローチに従うと、利用可能な 200 Mbps は 2：1 に分割されます。クラス データは 133 Mbps を受信し、class-default は 67 Mbps を受信します。

おそらくすべてのクラスがオーバーサブスクライブされていることに気づいたでしょう。これは、計算した予想スループットが各クラスの最低保証サービス レートでもあったことを意味します。階層型スケジューリングでは、親スケジュールでの帯域幅共有により、子スケジュール

に送信待ちのパケットがない場合でも帯域幅を浪費しないようにします。フラットポリシーでの帯域幅の共有と同様に、子が使用していない帯域幅は他の子が利用できます。

優先度の伝播

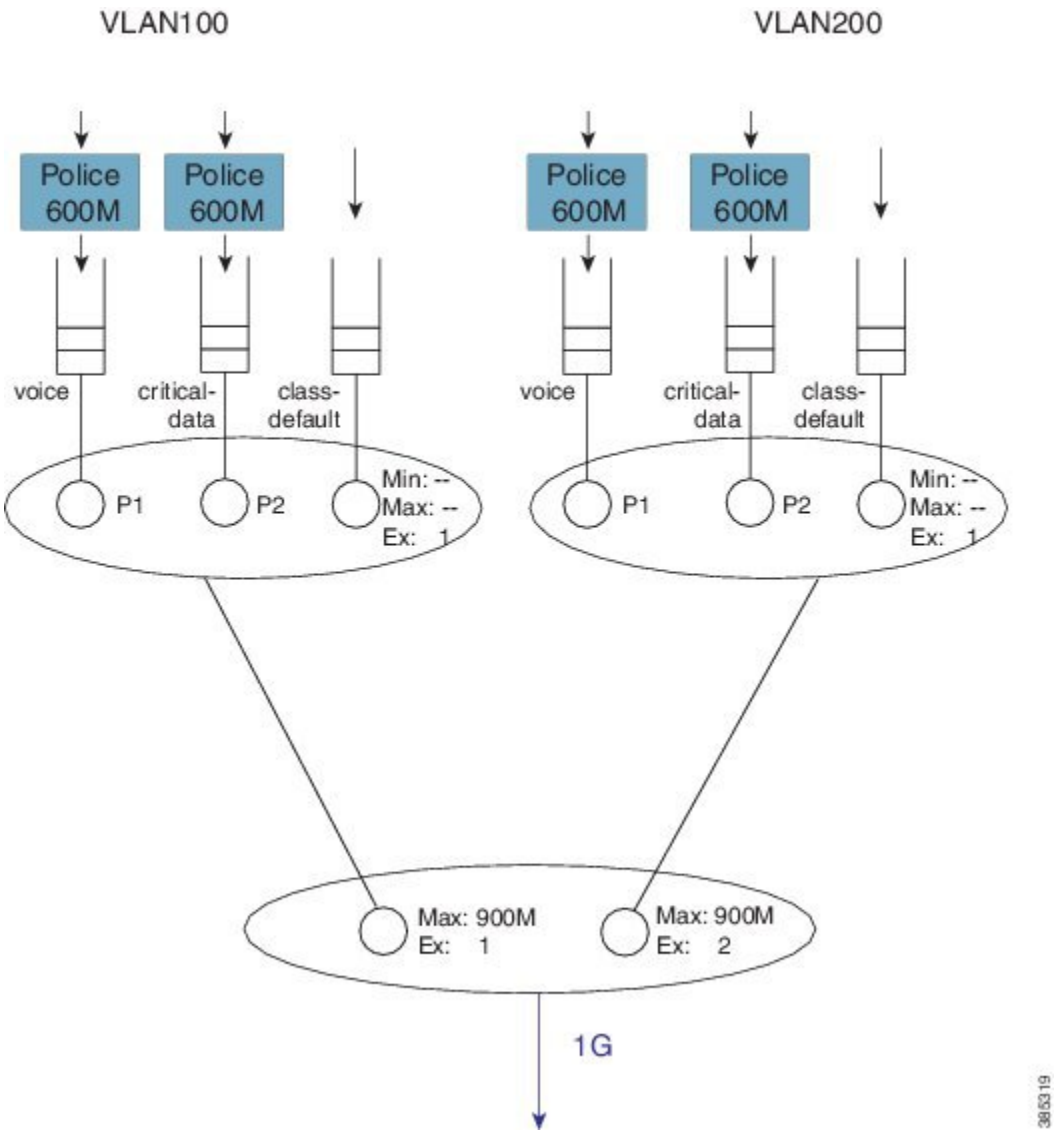
「優先度の伝搬の概念 （6 ページ）」に関しては、ここでいくつかの点を強調するために次の設定例を使用します。

```
policy-map child
  class voice
    priority
    police cir 600m
  class video
    priority
    police cir 600m
!
policy-map parent100
  class class-default
    shape average 900m
    service-policy child
!
policy-map parent200
  class class-default
    shape average 900m
    bandwidth remaining ratio 2
    service-policy child
!
int g1/0/4.100
  encaps dot1q 100
  service-policy out parent100
!
int g1/0/4.200
  encaps dot1q 200
  service-policy out parent200
```

（注） 両方の親ポリシーマップで同じ子ポリシーを使用しています。一意のポリシーマップは、どのレベルでも不要です。 要件が一致する場合は、子ポリシーマップ、さらには親ポリシーマップを共有できます。

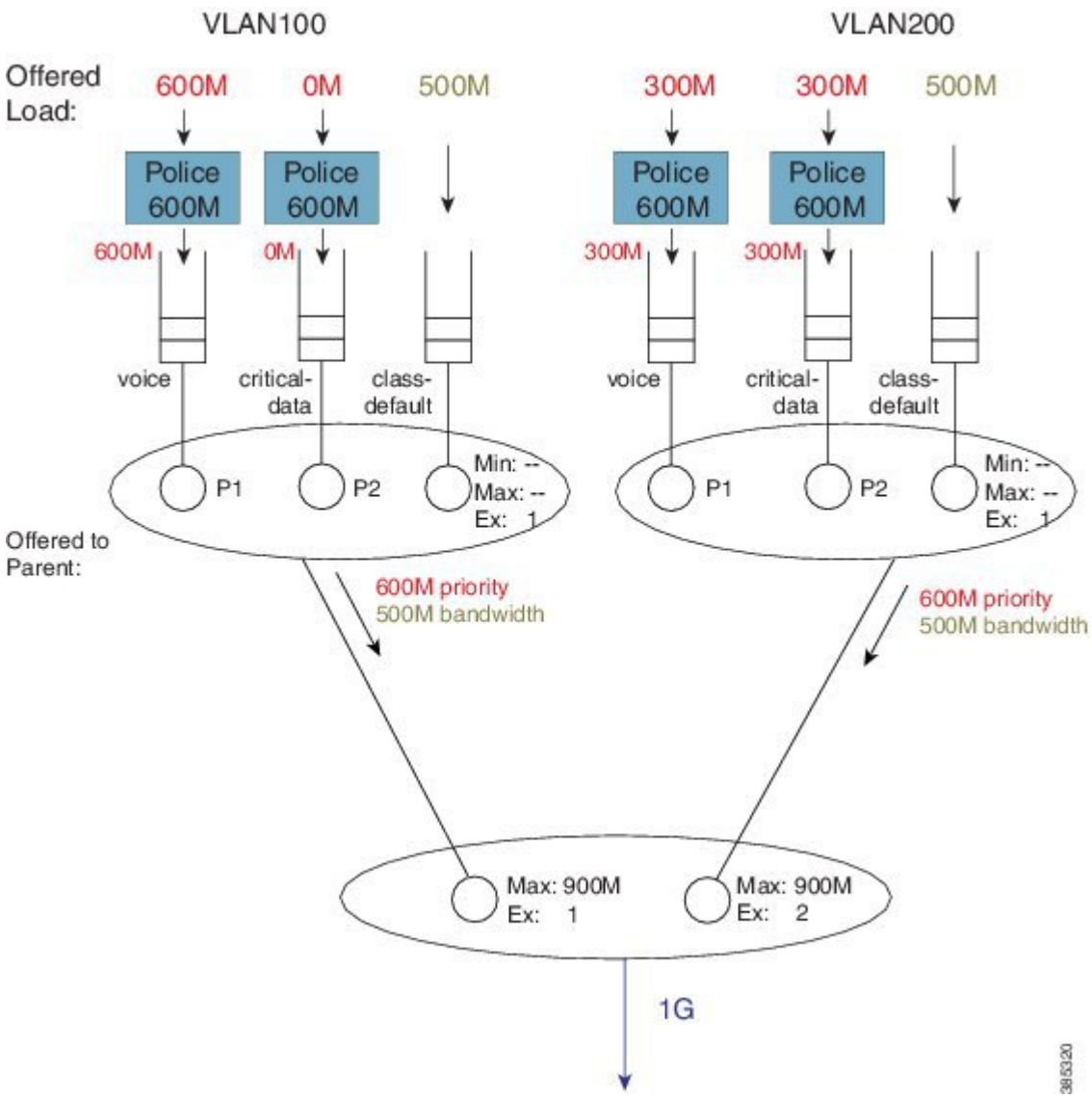
この設定用に作成された階層は次のようになります。

図 *10*：スケジューリング階層の例：子スケジュールにおけるマルチレベル プライオリティ キューイング



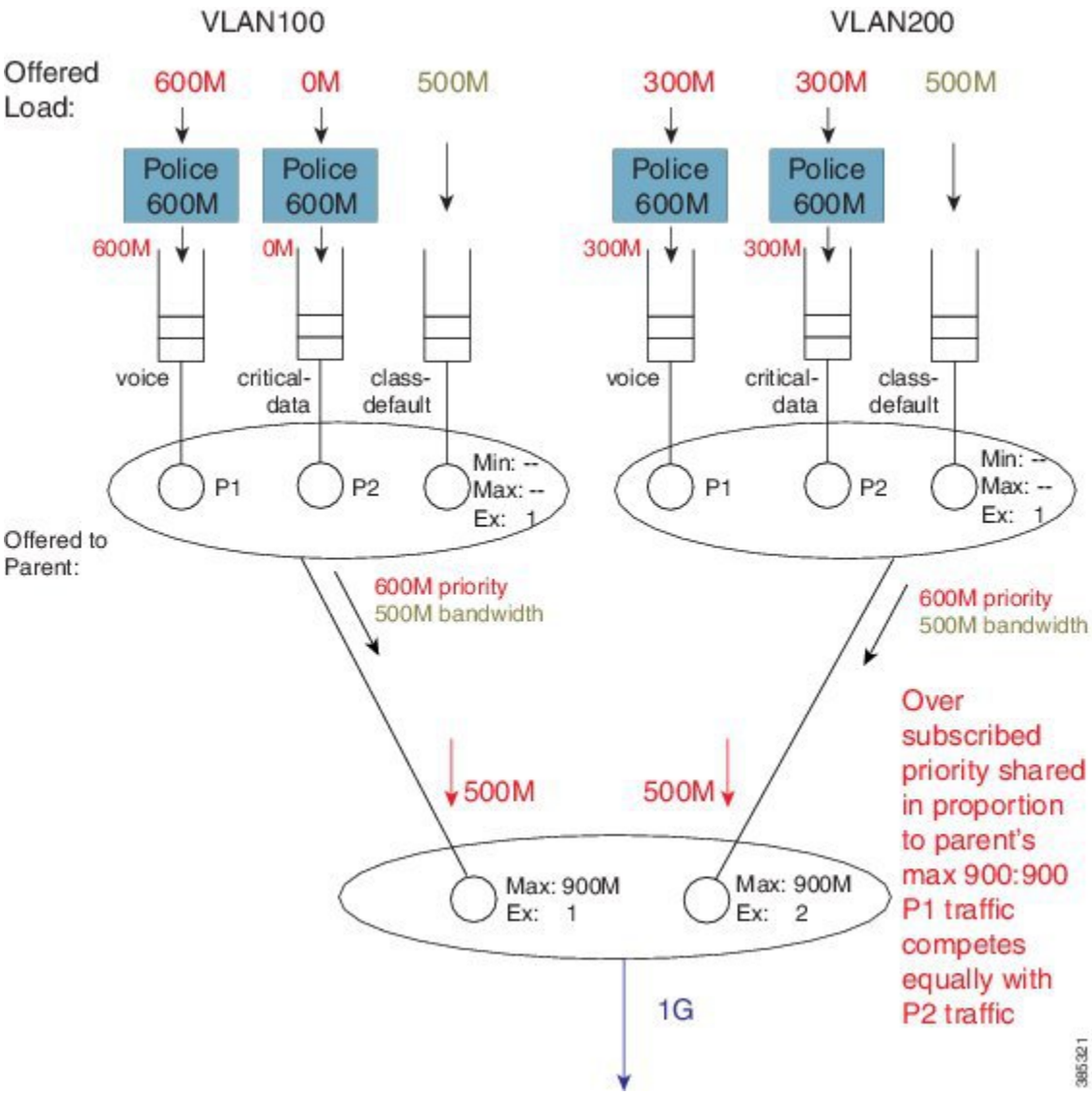
シナリオは、「優先度の伝搬の概念 （6 ページ）」によって異なります。子スケジュールでマルチレベルプライオリティ キューイングがサポートされるようになりました（例、P1 [プライオリティ レベル 1] および P2 [プライオリティ レベル 2] クラス）。 次の図は、各クラスに供給される負荷を示しています。

図 *11* : マルチレベル プライオリティ キューイング：各クラスに供給される負荷



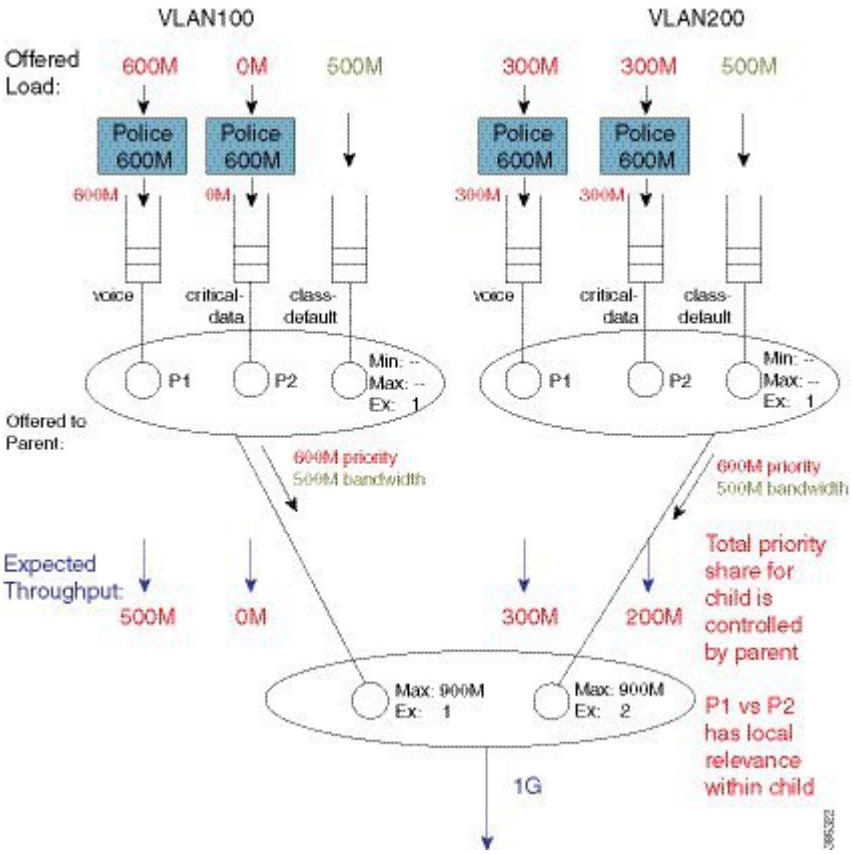
それでは、（各子の）プライオリティキューと帯域幅キューから親に供給される総負荷を見てみましょう。

図 *12*：親の最大比率に対して共有されるオーバーサブスクライブされたプライオリティ キュー



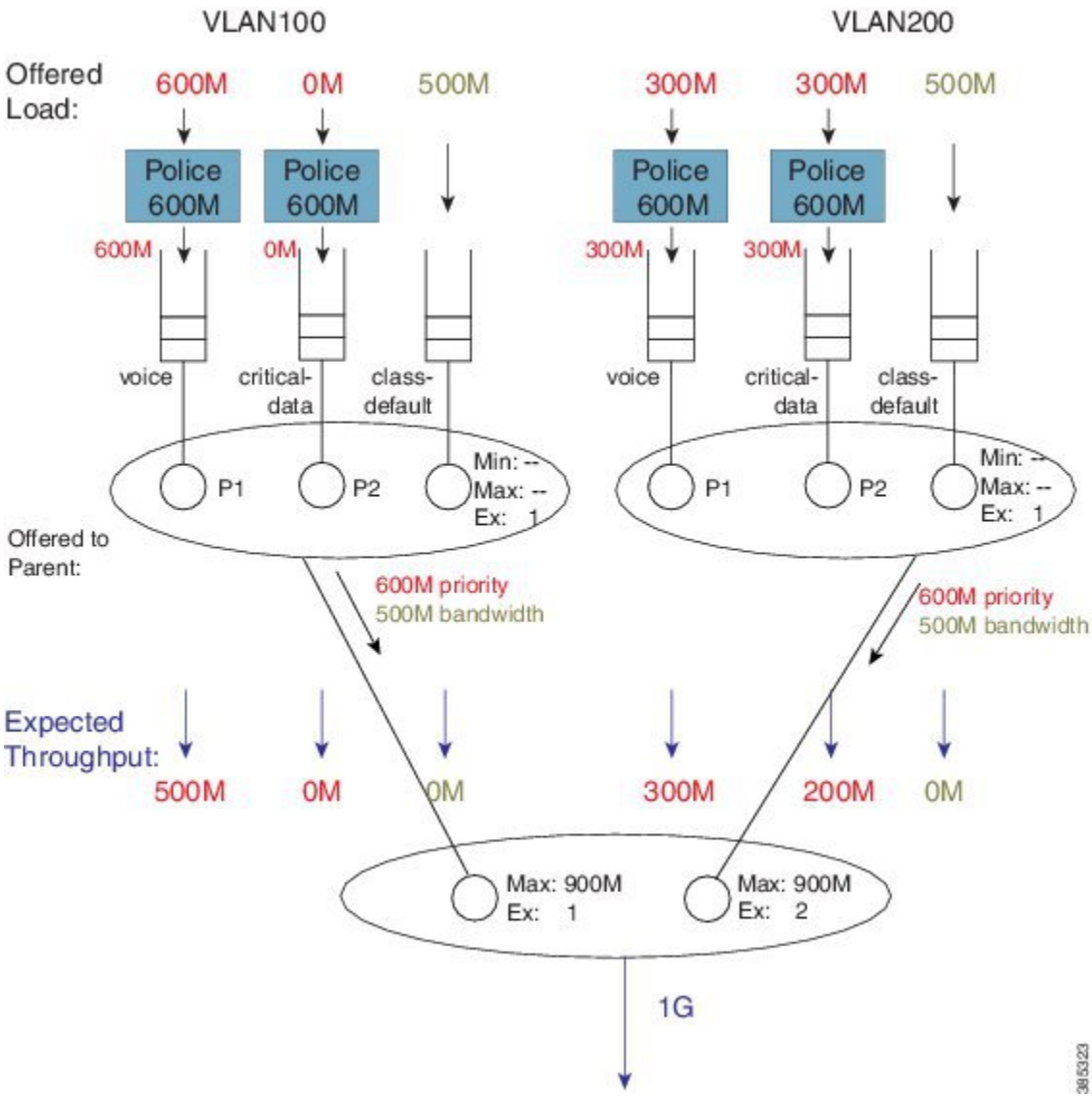
ルール 4 では、 階層型スケジューリングの操作 （7 ページ）親がスケジュール エントリで設定されたシェープ レートに比例して提供されるプライオリティ負荷をスケジュールすることを規定しています。ここでは、各子は 900 Mbps の最大レート（親ポリシーにおける「シェーピング」）を有し、600M のプライオリティ トラフィック（つまり、1 Gbps しか利用できない場合は（1.2 Gbps [600M + 300M + 300M トラフィック]）を提供します。親スケジュールは各子に 500 Mbps を割り当てます。ここで注意する必要がある重要な点は、VLAN 100 からの P1 が VLAN 200 からの P2 トラフィックと同等に競合することです。 （ルール 5 で説明があったように、優先度の伝搬により、パケットがプライオリティ キューから発生したにもかかわらず、優先順位を示していないことが親に警告されます。）

図 *13*：親が子の優先度共有の合計を制御する



親スケジュールは、VLAN200 から 500 Mbps のプライオリティ負荷を受け入れます。 子スケジュールは、その 500 Mbps の帯域幅を割り当てます。子ポリシーは、音声クラスで P1 を設定しました。これは、子スケジュールが常にそのキューから常にパケットを最初に選ぶことを意味します（つまり、スケジュールでは、プライオリティ レベルはローカルで有効となります）。VLAN 200 の音声クラスの予想されるスループットは 300 Mbps です。 critical-data クラスは 200 Mbps（この例では 500 Mbps ー 300 Mbps が未使用の割合）を受信します。

図 *14*：親スケジュールから受け取った帯域幅を子スケジュールが割り当てる



帯域幅キューから予想されるスループットはどうですか。 供給されたプライオリティ負荷が利用可能な物理帯域幅を超えたため、帯域幅キューには何も残りませんでした。 この例は、プライオリティクラスが帯域幅キューを完全に消費する可能性があることを効果的に強調しています。制御パケットがプライオリティキューに入っていないと、ネットワークが不安定になる可能性があります。 事実、プライオリティ キューに制御パケットを配置できなかった場合は、設定ミスが考えられます。

（注） 利用可能な物理帯域幅がすべてのプライオリティ クラス ポリサーの合計を超えていないかを確認してください。そうすれば、後者により他のサービスのキューが消費されることはありません。

優先度の伝播の概念が活用できるのは、スケジューリング階層のみではないことに留意してください。パケットをプライオリティクラスから発生したものとしてマークすると、そのタグは出力インターフェイスに転送されます。出力キャリア カードまたはインターフェイス カードでは、通過するレーンにより、優先パケットができるだけ早くインターフェイスに到達できる場所が複数あります。

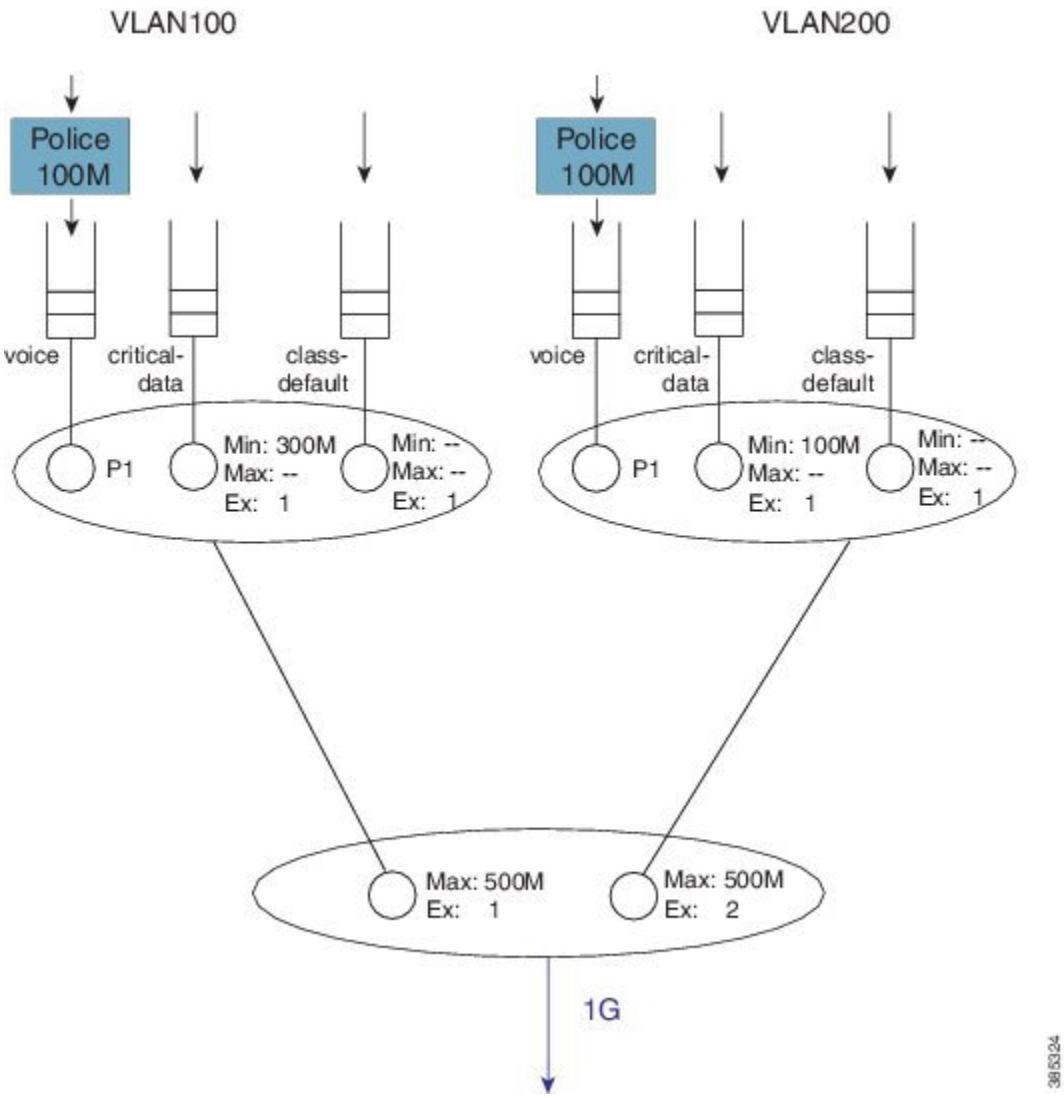
リーフ スケジュールにおける bandwidth コマンド

bandwidth コマンドは親スケジュールではサポートされていませんが（Min 設定はありません）、リーフ スケジュールではサポートされています。次の設定では、子ポリシーマップで **bandwidth** コマンドの動作について説明します。（アスタリスクが付いている行は、この設定が「優先度の伝播 （13 ページ）」に示されているものとどのように比較されるかを示しています。）

```
policy-map child100
  class voice
    priority
    police cir 100m
  class critical-data
    bandwidth 300000                 ****
!
policy-map child200
  class voice
    priority
    police cir 100m
  class critical-data
    bandwidth 100000                ****
!
policy-map parent100
  class class-default
    shape average 500m
    service-policy child100
!
policy-map parent200
  class class-default
    shape average 500m
    bandwidth remaining ratio 2
    service-policy child200
!
int g1/0/4.100
  encaps dot1q 100
  service-policy out parent100
!
int g1/0/4.200
  encaps dot1q 200
  service-policy out parent200
```

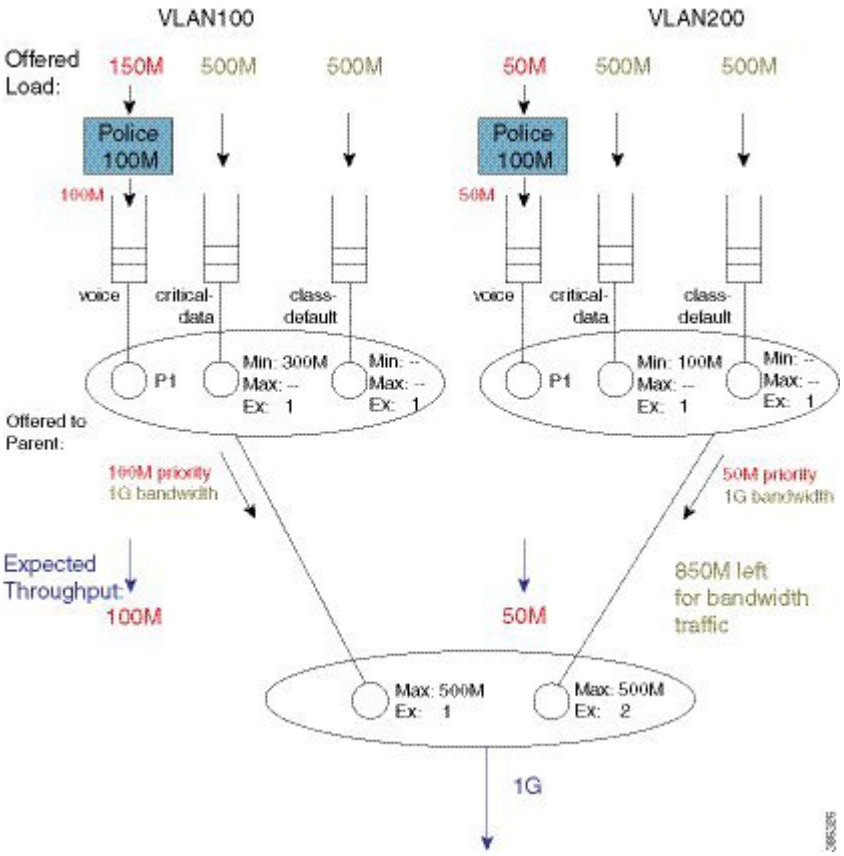
この設定用に作成された階層は次のようになります。

図 *15 :* リーフ スケジュールにおける *bandwidth* コマンドの応用



この階層の動作を説明する上で、（各クラスに）供給される次の負荷を考えてみましょう。

図 *16*：各クラスに供給される負荷

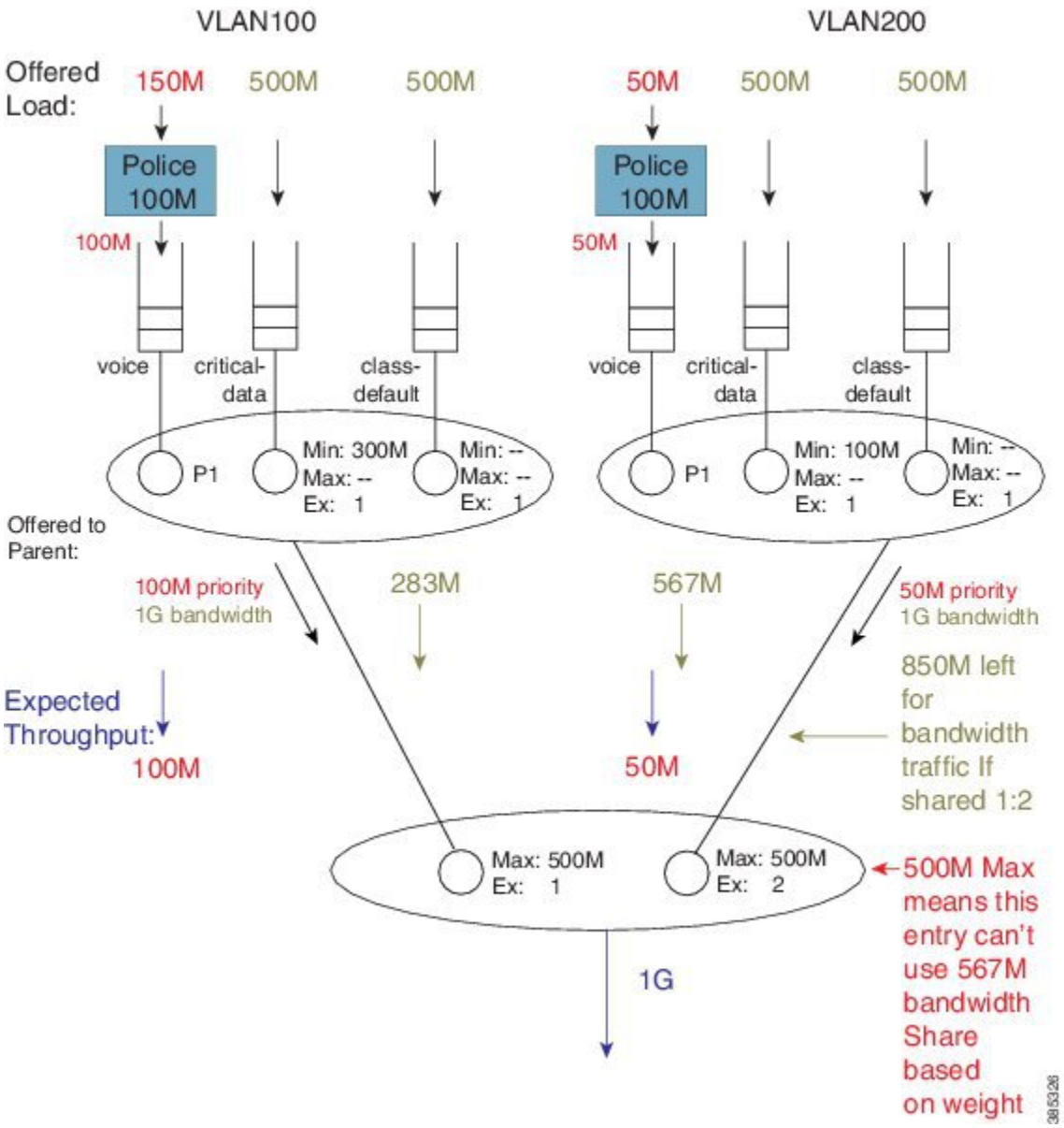


先ほどの例と同様に、（各子の）プライオリティキューと帯域幅キューが親に供給される総負荷について見てみましょう。

この例での総プライオリティ度負荷は150M です。各子は、最大レート（親ポリシーのシェーピング）よりも小さい値を供給しており、総プライオリティ負荷は 1 Gbps の合計使用可能帯域幅を下回っています。（提供されたプライオリティトラフィックの合計が利用可能な合計帯域幅を超えた「スケジュール操作」の例を思い出してください。）これは、各子から供給されるプライオリティ負荷すべてが転送されることを意味します。プライオリティキューからスケジュールされた 150 Mbps があるため、帯域幅キューでは 850 Mbps が使用可能です。

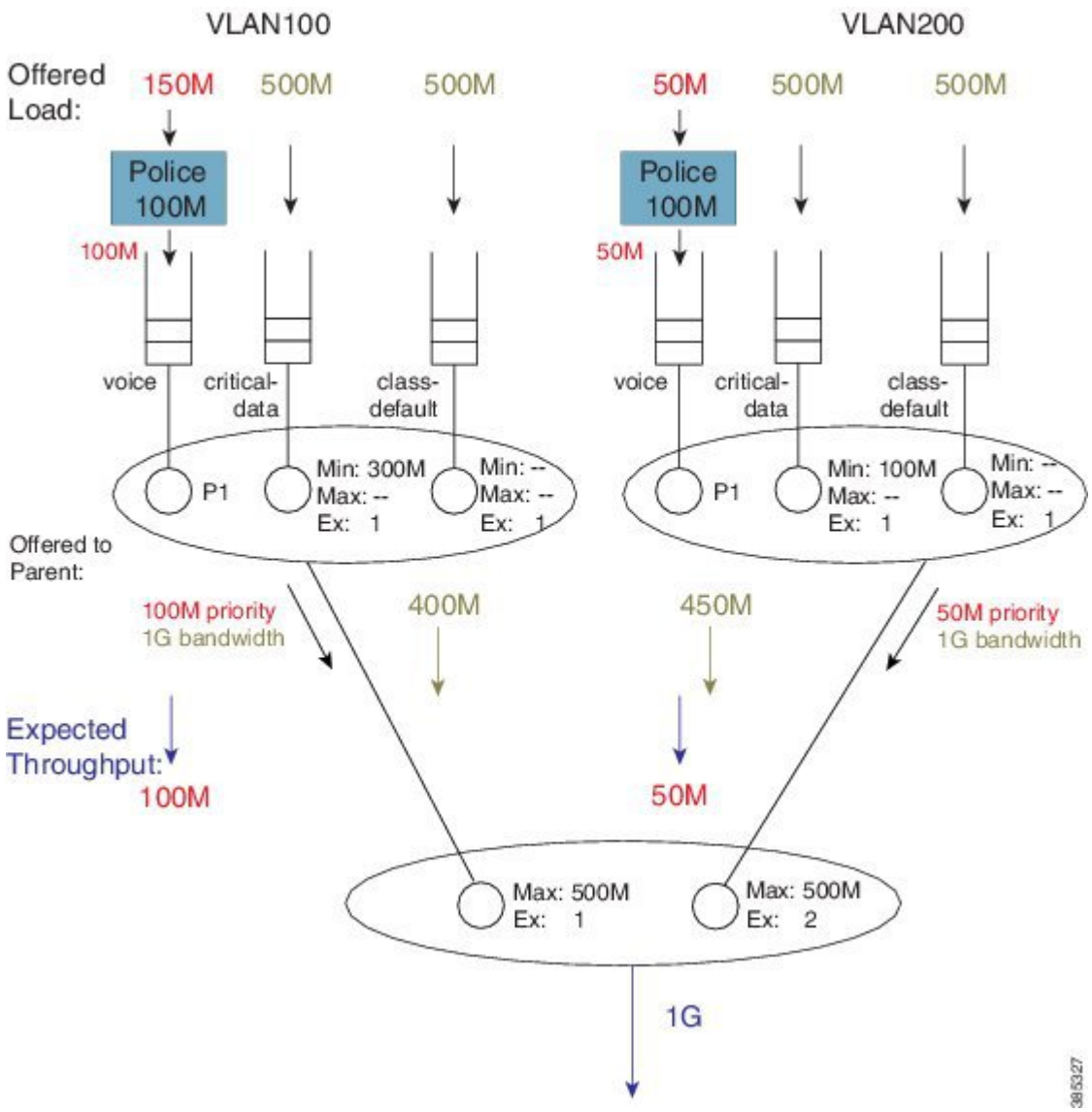
各子間の帯域幅の配分方法を計算するには、まず、親の各スケジュールエントリに設定されている超過ウェイトを見てみましょう。

図 *17*：子間での帯域幅の共有



超過ウェイトに焦点を当てる場合、VLAN200 には 567 Mbps のインターフェイス帯域幅（850 Mbps の 2/3）が割り当てられます。 ただし、スケジュール エントリに設定されている最大値（500 Mbps）も考慮する必要があります。これには、その子からの 50 Mbps のプライオリティ トラフィックも含まれます。つまり、VLAN 200 は実際には 450 Mbps の帯域幅トラフィックを転送し、VLAN 100 は 400 Mbps の帯域幅トラフィックを転送します（VLAN 200 では 850 Mbps ー 450 Mbps）。

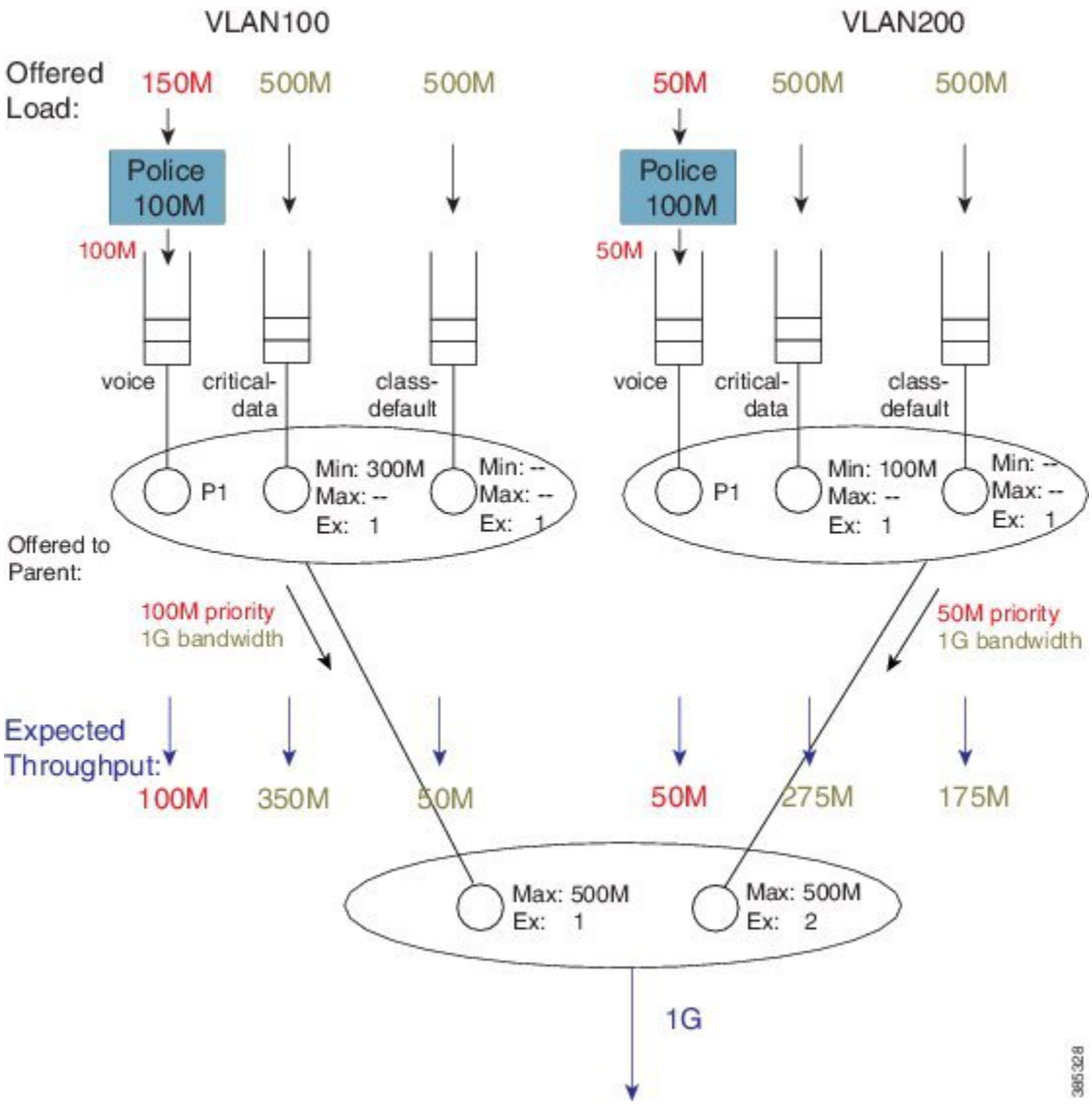
図 *18*：親のスケジュール エントリの最大値が帯域幅の割り当てに与える影響



親レベルの最大値の合計が利用可能な物理帯域幅以下であるため、親ポリシーの Ex 値により値は追加されません。各子はそのシェープ レートと一致する合計スループットを受け取ります（たとえば、VLAN 100 の場合、100M + 400M = 500M [シェープ レート]）。このような設定では、ある子が使用していない帯域幅は他の子が使用できないことに注意してください。すべての子は常に設定された最大値に制限されます。

各子の帯域幅クラスの合計スループットを使用して、その子の各クラスのスループットを計算できます。「スケジュール操作」で説明したように、最低帯域幅保証が常に最初に処理され、超過帯域幅は Ex 値に基づいて共有され、常に 1 がデフォルトになります。

図 *19*：各子スケジュール内で帯域幅を割り当てる上での合計スループットの因数分解



たとえば、VLAN200 の critical-data クラスに割り当てられた帯域幅は、275M（Ex の比率 1：1 から「1/2」が算出され、100M（最低帯域保証）＋1/2（450M-100M））になります。

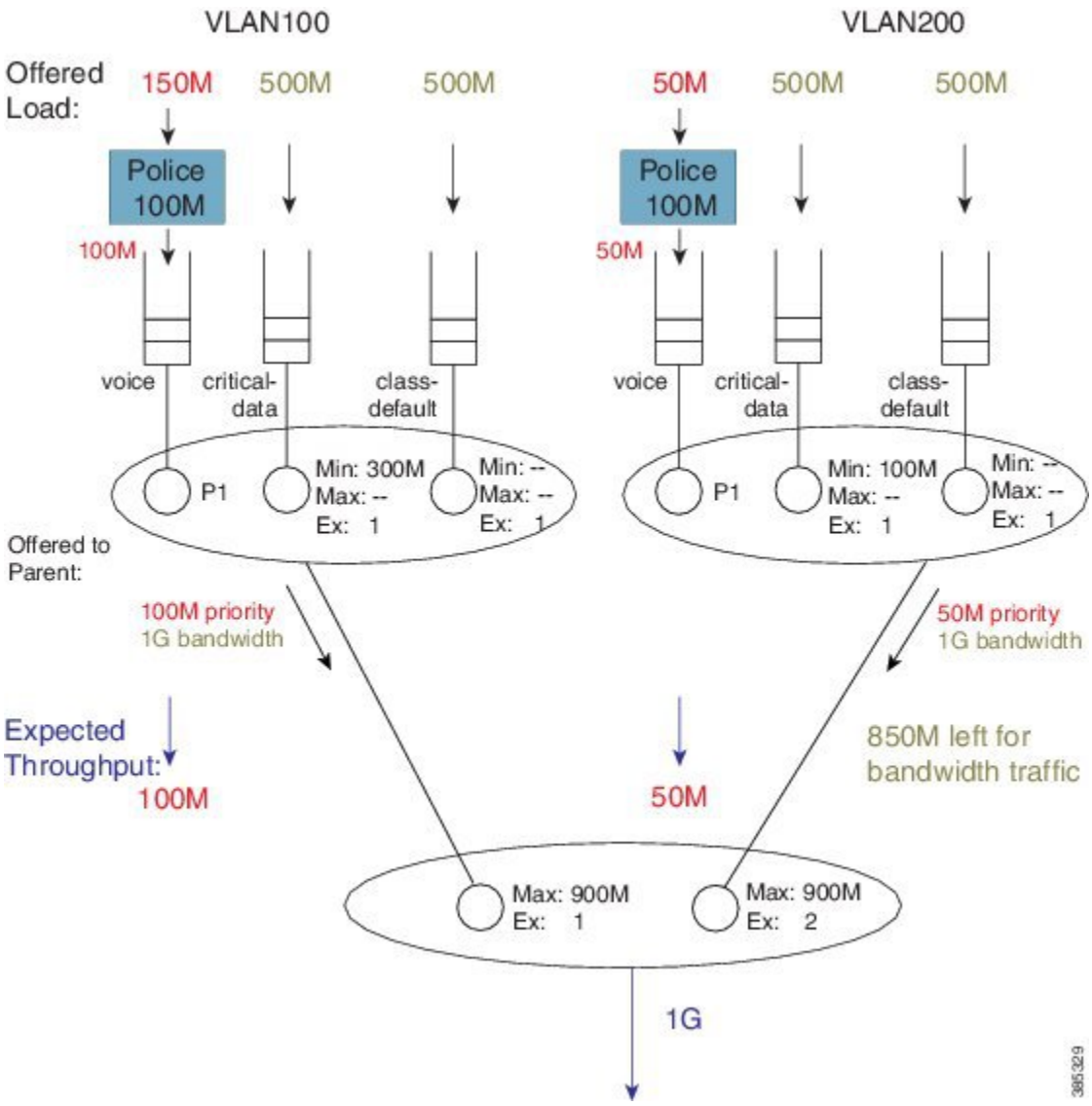
Bandwidth コマンドはローカルでのみ有効

階層型ポリシーで **bandwidth** コマンドを使用するリスクを説明する上で、親シェーパーを増やし、これらが制約要因ではなくなるように、前の設定例を変更します。修正された設定では、親シェーパーの合計数が、利用可能な物理帯域幅をオーバーサブスクライブします。（アスタリスクが付いているコマンドは、この設定が「リーフ スケジュールにおける bandwidth コマンド（19 ページ）」で説明された設定とどのように違っているかを示しています。）

```
policy-map child100
  class voice
    priority
    police cir 100m
  class critical-data
    bandwidth 300000
!
policy-map child200
  class voice
    priority
    police cir 100m
  class critical-data
    bandwidth 100000
!
policy-map parent100
  class class-default
    shape average 900m                        ****
    service-policy child100
!
policy-map parent200
  class class-default
    shape average 900m                        ****
    bandwidth remaining ratio 2
    service-policy child200
!
int g1/0/4.100
  encaps dot1q 100
  service-policy out parent100
!
int g1/0/4.200
  encaps dot1q 200
  service-policy out parent200
```

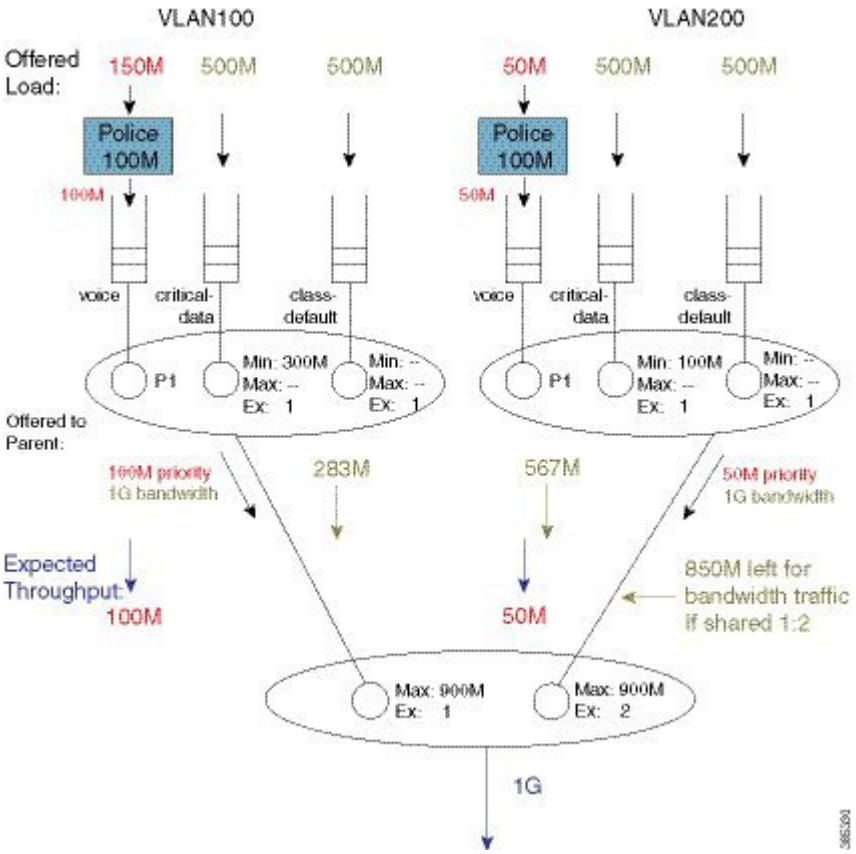
リーフ スケジュールにおける bandwidth コマンド （19 ページ）供給された負荷プロファイルを適用した場合、階層と負荷プロファイルは次のようになります。

図 *20 :* スケジューリング階層の例：親シェーパーは制約を受けなくなった



前の例と同様に、850 Mbps が帯域幅キューに使用できます（残存）。（各子に割り当てられたプライオリティ負荷と帯域幅トラフィックの合計割合を調べると、各子スケジュールの最大値を超えていないことがわかります。）親のスケジュールで設定されている超過ウェイトに基づいて、各子が親スケジュールから受け取る帯域幅の割合を計算します。283 Mbps と 567 Mbps になります。

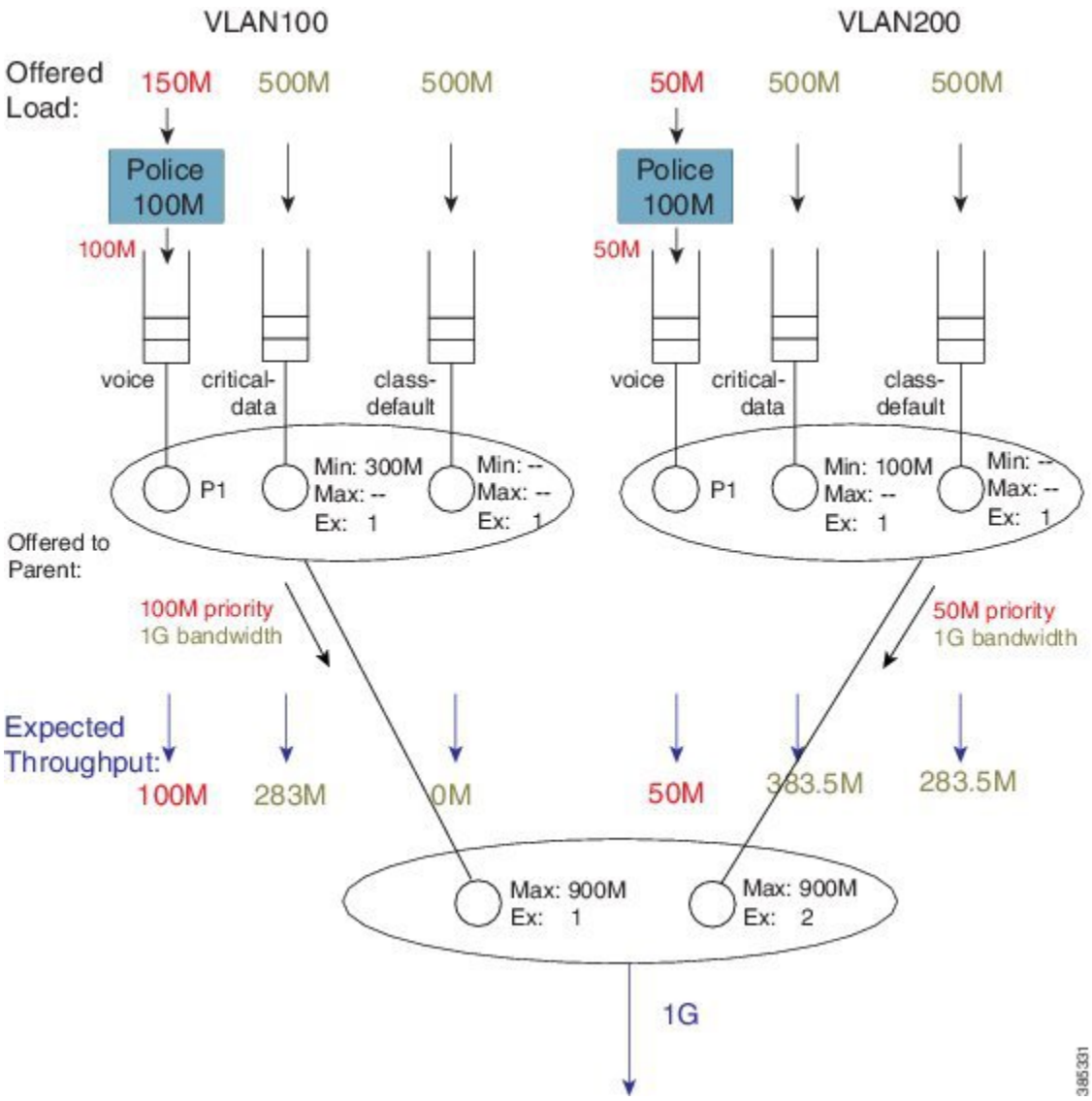
図 *21 :* 親スケジュールで設定された超過ウェイトに基づいて、各子が受け取る帯域幅分を計算する



（注） 前の例とは対照的に、シェープ値はもはや制約されていないため、各子の合計スループットはシェープ レートとは一致しません。

各子スケジュールのエントリを調べて、各クラスに帯域幅がどのように割り当てられるかを確認しましょう。

図 *22*：子エントリにどのように帯域を割り当てるかを指定する方法



VLAN100 の子スケジュールを確認すると、critical-data クラスのスケジュール エントリに 300 Mbps の最小値が設定されていることがわかります。 このスケジュールに割り当てられた 283 Mbps の帯域幅は、この保証を満たすには不十分です。

ここで重要なポイントは、最低帯域幅保証はローカルにのみ関連しているということです。最小帯域幅の伝播はされません。子スケジュールからのトラフィックは、別のスケジュールからの超過トラフィックと同等に競合します。

また、スケジューリング階層で Min を使用すると、他のサービス クラス（この例では VLAN 100 の class-default）のキューが消費される可能性があります。これを回避するには、子ポリシーの **bandwidth remaining** コマンドのみを使用します。

Tip

階層ポリシーで親シェーパーをオーバーサブスクライブし、一部のサービスクラスが消費されないようにするには、プライオリティキューのポリサーの合計が利用可能な帯域幅を超えないようにしてください。さらに、**bandwidth** コマンドの **bandwidth remaining** 使用を検討してください。

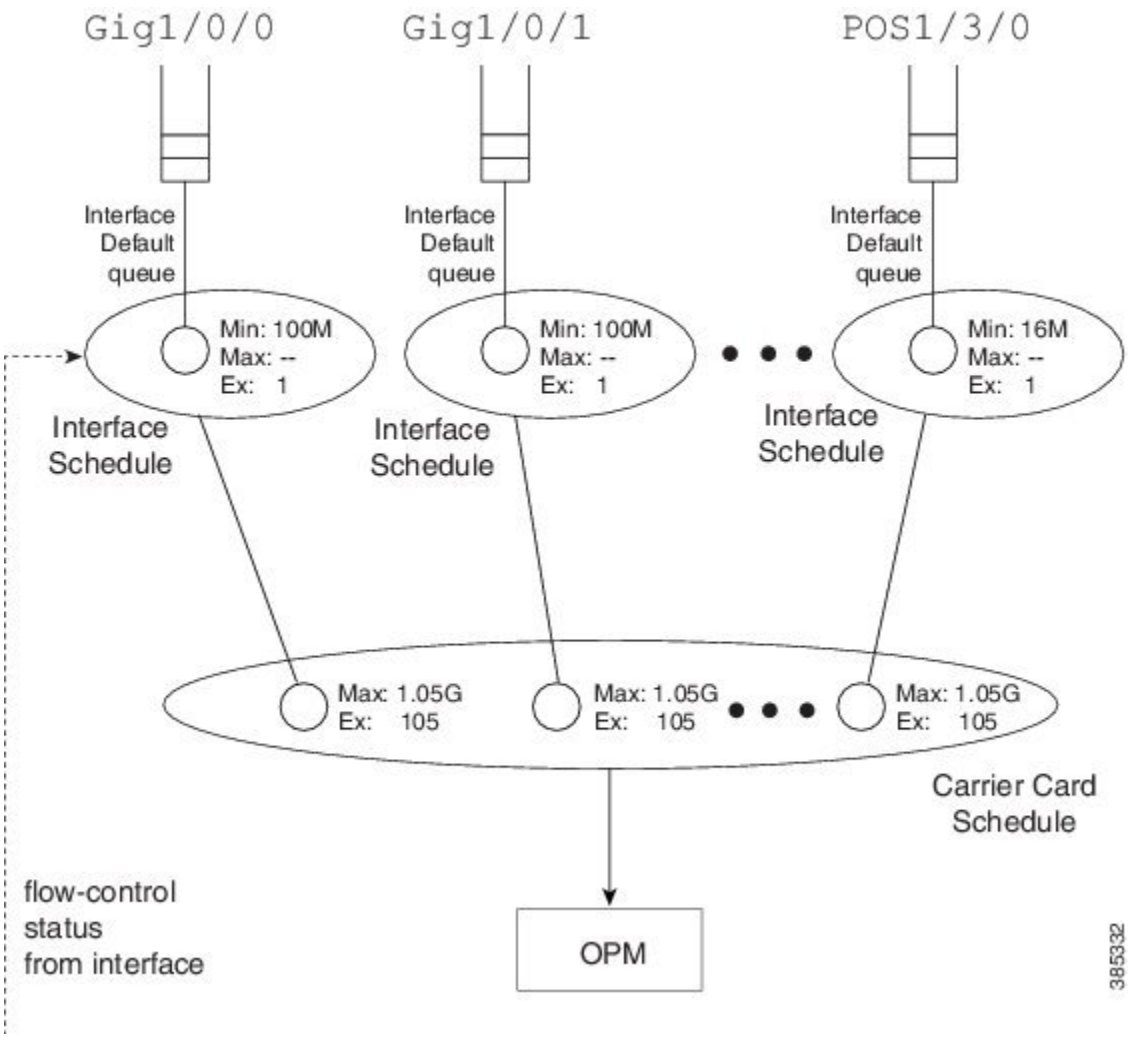
論理インターフェイスに適用されたポリシーマップ

この章の前半では、スケジューリング階層を作成するための 2 つの主な方法、論理インターフェイスに適当された QoS ポリシーと階層型ポリシーマップについて説明しました。 前の例では、論理インターフェイスに適用されるポリシーについて概説しました。 このシナリオをさらに詳しく見ていきましょう。

インターフェイス スケジューリング

論理インターフェイス上のポリシーによって階層がどのように変更されるかを考える前に、QoS ポリシーが適用される前に存在するインターフェイス スケジュールと階層を注意深く調べる必要があります。

図 *23 : QoS* ポリシーの適用前のインターフェイス スケジュールと階層



OPM（出力パケット モジュール）は、スケジューリング階層のルートにあります。パケット処理を受信すると、実際のパケットをメモリから取得し、それを物理インターフェイスにプッシュします。

（意思決定の観点から）OPM 層の直下に、キャリア カードのスケジュールが表示されます。. モジュラー プラットフォームでは、スロットごとにそのようなスケジュールが 1 つありますが、固定システムではシステム全体に対して 1 つあります。

ここで、モジュラ型シャーシについて考えてみてください。スロットが 1 つ搭載され、バックプレーンを介して 10 Gbps のリンクを持つ SIP10 を ESP（組み込み式サービス プロセッサ、転送プロセッサとも呼ばれる）に収容しています。. SIP10 には 4 つの SPA（共有ポート アダプタ）が搭載可能で、 それぞれ最大で 10 Gbps 容量のインターフェイスを搭載可能です。SIP 内で SPA を組み合わせ、バックプレーンの容量を超えると、そのリンクが輻輳ポイントになる可能性があります。万が一これが発生した場合、キャリア カードのスケジュールにより、インタフェース間の均等性が保証されます。各インターフェイスの超過ウェイトはインターフェイス速度に比例します。

プラットフォーム内のトラフィックを調整するために、 各インターフェイスのキャリア カードスケジュールの最大値において、インターフェイスの帯域幅をわずかに超えるように設定します。物理的なインターフェイスに向けて十分なトラフィックを送信する必要があるため、そのインターフェイスを消費させないようにします。さらに、出力バッファがいっぱいになったことがインターフェイスに表示されるたびに送信を中止する必要があります。これは、インターフェイスがダウンストリームデバイスから一時停止フレームを受信するときに発生する可能性があり、シリアルインターフェイスはビットまたはバイトスタッフィングなどによりデータを拡張します。

ここが重要な点で、物理的なインターフェイスが常にネットワークに送信できるデータを持てるように、そのインターフェイスに向けてトラフィックをプッシュし、十分なデータがバッファリングされていることがインターフェイスに表示されるたびに一時的に送信を一時停止する必要があります。

インターフェイスは、フロー制御メッセージによるトラフィックの送信を停止するように指示します。. 設計上、スケジュール（スケジュール エントリではない）がこのメッセージに応答します。送信を停止します。このため、ボックス内のすべての物理インターフェイスに対して常にインターフェイスのスケジュールを設定する必要があります。インターフェイスデフォルト キュー（QoS がないときに使用されるキュー）は、このインターフェイスのスケジュールの子です。

各インターフェイスは、プライオリティ トラフィックと帯域幅トラフィック用に個別のバッファとキューを維持しながら、異なる優先度の高いおよび低いフロー制御メッセージを（インターフェイス スケジュールに）送信できます。

帯域幅トラフィック バッファがいっぱいになっているというメッセージをスケジュールが受信すると、そのようなトラフィックは一時停止されますが、優先度の高いトラフィックは転送され続けます。

プライオリティ バッファがいっぱいになっているというメッセージを受信した場合、輻輳が解消されるまでパケットの送信を一時停止します。

この方式は優先度の伝播の概念を物理インタフェースに活用し（これは、パケット処理がプライオリティクラスまたは帯域幅クラスのどちらに派生するのかを意味します）、遅延の影響を受けやすいトラフィックのジッタを業界最先端レベルまで最小に抑えます。

親ポリシーでのシェーピング/子ポリシー上のキューイング

次に、論理インターフェイスに適用されている可能性のある通常のポリシーを見てみましょう（コンストラクトは「親ポリシーでのシェーピングー/子ポリシーでのキューイング」を参照）：

```
policy-map child100
  class voice
    priority
    police cir 100m
  class critical-data
    bandwidth 300000
!
policy-map parent100
  class class-default
    shape average 900m
    service-policy child100
```

```
!
int g1/0/0.100
  encaps dot1q 100
  service-policy out parent100
```

このコンストラクトでは、親ポリシーでシェーパーを設定する必要があります（シェーパーの平均 900M）。このコンストラクトの本来の目的は、各論理インターフェイスに帯域幅を割り当てることでした。シェーピング（Max）レートを、その論理インターフェイスが所有する帯域幅であるとみなし、子ポリシーがその所有共有内の帯域幅を分配できるようにします。

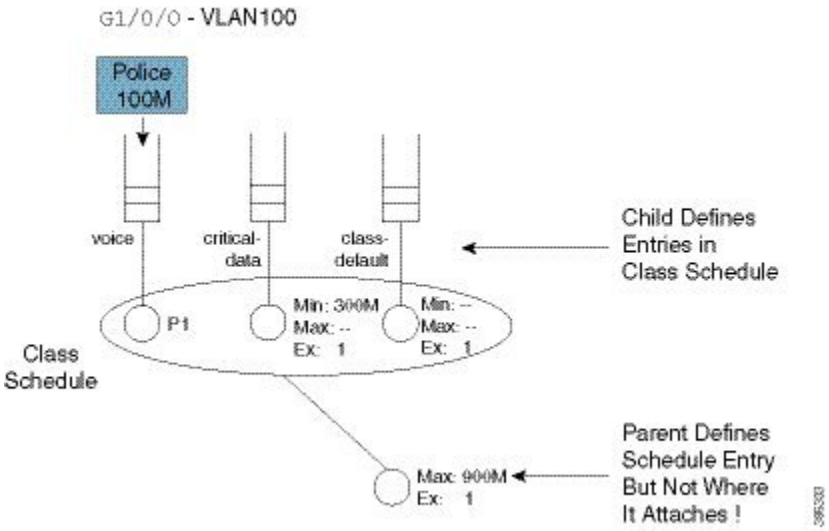
このコンストラクトの便利な応用方法の1つは、リモートサイトのトラフィックを調整することです。たとえば、所属企業のハブに GigabitEthernet リンクがあり、T1 接続でリモート ブランチにトラフィックを送信しているとします。リモートブランチが受信できるレートでトラフィックを送信する必要があります。そのブランチにサービスを提供するプロバイダのデバイス内のパケットがドロップする可能性を回避するには、親シェーパーを T1 レートで設定し、パケットをハブでキューイングします。これにより、そのブランチのリンクが輻輳ポイントであった場合に最初に転送されるものが制御されます。

顧客は、論理インタフェース（個々のサブスクライバーまたはリモートサイトのいずれかを表す）上でシェーパをオーバープロビジョニングするようリクエストしています。すべての論理インタフェースが常にアクティブであるとは限らないということが前提です。個々のサブスクライバのスループットを制限する必要があるため、個々の論理インターフェイスに割り当てられた帯域が消費されていない場合は、その帯域幅を浪費しないようにする必要があります。

それでは、オーバーサブスクライブはどうでしょうか。オーバーサブスクライブする場合、輻輳時に均等性を実現するために、超過ウェイト値を使用して親の帯域幅余剰比率を設定する必要があります。さらに、輻輳時に個々の論理インターフェイスがどのサービスを受けるかに注意してください。

それでは設定の話に戻ります。階層は次のようになります。

図 *24*：「親ポリシーでのシェーピング/子ポリシーでのキューイング」コンストラクト

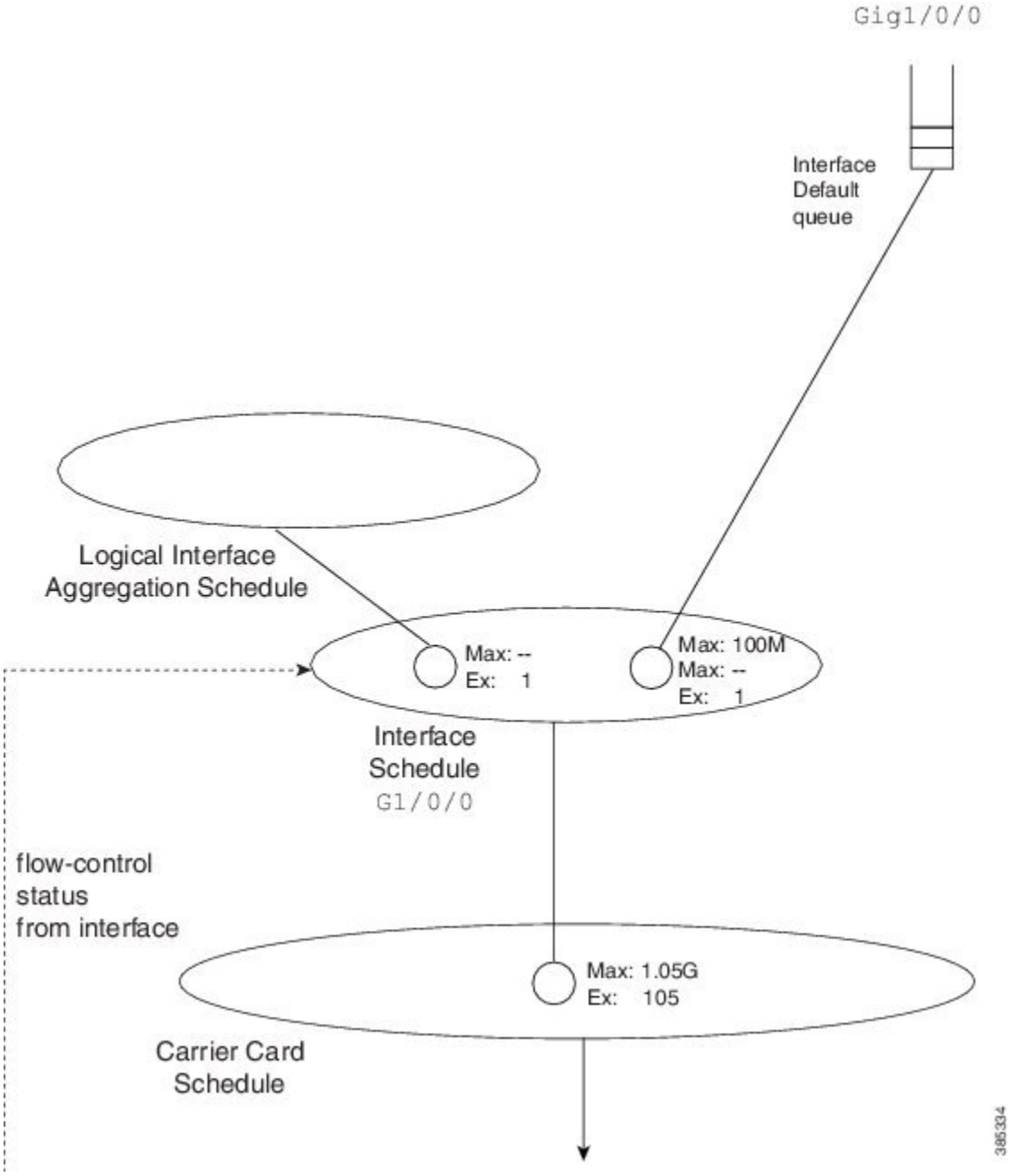


前述のように、子ポリシーは論理インターフェイス内の帯域幅の共有を定義します。通常、ここでは（音声などの）キューをクラス キュー (ポリシーマップ内のクラスによって定義された処理を含む）と呼び、このレイヤでのスケジュールをクラス レイヤ スケジュールと呼びます。.

親ポリシーでは、親シェーパー（最大：900m）と、暗黙の帯域幅の共有である '1'（Ex：1）を定義します。QoS設定では、この論理インターフェイスを既存のインターフェイス階層に接合する場所が明示的に指定されない（アタッチされていないスケジュールエントリに注意）ため、ルータは論理インターフェイスがどの物理インターフェイスに関連付けられているかを認識する必要があります。

VLAN 上のポリシーでは、どのインターフェイスが関連しているかは明らかです。サブインターフェイス設定に（論理インターフェイス）ポリシーを適用します。他のインターフェイスの種類（トンネルインターフェイスなど）については、ルーティング情報を調べてその特定の論理インターフェイスに対して、出力物理インターフェースを指定する必要があります。

図 *25 :* 既存のインタフェース階層（接合前）

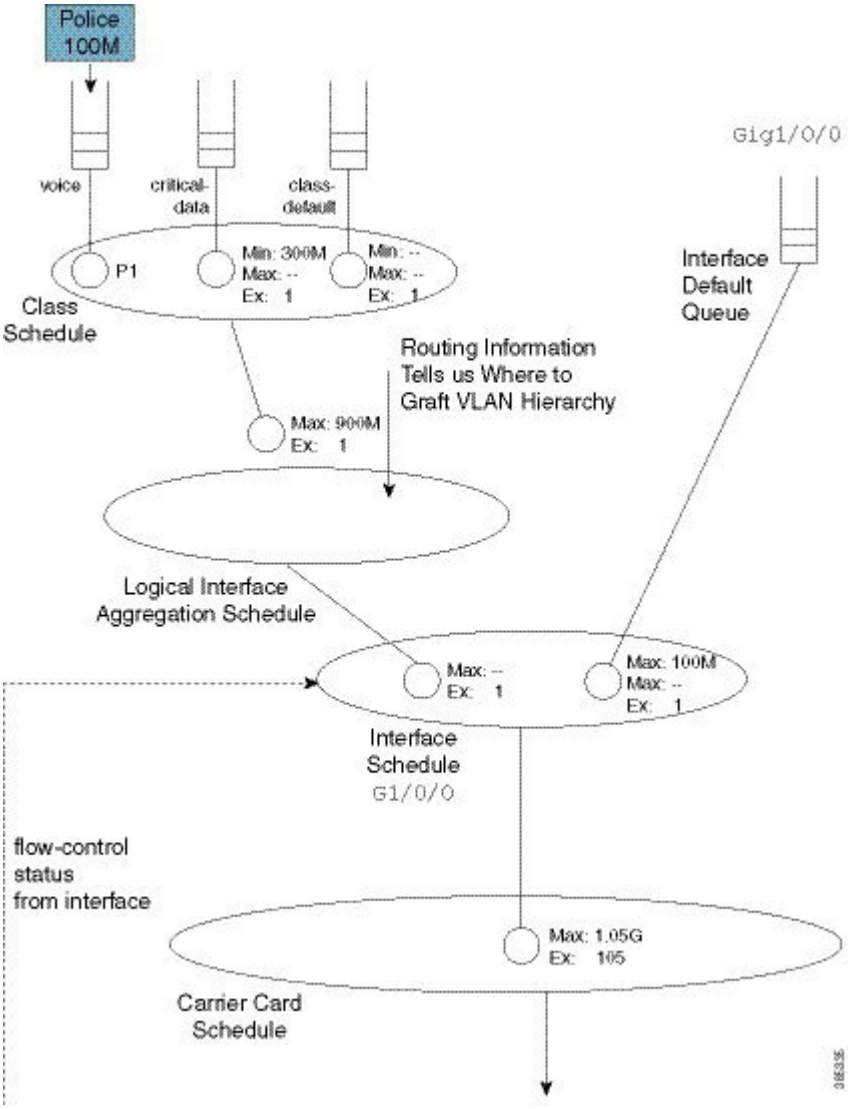


どのインターフェースが関連していることがわかったら、そのインターフェースの階層を変更できます。 最初に、「親ポリシーでのシェーピング（または子ポリシーでのキューイング）」で定義された論理インターフェイス階層の接合場所として機能するスケジュール（論理インターフェイス集約）を作成します。

当初、インターフェース スケジュールには単一の子、インターフェース デフォルト キューしかありませんでした。今日では2番目の子である、論理インターフェイス集約スケジュールを

作成できるようになりました。 このスケジュールの超過ウェイトが、インターフェイスのデフォルトキューのウェイトとどのように一致するかを確認します。通常どおり、デフォルトの「1」になります。

図 *26*：既存のインターフェース階層（接合後）



親ポリシーでのシェーピングでは、子ポリシーを含む class-default しかありません。

```
policy-map parent100
  class class-default
    shape average 900m
    service-policy child100
```

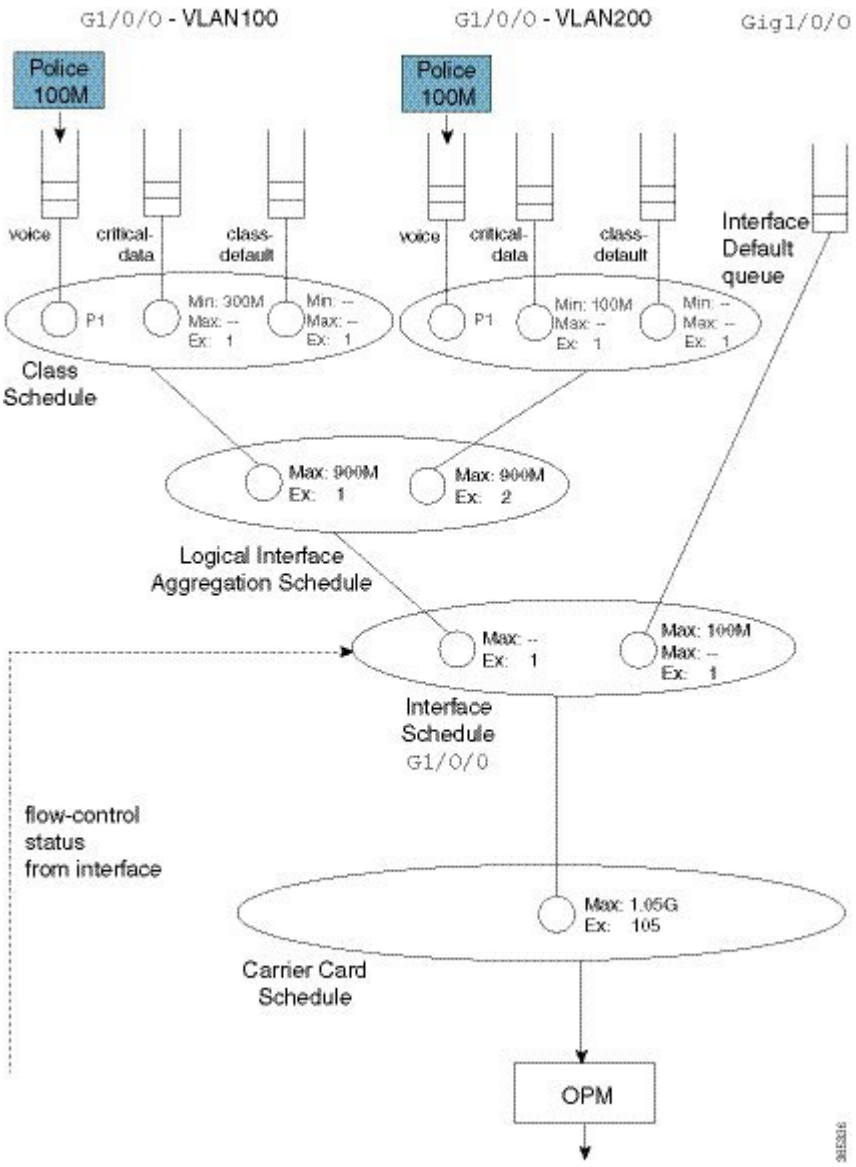
これは特殊なケースで、このポリシーのスケジュールを作成するのではなく、単にスケジュール エントリを定義します。このエンティティを *collapsed class-default* と呼びます.

この概念の重要性を理解するために、別のVLAN（VLAN200）にポリシーを追加してみましょう。（トピックの冒頭に記載されている`policy-map parent100`に関連して、アスタリスクを追加しました）：

```
policy-map child200
  class voice
    priority
    police cir 100m
  class critical-data
    bandwidth 100000
!
policy-map parent200
  class class-default
    shape average 900m                    ****
    bandwidth remaining ratio 2
    service-policy child200
!
int g1/0/0.200
  encaps dot1q 200
  service-policy out parent200
```

完全なスケジューリング階層は、次のようになります。

図 *27 :* 輻輳を処理し、帯域幅の浪費を回避するための完全な階層型スケジューリング フレームワーク



2 番目の親ポリシー（VLAN200 へのポリシー）では、帯域幅余剰比率に 2 を指定し、VLAN 間の均等性を制御しています。 Qos スケジューリングの章で説明したように、フラット ポリシーの親ポリシーにはピアが存在します。これにより、**bandwidth remaining ratio** または **bandwidth remaining percent** コマンドを使用して超過重量を指定することができます。「親ポリシーでのシェーピング」コンストラクトでは、ピアは存在しません. QoS ポリシーマップを設定すると、QoS は論理インターフェイス集約スケジュールでピアとして具体化するものを認識できません。したがって、**bandwidth remaining ratio** も **bandwidth remaining percent** コマンドもサポートされていません。

この完全なスケジューリング階層は、シスコ モジュラ QoS CLI（MQC）および階層型スケジューリング フレームワーク（HQF）の利点を真に強調しています。どのようなインタフェースでも、階層は決定論的です。次に転送されるパケットは明確にわかっています。すべての輻

輳ポイントを処理するスケジュールがあるため、輻輳が発生する可能性がある場所に関係なく、帯域幅は無駄になりません。

論理インターフェイス上のポリシーの利点

ポリシーマップを論理インターフェイスに適用する機能には、スケーリングされた環境での管理と簡単な設定という大きな利点があります。論理インターフェイスごとに、ポリシーマップを再利用または作成できます。つまり、イーサネットタイプのインターフェイスに設定されている 1000 個の VLAN それぞれにポリシーマップを適用できます。個々の論理インターフェイスの QoS 統計データを確認するには、 **show policy-map interface** *interface-name* を発行します。

利点と危険性は紙一重となる場合があります。利用可能な物理帯域幅が親シェーパーの合計を超える場合は、単一の論理インターフェイスを単独で確認するだけで十分です。 ただし、親シェーパーの合計が使用可能な物理帯域幅を超える場合は、論理インターフェイス間の競合、および個々のインターフェイスに対して本当に保証されている帯域幅を考慮する必要があります。個別のインターフェイスを単独で確認すると、誤解を招く可能性が高くなります。

複数のポリシー定義と制限

複数ポリシー（*MPOL*）を使用し、ポリシーマップが論理インターフェイスに適用されている場合に、その論理インターフェイスがバインドされている物理インターフェイス（VLAN サブインターフェイスや物理イーサネットインターフェイスなど）にポリシーマップを同時に適用している状況について説明します。

MPOL は、ポリシーマップが同じ物理インターフェイスにバインドされている異なる論理インターフェイス タイプに付加されているインスタンスを参照することもできます。たとえば、VLAN サブインターフェイスとトンネル インターフェイスの両方に適用されたポリシーがあるとします。この場合、両方とも同じ物理インターフェイスから出ます。

現在、ASR 1000 シリーズ アグリゲーション サービス ルータは MPOL の非常に限られた実装に対応しています。論理インターフェイスに適用されたポリシーマップがある場合、物理インターフェイスに適用できる唯一のポリシーは、次の例のように class-default とシェーパーのみが設定されたフラットです。このトポロジは、（プロバイダーからの）サービス レートが物理アクセス レートと異なる場合に役立ちます。たとえば、GigabitEthernet インターフェイスを介してプロバイダと接続しており、200 Mbps のサービスのみを支払っているとします。サービス プロバイダはそのレートを超えるトラフィックを監視するため、送信するすべてのものを 200 Mbps にシェーピングし、その帯域幅をローカルに割り当てる必要があります。

（注） ポリシーを論理インターフェイスに適用する前に、物理インターフェイスに適用する必要があります。 さらに、単一の物理インターフェイスにバインドされた複数の論理インターフェイス タイプにポリシーマップを適用することはできません。

それでは、2 つの VLAN サブインターフェイスに適用されたポリシーの前の例（「親ポリシーでのシェーピング/子ポリシー上のキューイング （31 ページ）」を参照）に戻り、物理イン

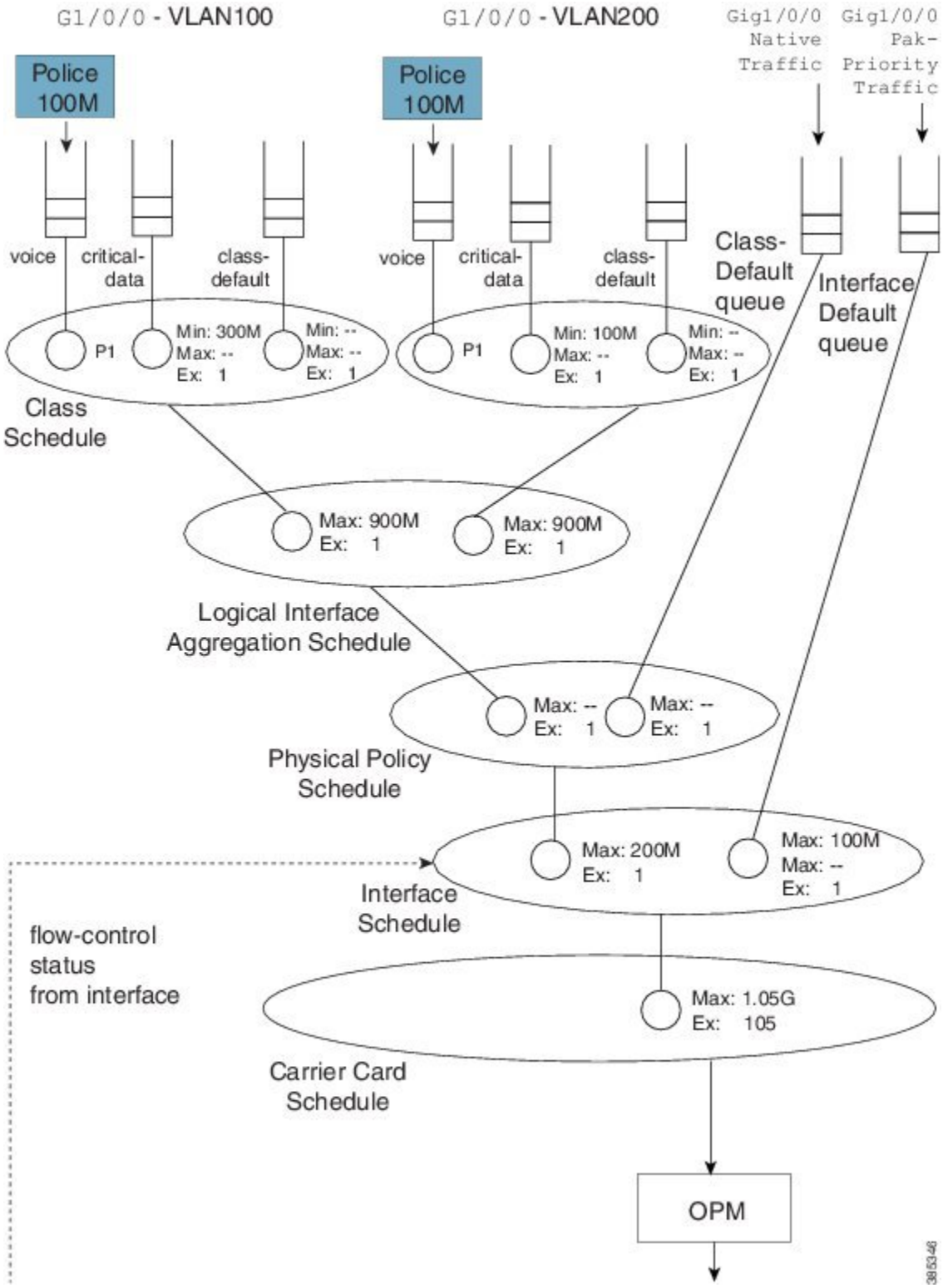
ターフェイスに 200 Mbps のシェーパーを追加します。完全な設定は次のようになります。アスタリスクは、これと以前の設定との違いを示しています。

```
policy-map physical-shaper                          ****
  class class-default                               ****
    shape average 200m                              ****
 !
policy-map child100
  class voice
    priority
    police cir 100m
  class critical-data
    bandwidth 300000
!
policy-map child200
  class voice
    priority
    police cir 100m
  class critical-data
    bandwidth 100000
!
policy-map parent100
  class class-default
    shape average 900m
    service-policy child100
!
policy-map parent200
  class class-default
    shape average 900m
    bandwidth remaining ratio 2
    service-policy child200
!
! Note - must attach physical policy before logical policies
!
int g1/0/0                                           ****
  service-policy output physical-shaper              ****
!
int g1/0/0.100
  encaps dot1q 100
  service-policy out parent100
!
int g1/0/0.200
  encaps dot1q 200
  service-policy out parent200
```

物理インターフェイスを介して送信されるユーザー トラフィックに使用されるキューと同様に、別のスケジュールについて学んだことを忘れないでください。 論理インターフェース集約スケジュールは、インターフェース スケジュールの直接の子としてではなく、物理ポリシースケジュールの子として作成されました。これで、論理インターフェイスを通過するトラフィックと物理インターフェイスを通過するユーザ トラフィックの組み合わせは、200 Mbps にシェープされました。

完全なスケジューリング階層は次のようになります。

図 *28 :* 物理ポリシー スケジュールの子としての論理インターフェイス集約の作成



階層型ポリシーマップ

前の章では、ポリシーマップが論理インターフェイスに適用されている場合の階層の作成方法を学びました。2 番目のアプローチは、階層型ポリシーマップを使用し、希望する階層を明示的に作成します。この方法を使用すると、ある程度の柔軟性が得られますが、ある程度の拡張性は失います。（論理インターフェイスに関するポリシーを使用すると、スケーリングされた環境で管理できることを思い出してください。）ASR 1000 シリーズ アグリゲーション サービス ルータは、ポリシーマップで最大 1,000 のクラスをサポートします。つまり、表記できる論理インターフェイスの最大数は 1,000 です。

階層型ポリシーマップ内のクラスに属するには、パケットは子および（すべての）親の分類規則と一致する必要があります。 以前の VLAN の例では、親クラスで VLAN ID ベースの分類を使用し、子クラスで DSCP ベースの分類を使用する方法を学びました。

次の設定では、MPOL 物理シェーパー（「複数のポリシー定義と制限（38 ページ）」を参照）と同じような動作を実現する方法を示しています。 ここでは、3 レベルの階層型ポリシーマップ（サポートしている最大レイヤ数）を使用します。

親ポリシーには class-default しかありません。つまり、インターフェイスを通過するすべてのトラフィックはこのクラスに属します。

```
policy-map physicalshaper
  class class-default
    shape average 200m
    service-policy vlansharing
```

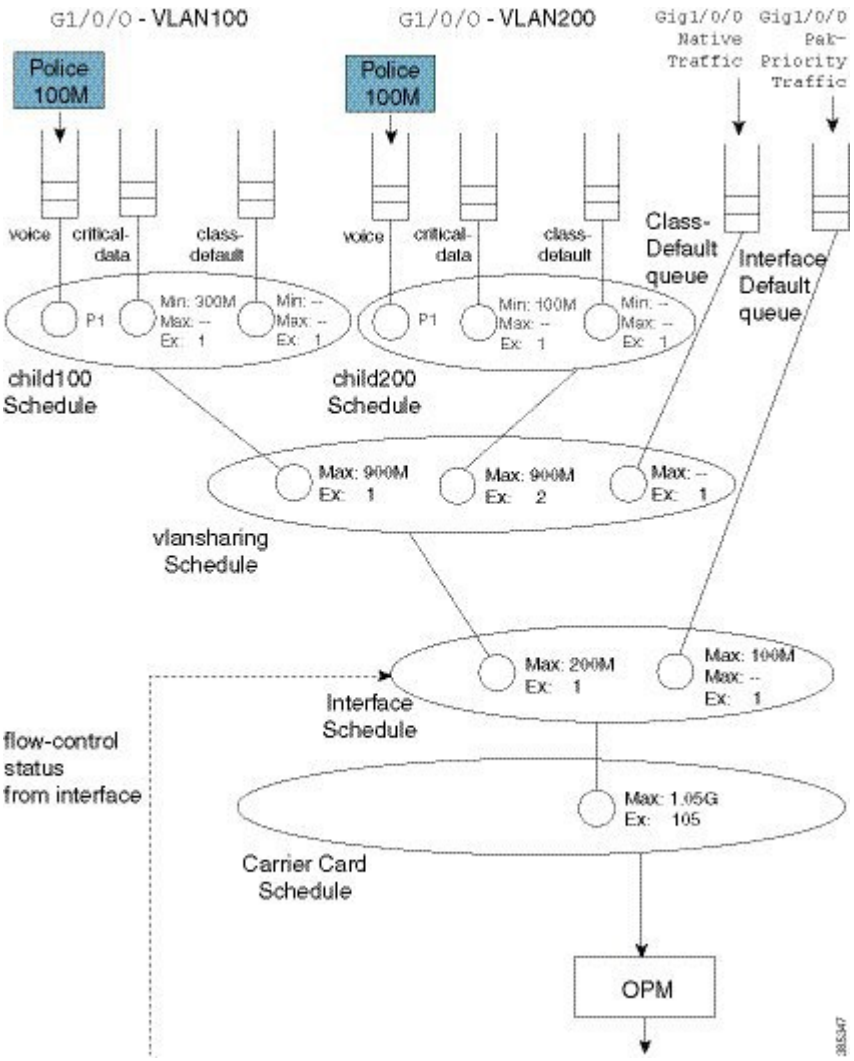
子レベルには VLAN ベースの分類があります。 VLAN 100 または VLAN 200 に属するトラフィックは、ユーザ定義クラスの 1 つに分類されます。（さらに、このポリシーには、他の VLAN からのトラフィックまたは VLAN タグなしのトラフィックをキャプチャする暗黙の class-default があります。）各 VLAN クラスには、DSCP に基づいてトラフィックをさらに分類するポリシーがあります。

```
class-map vlan100
  match vlan 100
class-map vlan200
  match vlan 200
class-map voice
  match dscp ef
class-map critical-data
  match dscp af21
        !
policy-map child100
 class voice
   priority
   police cir 100m
 class critical-data
   bandwidth 300000
!
policy-map child200
 class voice
   priority
   police cir 100m
 class critical-data
   bandwidth 100000
```

```
!
policy-map vlansharing
 class vlan100
   shape average 900m
   bandwidth remaining ratio 1
   service-policy child100
 class vlan200
   shape average 900m
   bandwidth remaining ratio 2
   service-policy child200
!
policy-map physicalshaper
   class class-default
     shape average 200m
    service-policy vlansharing
 !
int g1/0/0
  service-policy output physicalshaper
```

上記の設定に基づいて作成された階層は、次のようになります。

図 *29*：階層を明示的に作成するための階層型ポリシーマップ



この階層を前の MPOL の例（図 25）と比較すると、若干の違いがあることがわかります。

まず、ネイティブ インターフェイス トラフィック（VLAN 100 と 200 のどちらでもないトラフィック）が、各 VLAN のスケジュール エントリと vlansharing スケジュールを共有するようになりました。MPOL の例では、ネイティブ トラフィックは、すべての（両方の）VLAN（使用可能帯域幅の 1/2）と同等の帯域を受け取りました。 これとは対照的に、この階層では、同じスケジュールで VLAN と競合するため、使用可能な帯域幅の 1/(1 + 2 + 1) だけが保証されます。

次に、物理インターフェイス上の単一のポリシーマップでは、単一の VLAN の統計データしか見ることができなくなりました。MPOL の例とこのコードを比較してください。

```
int g1/0/0
  service-policy output physical-shaper
!
int g1/0/0.100
  encaps dot1q 100
```

```
  service-policy out parent100
!
int g1/0/0.200
  encaps dot1q 200
  service-policy out parent200
```

次を使用すると：

```
int g1/0/0
  service-policy output physicalshaper
```

show policy-map interface GigabitEthernet1/0/0 コマンドの出力には、階層型ポリシーマップのすべてのレベルが反映されます。

階層型ポリシーマップにより、論理インターフェイス上のポリシーマップでは実現不可能な柔軟性が強化されます。次の例は、この動作について説明しています。

例 1 異なるトラフィック クラスにキューを追加する

MPOL の例（および以下のコード）では、物理インターフェイス ポリシーには class-default とそのクラスのシェーパーしか含めることがないことをを学びました。

```
policy-map physical-shaper
  class class-default
    shape average 200m
```

つまり、ネイティブインターフェイスを介して転送された固有のクラスのトラフィック（VLAN タグなしのトラフィック）に対して異なる方法で処理することはできません。

これとは対照的に、階層コンストラクトでは、（物理インターフェイスを介して）転送するトラフィックのクラスごとにキューを追加できます。たとえば、物理インターフェイス上の音声トラフィックにプライオリティクラスを追加する場合は、vlansharing ポリシーマップを次のように変更できます（アスタリスクを参照）。

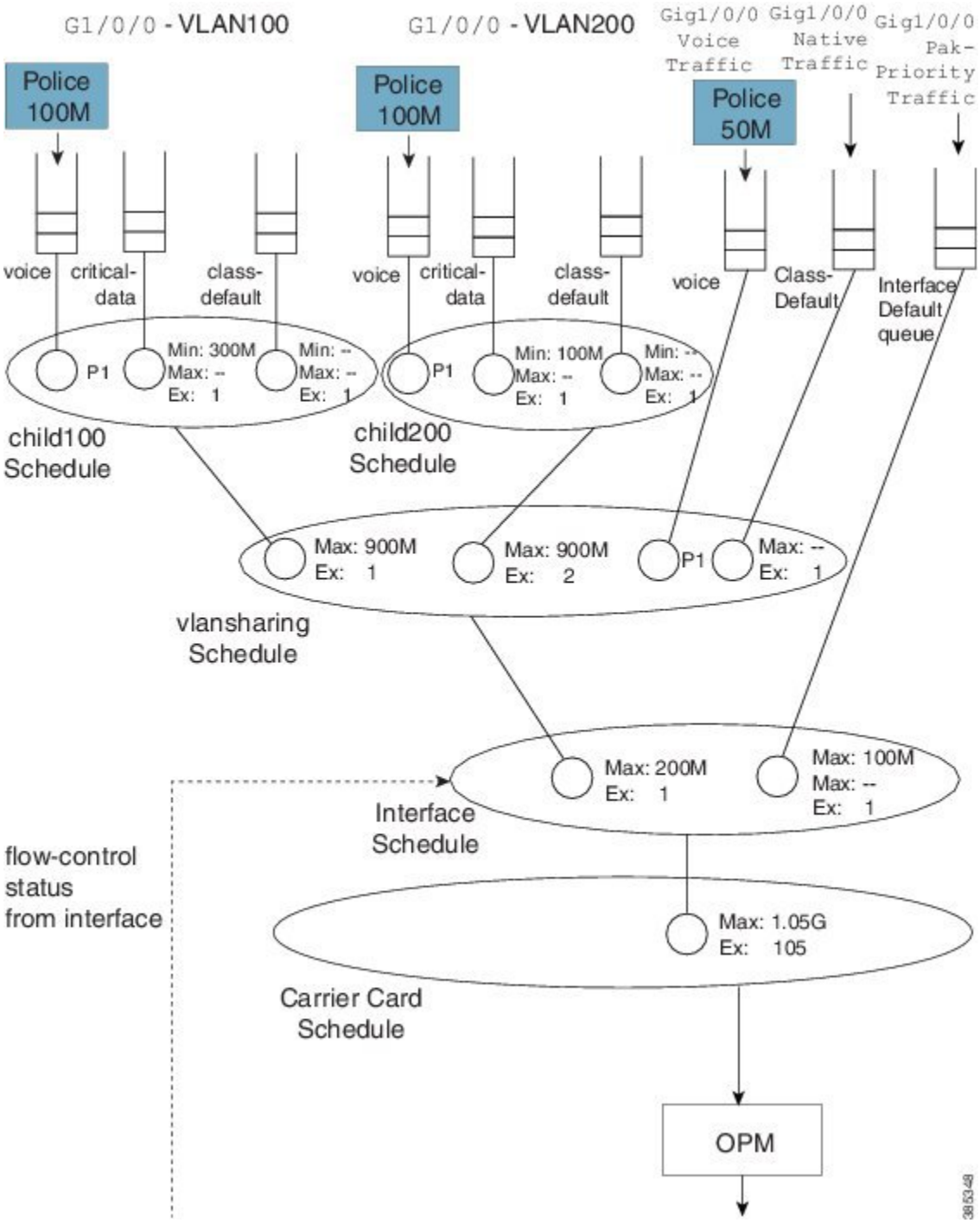
```
class-map vlan100
  match vlan 100
class-map vlan200
  match vlan 200
class-map voice
  match dscp ef
class-map critical-data
  match dscp af21
        !
policy-map child100
 class voice
   priority
   police cir 100m
 class critical-data
   bandwidth 300000
!
policy-map child200
 class voice
   priority
   police cir 100m
 class critical-data
   bandwidth 100000
!
```

```
policy-map vlansharing
  class vlan100
    shape average 900m
    bandwidth remaining ratio 1
    service-policy child100
  class vlan200
    shape average 900m
    bandwidth remaining ratio 2
    service-policy child200
  class voice                              ****
    priority                               ****
    police cir 50m                         ****
!
policy-map physicalshaper
  class class-default
    shape average 200m
    service-policy vlansharing
 !

int g1/0/0
  service-policy output physicalshaper
```

この設定の階層は、次のようになります。

図 *30*：階層コンストラクトで異なるトラフィック クラスのキューを表す



EF の DSF コードポイントでマークされているが、VLAN ID 100 または 200 でタグ付けされていないトラフィックをキャプチャする新しいキャプチャに留意してください。

この階層では、ローカル キューからの P1 トラフィック（Gig1/0/0 音声トラフィック）が、VLAN共有スケジュール内の優先度伝播トラフィックと競合することに注意してください（「優

先度の伝搬の概念 （6 ページ）」を参照）。そのような場合には、プライオリティが設定されたローカルエントリは優先度伝播トラフィックの前に処理されます。つまり、物理インターフェイス（Gig1/0/0）からの音声パケットは、VLAN 100 または 200 からの音声パケットよりわずかに高い優先順位を持ちます。他のクラスのキューが消費されるのを回避するために、プライオリティ キューでアドミッション コントロールを使用します。

例 2 異なる論理インターフェイス タイプへのポリシーの適用

「論理インターフェイスに適用されたポリシーマップ （29ページ）」では、同じ物理インターフェイス上の異なる論理インターフェイス タイプにポリシーを適用できないことを学びました。この制限は、階層型 class-map には適用されません。

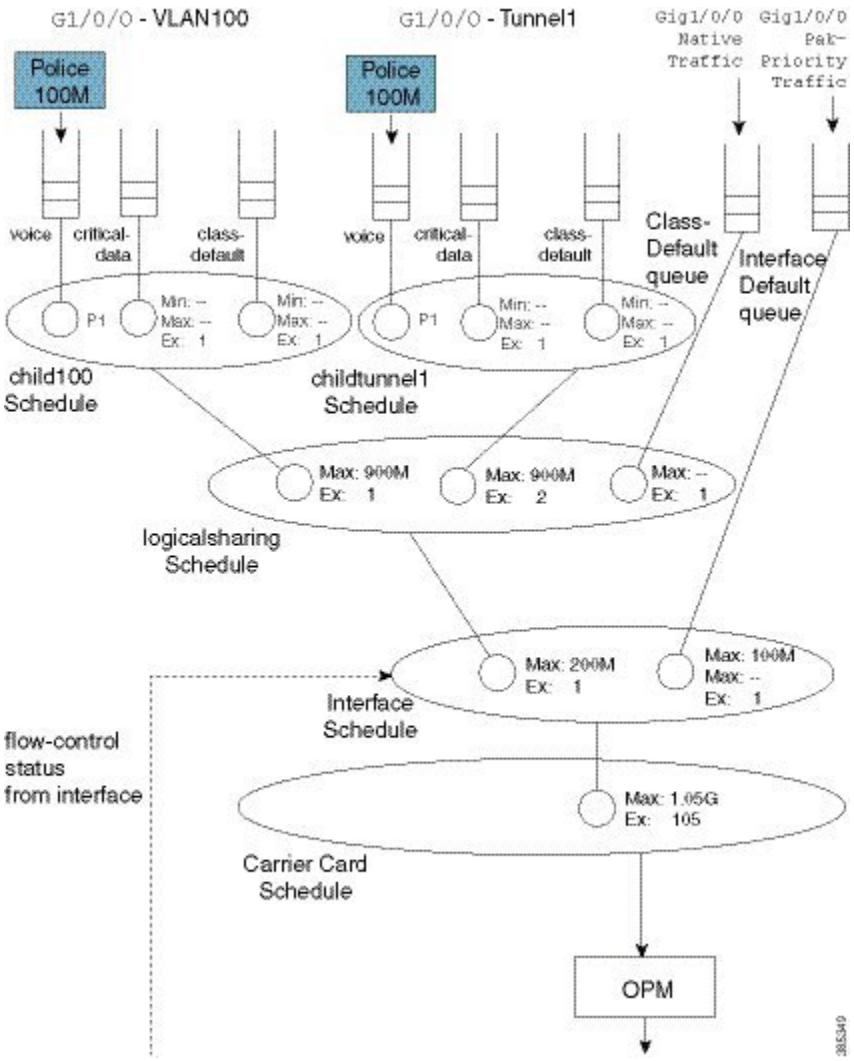
両方とも同じ物理インターフェイスを通過するトンネルで、VLAN 100 用に 1 つの子、QoS 用に 1 つの子が必要だとします。 同じポリシーマップ内で、アクセス リストを使用してトンネル トラフィックを分類し、VLAN ID を使用して VLAN トラフィックを分類できます（アスタリスクを参照）。

```
ip access-list extended tunnel1traffic
  permit ip host 192.168.1.1 host 10.0.0.1
!
class-map vlan100
  match vlan 100
class-map tunnel1traffic
  match access-group name tunnel1traffic
!
class-map voice
  match dscp ef
class-map critical-data
  match dscp af21
        !
policy-map child
  class voice
    priority
    police cir 100m
  class critical-data
    bandwidth remaining ratio 1
!
policy-map logicalsharing                  ****
  class vlan100
    shape average 900m
    bandwidth remaining ratio 1
    service-policy child
  class tunnel1traffic
    shape average 900m
    bandwidth remaining ratio 2
    service-policy child
!
policy-map physicalshaper
  class class-default
    shape average 200m
    service-policy vlansharing
 !

int g1/0/0
  service-policy output physicalshaper
```

この設定の階層は、次のようになります。

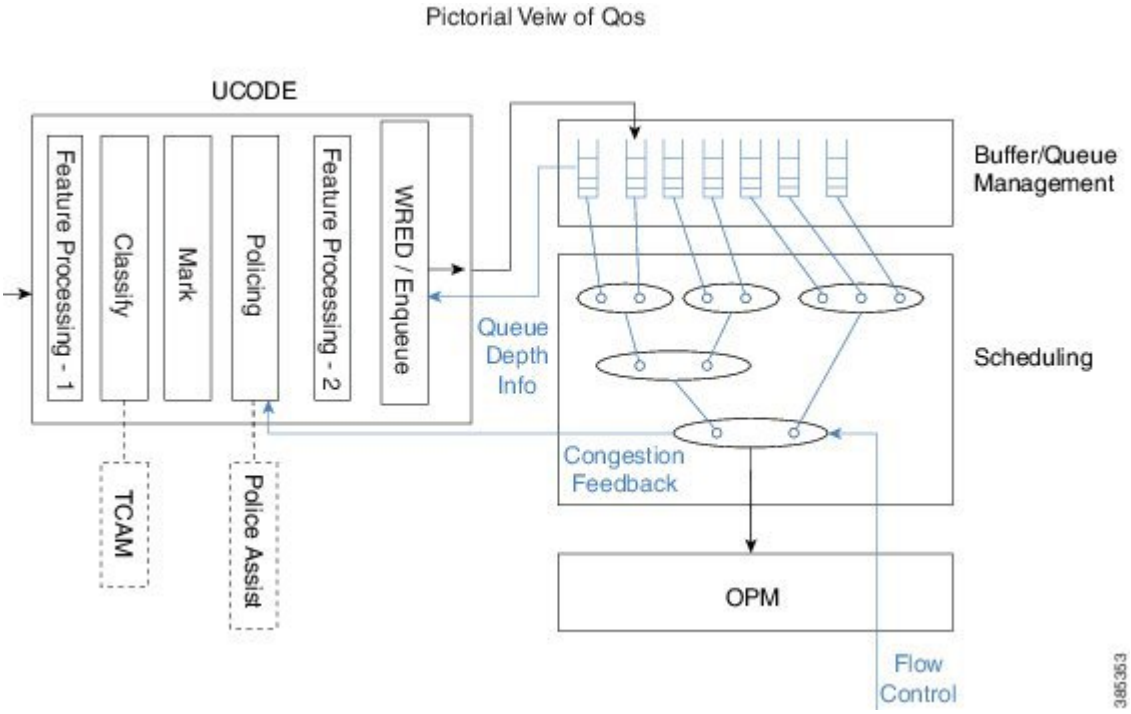
図 *31* : 異なる論理インターフェイス タイプへのポリシーの適用



オーバーヘッド アカウンティングにおける留意点

ポリシングの章では、ポリシング長（ポリサーが設定済みポリシング レートへの適合性を評価するときに、パケットの長さを認識する方法。「ポリサー レート計算（オーバーヘッド アカウンティング）に含まれるもの」を参照）の概念について学びました。同様に、スケジューリングの章では、スケジューリング長（設定済みのスケジューラ レートへの適合性を評価する際のパケットの長さを考慮する方法。「スケジューリング レート計算（オーバーヘッド アカウンティング）に含まれるもの」を参照）の概念について学びました。慣例により、どちらの場合も、レイヤ2ヘッダーとデータグラムの長さ、およびCRCまたはパケット間のオーバーヘッドを除外します。

階層型スケジューリング構成では、ポリシングとスケジューリングの長さが異なる場合があります。 これを理解するために、機能の実行順序について考慮します。



ASR 1000 シリーズアグリゲーション サービス ルータでは、キューイングとスケジューリングはハードウェアで実行されます。パケットをエンキューした後、ハードウェアにより制御が実行され、それ以上の処理は実行されません。パケットにはすべてのヘッダーがあり、ワイヤを通過する準備ができている必要があります。予想どおり、非キューイング機能は、処理要素のうちの 1 つのマイクロコードで実行され（場合によってはハードウェア支援を用いて）ます。

2 つのシナリオについて考えてみます。

GRE トンネルでの QoS キューイング ポリシーの設定

（最終的に外部 IP/GRE ヘッダーにカプセル化される）着信 IP パケットを分類するときは、元の IP パケットのみを確認します。その結果、分類統計データは、その時点で欠落しているため、外側の IP/GRE ヘッダーを除外します。 図で示されているように、ここではポリサーのマーキングと評価を行います。分類長と同様に、ポリシング長には外部 IP/GRE ヘッダーも出力レイヤ2ヘッダーも含まれません。パケットがどの物理インターフェイスまたはカプセル化タイプを出力するのかまだわからないためです。QoS 非キューイング機能の後、外部 IP/GRE ヘッダーと最終的な出力インターフェイス用の適切なレイヤ2ヘッダーを追加して、パケットの処理を続けます。すべての処理が終了したら、パケットを WRED/Enqueue ブロックにパスします。この動作により、すべてのヘッダーが追加されたパケットが、ハードウェア内の適切な出力キューに配置されます。これで、スケジューリング長には、外側の IP/GRE とレイヤ2ヘッダーが含まれるようになりました。

物理出力インターフェイスでの QoS ポリシーの設定

結果は異なります。トンネル上の機能を確認すると、QoS が設定されていないため、機能処理に進みます。QoS ポリシーに到達する前に、すべてのトンネル処理を完了し、出力ヘッダーを追加します。これで、分類統計データとポリシング長には外部ヘッダーが含まれるようになり、ポリシングとスケジューリング長が一致するようになります。

確認

すべての QoS 設定作業において、階層型スケジューリングの設定を検証するための主要ツールは、**show policy-map interface** *interface-name* コマンドです。このコマンドの出力は、構成を階層化した方法を反映して、階層的に編成されています。

たとえば、物理インターフェイスに付加された階層型ポリシーでは、**show policy-map interface** インターフェイス名 **| include Class** を使用してその階層を表示します。

```
show policy-map int g1/0/0 | inc Class

    Class-map: class-default (match-any)
        Class-map: vlan100 (match-all)
            Class-map: voice (match-all)
            Class-map: critical-data (match-all)
            Class-map: class-default (match-any)
        Class-map: vlan200 (match-all)
            Class-map: voice (match-all)
            Class-map: critical-data (match-all)
            Class-map: class-default (match-any)
        Class-map: vlan300 (match-all)
            Class-map: voice (match-all)
            Class-map: class-default (match-any)
        Class-map: voice (match-all)
        Class-map: class-default (match-any)
```

この例では、インターフェイス GigabitEthernet1/0/0 に3 レベルの階層型ポリシーを適用しました。 class-map のインデントはその階層を表します。子ポリシーを含むクラス内の `Service-policy: <ポリシーマップ名>` は、次にインデントされているセクションが子ポリシーに関連することを示します。

```
  Class-map: vlan100 (match-all)
          0 packets, 0 bytes
          5 minute offered rate 0000 bps, drop rate 0000 bps
          Match: vlan  100
          Queueing                                                 ****
          queue limit 3748 packets                                 ****
          (queue depth/total drops/no-buffer drops) 0/0/0          ****
          (pkts output/bytes output) 0/0
          shape (average) cir 900000000, bc 3600000, be 3600000
          target shape rate 900000000
          bandwidth remaining ratio 1

          Service-policy : child100

            queue stats for all priority classes:
              Queueing
              queue limit 512 packets
              (queue depth/total drops/no-buffer drops) 0/0/0
              (pkts output/bytes output) 0/0
```

```
Class-map: voice (match-all)
  0 packets, 0 bytes
  5 minute offered rate 0000 bps, drop rate 0000 bps
  Match:  dscp ef (46)
  Priority: Strict, b/w exceed drops: 0

  police:
      cir 100000000 bps, bc 3125000 bytes
    conformed 0 packets, 0 bytes; actions:
      transmit
    exceeded 0 packets, 0 bytes; actions:
      drop
    conformed 0000 bps, exceeded 0000 bps

Class-map: critical-data (match-all)
  0 packets, 0 bytes
  5 minute offered rate 0000 bps, drop rate 0000 bps
  Match:  dscp af11 (10)
  Match:  dscp af21 (18)
  Queueing
  queue limit 1249 packets
  (queue depth/total drops/no-buffer drops) 0/0/0
  (pkts output/bytes output) 0/0
  bandwidth 300000 kbps

Class-map: class-default (match-any)
  0 packets, 0 bytes
  5 minute offered rate 0000 bps, drop rate 0000 bps
  Match: any

  queue limit 3748 packets
  (queue depth/total drops/no-buffer drops) 0/0/0
  (pkts output/bytes output) 0/0
```

階層型スケジューリングを含むポリシーの**show**コマンド出力については、親クラス内のキューに関連するすべての情報は有効ではないことに注意してください（上記の例ではアスタリスクで強調表示されています）。**show policy-map interface** コマンドの出力形式は、IOS がキューの階層をソフトウェアに実際に実装したときに作成されました。 ASR 1000 シリーズ アグリゲーション サービス ルータ ハードウェアは、リーフにのみ存在するスケジュールとキューの階層を実装します。IOS コントロールプレーンにより、キュー制限がまだ計算され、表示されますが、それを使用することはありません。したがって、この値をチューニングしても意味がありません。

確認