Cisco QoS配置说明（CBWFQ/LLQ/PQ/CQ/WFQ）

CBWFQ 基于类别的加权公平排队,通常使用ACL定义数据流类别，并将注入宽带和队列限制等参数应用于这些类别.

CBWFQ特点:

1)能够给不同的类保障一定的带宽

2)对传统的WFQ作了扩展支持用户自己定义流量的分类:

3)队列的个数和类别是一一对应,给每个class 保留带宽

CBWFQ与WFQ的区别：

WFQ: 用户无法控制分类，由HASH算法自己决定

CBWFQ:让用户对流量自己来分类

WFQ 对正常流量 处理没问题，但是对语音流量显得”太公平”(语音要求低延迟)

CBWFQ:考虑到公平特性，并没有考虑到语音的应用

CBWFQ Configuration:

CBWFQ(config)#

CBWFQ(config)#class-map match-any CBWFQ1

CBWFQ(config-cmap)#match dscp 6

CBWFQ(config-cmap)#match protocol http //两个条件，满足其中一个就可以匹配CBWFQ1//

CBWFQ(config-cmap)#exit

CBWFQ(config)#

CBWFQ(config)#class-map match-all CBWFQ2

CBWFQ(config-cmap)#match precedence 3

CBWFQ(config-cmap)#match protocol telnet //两个条件必须全部满足才能匹配CBWFQ2//

CBWFQ(config-cmap)#exit

CBWFQ(config)#

CBWFQ(config)#policy-map CBWFQ

CBWFQ(config-pmap)#class CBWFQ1 //调用class-map CBWFQ1//

CBWFQ(config-pmap-c)#bandwidth 60

CBWFQ(config-pmap)#

CBWFQ(config-pmap)#class CBWFQ2

CBWFQ(config-pmap-c)#bandwidth 30

CBWFQ(config)#

CBWFQ(config)#int s0/0

CBWFQ(config-if)#service-policy output CBWFQ   //CBWFQ只能在出方向上调用//

CBWFQ#

查看：

CBWFQ#

CBWFQ#show class-map

Class Map match-any class-default (id 0)

Match any

Class Map match-any CBWFQ1 (id 1)

Match dscp 6

Match protocol http

Class Map match-all CBWFQ2 (id 2)

Match precedence 3

CBWFQ#

CBWFQ#

CBWFQ#show policy-map

Policy Map CBWFQ

Class CBWFQ1

Bandwidth 60 (kbps) Max Threshold 64 (packets)

Class CBWFQ2

Bandwidth 30 (kbps) Max Threshold 64 (packets)

CBWFQ#

CBWFQ#

CBWFQ(config)#

CBWFQ(config)#policy-map CBWFQ

CBWFQ(config-pmap)#class CBWFQ1

CBWFQ(config-pmap-c)#queue-limit 30 // 定义每个队能存放的报文数量，超过后丢包方式：Tail drop//

CBWFQ#

CBWFQ#

CBWFQ#show policy-map

Policy Map CBWFQ

Class CBWFQ1

Bandwidth 60 (kbps) Max Threshold 30 (packets)

Class CBWFQ2

Bandwidth 30 (kbps) Max Threshold 64 (packets)

CBWFQ#

配置实例：

一家公司需求；HTTP流量保障256Kbps带宽，FTP流量保证512Kbps带宽，禁止BT流量.

CBWFQ(config)#

CBWFQ(config)#class-map class\_HTTP   //定义一个匹配HTTP的类//

CBWFQ(config-cmap)#match protocol http

CBWFQ(config)#

CBWFQ(config)#class-map class\_FTP

CBWFQ(config-cmap)#match protocol ftp

CBWFQ(config)#

CBWFQ(config)#class-map class\_BT

CBWFQ(config-cmap)#match protocol bittorrent

CBWFQ(config)#

CBWFQ(config)#policy-map CBWFQ   //定义策略，调用类class//

CBWFQ(config-pmap)#class class\_HTTP

CBWFQ(config-pmap-c)#bandwidth 256

CBWFQ(config-pmap)#

CBWFQ(config-pmap)#class class\_FTP

CBWFQ(config-pmap-c)#bandwidth 512

CBWFQ(config-pmap)#

CBWFQ(config-pmap)#class class\_BT

CBWFQ(config-pmap-c)#drop

CBWFQ(config-pmap)#

CBWFQ(config-pmap)#class class-default

CBWFQ(config-pmap-c)#fair-queue  //网络中剩下的流量除了HTTP,FTP之使用WFQ放到fair-queue中了//

CBWFQ(config)#

CBWFQ#

CBWFQ(config)#int s0/0

CBWFQ(config-if)#service-policy output CBWFQ

CBWFQ(config)#

QOS-WFQ

Weighted Fair Queue，加权公平队列。WFQ将分组按照不同的业务流、不同的IP优先级，自动按照HASH算法，划分成不同的队列，在保证高优先级业务的同时，按照配置权重，将带宽公平地分给低优先级别的业务。Quidway[路由器](https://www.2cto.com/net/router/" \t "/Users/kingdom/Documents\\x/_blank)在每个接口上最大支持4096个队列，在此范围之内，网络治理员可以配置队列的数目和相应权值。WFQ的优点是对所有应用都能比较公平地提供服务质量，对于要求较高的业务可以通过设置权重保障优先级，能智能划分队列和调度，配置相对简单，对网络治理员要求相对较低；缺点是由于相对公平，对于QOS要求极高的应用，不能象PQ那样绝对保证，策略计算复杂，比较消耗路由器处理能力。它适用于应用较复杂，并且应用相对公平的网络，如Internet网。WFQ的思想:

i,为每个流创建一个专用的队列，避免队列的饥饿，延迟，抖动 等

ii,在所有流间公平，正确地分配带宽

iii,WFQ使用 [IP优先级] 作为分配带宽的权重 在CISCO路由器上，接口小于E1的链路会默认启用WFQ.

注：在WFQ中，weight的计算方式为4096/(IP优先级+1)或者32384r/(IP优先级+1)

因此在show queue中看到的weight值越大，表示权重越低。

WFQ优点

1.配置简单(不用手工分类)

2.保证所有的流都有一定的带宽

3.丢弃野蛮流量

4.大多数平台上都支持

5.支持所有[IOS](https://www.2cto.com/kf/yidong/iphone/" \t "/Users/kingdom/Documents\\x/_blank)版本(11.0以上)

WFQ缺点

1.每个子队列都继承了FIFO的缺点

2.多个不同的流可能会被分入同一个队列(流的数量超过了配置的队列数)

3.不支持手工分类

4.不能提供固定带宽保证

5.因为使用了复杂的分类和调度机制，对[系统](https://www.2cto.com/os/" \t "/Users/kingdom/Documents\\x/_blank)资源有一定的限制

配置命令:

route(config-if):fair-queue cdt dynamic-queues reservable-queues

动态队列个数 保留队列个数

reservable-queues：

保留队列个数：针对RSVP流,可以保留一定的队列，缺省是0，范围0~1000

dynamic-queues：

动态队列个数: 缺省是256,流确实很多，可以调大，最大4096

cdt:

每个队自己的长度

一个数据排到第一个队中，cdt=64，如果该队的报文己达到64,新的报文丢包!

每个队中排的报文数量是有限的

所有队列加起来，上限:

router(config-if)#hold-queue max-limit out 缺省1000

一个报文是否在WFQ中排到队列中的二个因素:

1)–本队列是否己满

2)–所有队列是否超出队列上限   —-超出报文丢弃

WFQ Configuration:

WFQ(config)#

WFQ(config)#int s0/1

WFQ(config-if)#fair-queue //表面接口启用WFQ，CISCO路由器小于E1会默认启用//

WFQ(config-if)#fair-queue 128 1024 100 //设置CDT为128，动态队列数量为1024个，保留队列个数100//

WFQ(config-if)#hold-queue 1200 out //所有队列加起来CDT上限是1200//

WFQ(config-if)#

查看：

WFQ#

WFQ#show int s0/1

Serial0/1 is up, line protocol is up

Queueing strategy: weighted fair

Output queue: 0/1200/64/0 (size/max total/threshold/drops)

Conversations   0/1/256 (active/max active/max total)

Reserved Conversations 0/0 (allocated/max allocated)

Available Bandwidth 1158 kilobits/sec

5 minute input rate 3000 bits/sec, 3 packets/sec

5 minute output rate 3000 bits/sec, 3 packets/sec

1001 packets input, 97009 bytes, 0 no buffer

Received 0 broadcasts, 0 runts, 0 giants, 0 throttles

0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort

1004 packets output, 96674 bytes, 0 underruns

0 output errors, 0 collisions, 3 interface resets

0 output buffer failures, 0 output buffers swapped out

3 carrier transitions     DCD=up   DSR=up   DTR=up   RTS=up   CTS=up

WFQ#

QOS-CQ

Customized Queue 用户定制队列

CQ使用了17个子队列(其中0子队列是PQ队列，优先级很高，留给系统使用)，CQ使用WRR(Round-Robin)机制。首先谈谈RR机制，RR在处理完一个队列的一个数据包之后，会接着处理另一个队列的一个数据包，一直下去，最后又从第一个队列开始轮流处理每个队列中的数据包，RR中的每个队列的优先级都是一样的。RR的改进版是WRR(WeightedRound-Robin)，WRR允许用户为每个队列分配一个权值，根据这个权值，每个队列都能获得一定的接口带宽。在CQ中，权值就是一次轮循中可以转发的字节数。

前面说到了0队列是PQ队列，实际上可以把其他队列也设置成PQ队列：

可以通过以下命令来设置：

queue-list list-number lowest-custom queue-number

比如命令queue-list 1 lowest-custom 3，说明0，1，2都是优先级队列，3以及3以上编号的队列都是定制队列。

CQ configuration:

r2(config)#

r2(config)#access-list 101 permit ip any any precedence 5

r2(config)#

r2(config)#queue-list 16 protocol ip 1 list 101 //把ACL101定义的数据流映射到子队列1中//

r2(config)#queue-list 16 queue 1 limit 40 //设置子队列1的队列深度为40个数据包//

r2(config)#queue-list 16 lowest-custom 2  //设置queue 0,1为优先级队列PQ，其余的为CQ//

r2(config)#queue-list 16 interface s0/0 2 //把s0/0接口进入的流量映射到子队列2中//

r2(config)#queue-list 16 queue 2 byte-count 3000 //设置子队列2在一个轮循内可以传输3000字节数据包

r2(config)#queue-list 16 protocol ip 3 //把所有IP流量映射到子队列3中//

r2(config)#queue-list 16 queue 3 byte-count 5000

r2(config)#queue-list 16 default 4 //其它所有流量映射到子队列4中//

r2(config)#

r2(config)#

r2(config)#int s0/1

r2(config-if)#custom-queue-list 16 //应用CQ到接口s0/1上//

r2(config)#

QOS-PQ

PQ使用了4个子队列，优先级分别是high，medium，normal，low。PQ会先服务高优先级的子队列，若高优先级子队列里没有数据后，再服务中等优先级子队列，依次类推。如果PQ正在服务中等优先级子队列，但是高优先级里又来了数据包，则PQ会中断中等优先级子队列的服务，转而服务高优先级子队列。每一个子队列都有一个最大队列深度(queue-size)，如果达到了最大队列深度，则进行尾丢弃。

PQ优点

1.对高优先级的数据流提供了低延迟的转发

2.大多数平台上都支持该队列机制

3.支持所有的IOS版本(10.0以上)

PQ缺点

1.对单一子队列而言，会继承FIFO队列的所有缺点

2.对低优先级的数据流而言，可能会被“饿死”，因为只有高优先级队列里有数据，PQ就不会服务低优先级队列

3.需要在每一跳上都手工的配置分类

PQ Configuration:

PQ(config)#

PQ(config)#priority-list 1 protocol ip high list 101 //把Acl101定义的数据映射到high优先级队列//

PQ(config)#priority-list 1 interface s0/0 medium  //把来自s0/1口的流量映射到Medium优先级队列//

PQ(config)#priority-list 1 default normal //所有其它流量映射到Normal优先级队列//

PQ(config)#priority-list 1 queue-limit 20 30 40 50 //分别设置高，中，普通，低优先级队列长度//

PQ(config)#

PQ(config)#int s0/1

PQ(config-if)#priority-group 1 //把PQ映射到接口s0/1上//

PQ(config)#

查看：

PQ#

PQ#show queueing int s0/1

Interface Serial0/1 queueing strategy: priority

Output queue utilization (queue/count)

high/13 medium/0 normal/2056 low/0

PQ#

PQ#

PQ#show queueing priority

Current DLCI priority queue configuration:

Current priority queue configuration:

List Queue   Args

1    high protocol ip       list 101

1    medium interface Serial0/0

1    medium limit 30

1    normal limit 40

1    low limit 50

PQ#

QOS-LLQ

Low Latency Queueing（低延迟队列LLQ）

i、特点

在CBWFQ中添加一个优先级队列用于实时的流量。

\* 高优先级队列得到如下保障：

a）低延迟的报文转发

b）带宽

注：在拥塞发生时，高优先级的流量同时受到管制—即它们占用的带宽不能超过它们所保障的带宽。

\* 低优先级队列使用CBWFQ。

ii、配置LLQ

priority 带宽值—-为一个类分配固定的带宽值确保快速转发；若拥塞时，超过该带宽的流量将被丢弃。（若没有拥塞，将不会使用管制）

LLQ基本可以满足企业中融合的网络应用

支持语音对网络的低延迟，抖动小，保障带宽

对其它流量提供公平处理

LLQ= CBWFQ+PQ

LLQ configuration:

LLQ(config)#

LLQ(config)#class-map VOIP

LLQ(config-cmap)#match ip precedence 5

LLQ(config)#

LLQ(config)#class-map cbwfq1

LLQ(config-cmap)#match ip precedence 3 4

LLQ(config)#

LLQ(config)#class-map cbwfq2

LLQ(config-cmap)#match ip precedence 1 2

LLQ(config)#

LLQ(config)#policy-map LLQ

LLQ(config-pmap)#class VOIP

LLQ(config-pmap-c)#priority percent 10 //针对VOIP类流量使用PQ，这路流量在任何情况下都优先发送

LLQ(config-pmap-c)#                   同时最大带宽可以为接口带宽的10%//

LLQ(config-pmap)#class cbwfq1

LLQ(config-pmap-c)#bandwidth percent 30

LLQ(config-pmap-c)#

LLQ(config-pmap)#class cbwfq2

LLQ(config-pmap-c)#bandwidth percent 20 //CBWFQ方式进行调度，分别保障30%和20%接口带宽//

LLQ(config-pmap-c)#

LLQ(config-pmap)#class class-default

LLQ(config-pmap-c)#fair-queue    //剩下其它队列采用缺省的WFQ调度//

LLQ(config-pmap-c)#

LLQ(config)#

LLQ(config)#int s0/0

LLQ(config-if)#service-policy output LLQ

LLQ(config)#