**linux tc实现ip流量限制**

tc是个配置Linux内核流量控制的工具

　　名字

　　tc - 显示／维护流量控制配置

　　摘要

　　tc qdisc [ add | change | replace | link ] dev DEV [ parent qdisc-id | root ] [ handle qdisc-id ] qdisc [ qdisc specific parameters ]

　　tc class [ add | change | replace ] dev DEV parent qdisc-id [ classid class-id ] qdisc [ qdisc specific parameters ]

　　tc filter [ add | change | replace ] dev DEV [ parent qdisc-id | root ] protocol protocol prio priority filtertype [ filtertype specific parameters ] flowid flow-id

　　tc [-s | -d ] qdisc show [ dev DEV ]

　　tc [-s | -d ] class show dev DEV tc filter show dev DEV

　　简介

　　Tc用于Linux内核的流量控制。流量控制包括以下几种方式：

　　SHAPING(限制)

　　当流量被限制，他的传输速率就被控制在某个值以下。限制值能够大大小于有效带宽，这样能够平滑突发数据流量，使网络更为稳定。shaping（限制）只适用于向外的流量。

　　SCHEDULING(调度) bitsCN.Com

　　通过调度数据包的传输，能够在带宽范围内，按照优先级分配带宽。SCHEDULING(调度)也只适于向外的流量。

　　POLICING(策略)

　　SHAPING用于处理向外的流量，而POLICIING(策略)用于处理接收到的数据。

　　DROPPING(丢弃)

　　假如流量超过某个设定的带宽，就丢弃数据包，不管是向内还是向外。

　　流量的处理由三种对象控制，他们是：qdisc(排队规则)、class(类别)和filter(过滤器)。

　　QDISC(排队规则)

　　QDisc(排队规则)是queueing discipline的简写，他是理解流量控制(traffic control)的基础。无论何时，内核假如需要通过某个网络接口发送数据包，他都需要按照为这个接口配置的qdisc(排队规则)把数据包加入队列。然后，内核会尽可能多地从qdisc里面取出数据包，把他们交给网络适配器驱动模块。

　　最简单的QDisc是pfifo他不对进入的数据包做任何的处理，数据包采用先入先出的方式通过队列。但是，他会保存网络接口一时无法处理的数据包。

　　CLASS(类)

　　某些QDisc(排队规则)能够包含一些类别，不同的类别中能够包含更深入的QDisc(排队规则)，通过这些细分的QDisc还能够为进入的队列的数据包排队。通过配置各种类别数据包的离队次序，QDisc能够为配置网络数据流量的优先级。 中国网管论坛

　　FILTER(过滤器)

　　filter(过滤器)用于为数据包分类，决定他们按照何种QDisc进入队列。无论何时数据包进入一个划分子类的类别中，都需要进行分类。分类的方法能够有多种，使用fileter(过滤器)就是其中之一。使用filter(过滤器)分类时，内核会调用附属于这个类(class)的任何过滤器，直到返回一个判决。假如没有判决返回，就作进一步的处理，而处理方式和QDISC有关。

　　需要注意的是，filter(过滤器)是在QDisc内部，他们不能作为主体。

　　CLASSLESS QDisc(不可分类QDisc)

　　无类别QDISC包括：

　　[p|b]fifo

　　使用最简单的qdisc，纯粹的先进先出。只有一个参数：limit，用来配置队列的长度,pfifo是以数据包的个数为单位；bfifo是以字节数为单位。

　　pfifo\_fast

　　在编译内核时，假如打开了高级路由器(Advanced Router)编译选项，pfifo\_fast就是系统的标准QDISC。他的队列包括三个波段(band)。在每个波段里面，使用先进先出规则。而三个波段(band)的优先级也不相同，band 0的优先级最高，band 2的最低。假如band里面有数据包，系统就不会处理band 1里面的数据包，band 1和band 2之间也是相同。数据包是按照服务类型(Type of Service,TOS)被分配多三个波段(band)里面的。 [url]www.bitsCN.com[/url]

　　red

　　red是Random Early Detection(随机早期探测)的简写。假如使用这种QDISC，当带宽的占用接近于规定的带宽时，系统会随机地丢弃一些数据包。他很适合高带宽应用。

　　sfq

　　sfq是Stochastic Fairness Queueing的简写。他按照会话(session--对应于每个TCP连接或UDP流)为流量进行排序，然后循环发送每个会话的数据包。

　　tbf

　　tbf是Token Bucket Filter的简写，适合于把流速降低到某个值。

　　不可分类QDisc的配置

　　假如没有可分类QDisc，不可分类QDisc只能附属于设备的根。他们的用法如下：

　　tc qdisc add dev DEV root QDISC QDISC-PARAMETERS

　　要删除一个不可分类QDisc，需要使用如下命令：

　　tc qdisc del dev DEV root

　　一个网络接口上假如没有配置QDisc，pfifo\_fast就作为缺省的QDisc。

　　CLASSFUL QDISC(分类QDisc)

可分类的QDisc包括：

　　CBQ

　　CBQ是Class Based Queueing(基于类别排队)的缩写。他实现了一个丰富的连接共享类别结构，既有限制(shaping)带宽的能力，也具备带宽优先级管理的能力。带宽限制是通过计算连接的空闲时间完成的。空闲时间的计算标准是数据包离队事件的频率和下层连接(数据链路层)的带宽。

　　[url]www.bitsCN.com[/url]

　　HTB

　　HTB是Hierarchy Token Bucket的缩写。通过在实践基础上的改进，他实现了一个丰富的连接共享类别体系。使用HTB能够很容易地确保每个类别的带宽，虽然他也允许特定的类能够突破带宽上限，占用别的类的带宽。HTB能够通过TBF(Token Bucket Filter)实现带宽限制，也能够划分类别的优先级。

　　PRIO

　　PRIO QDisc不能限制带宽，因为属于不同类别的数据包是顺序离队的。使用PRIO QDisc能够很容易对流量进行优先级管理，只有属于高优先级类别的数据包全部发送完毕，才会发送属于低优先级类别的数据包。为了方便管理，需要使用iptables或ipchains处理数据包的服务类型(Type Of Service,ToS)。

　　操作原理

　　类(Class)组成一个树，每个类都只有一个父类，而一个类能够有多个子类。某些QDisc(例如：CBQ和HTB)允许在运行时动态添加类，而其他的QDisc(例如：PRIO)不允许动态建立类。

　　允许动态添加类的QDisc能够有零个或多个子类，由他们为数据包排队。

　　此外，每个类都有一个叶子QDisc，默认情况下，这个叶子QDisc使用pfifo的方式排队，我们也能够使用其他类型的QDisc代替这个默认的QDisc。而且，这个叶子叶子QDisc有能够分类，但是每个子类只能有一个叶子QDisc。 bitsCN\_com

　　当一个数据包进入一个分类QDisc，他会被归入某个子类。我们能够使用以下三种方式为数据包归类，但是不是任何的QDisc都能够使用这三种方式。

　　tc过滤器(tc filter)

　　假如过滤器附属于一个类，相关的指令就会对他们进行查询。过滤器能够匹配数据包头任何的域，也能够匹配由ipchains或iptables做的标记。

　　服务类型(Type of Service)

　　某些QDisc有基于服务类型（Type of Service,ToS）的内置的规则为数据包分类。

　　skb->priority

　　用户空间的应用程式能够使用SO\_PRIORITY选项在skb->priority域配置一个类的ID。

　　树的每个节点都能够有自己的过滤器，但是高层的过滤器也能够直接用于其子类。

　　假如数据包没有被成功归类，就会被排到这个类的叶子QDisc的队中。相关细节在各个QDisc的手册页中。

　　命名规则

　　任何的QDisc、类和过滤器都有ID。ID能够手工配置，也能够有内核自动分配。

　　ID由一个主序列号和一个从序列号组成，两个数字用一个冒号分开。

　　QDISC

　　一个QDisc会被分配一个主序列号，叫做句柄(handle)，然后把从序列号作为类的命名空间。句柄采用象10:相同的表达方式。习惯上，需要为有子类的QDisc显式地分配一个句柄。

　　www\_bitscn\_com

　　类(CLASS)

　　在同一个QDisc里面的类分享这个QDisc的主序列号，但是每个类都有自己的从序列号，叫做类识别符(classid)。类识别符只和父QDisc有关，和父类无关。类的命名习惯和QDisc的相同。

　　过滤器(FILTER)

　　过滤器的ID有三部分，只有在对过滤器进行散列组织才会用到。详情请参考tc-filters手册页。

　　单位

　　tc命令的任何参数都能够使用浮点数，可能会涉及到以下计数单位。

　　带宽或流速单位：

　　kbps

　　千字节／秒

　　mbps

　　兆字节／秒

　　kbit

　　KBits／秒

　　mbit

　　MBits／秒

　　bps或一个无单位数字

　　字节数／秒

　　数据的数量单位：

　　kb或k

　　千字节

　　mb或m

　　兆字节

　　mbit

　　兆bit

　　kbit

　　千bit

　　b或一个无单位数字

　　字节数

时间的计量单位：

　　s、sec或secs

　　秒

　　ms、msec或msecs

　　分钟

　　us、usec、usecs或一个无单位数字

　　微秒

　　TC命令

　　tc能够使用以下命令对QDisc、类和过滤器进行操作：

　　add

　　bitsCN\_com

　　在一个节点里加入一个QDisc、类或过滤器。添加时，需要传递一个祖先作为参数，传递参数时既能够使用ID也能够直接传递设备的根。假如要建立一个QDisc或过滤器，能够使用句柄(handle)来命名；假如要建立一个类，能够使用类识别符(classid)来命名。

　　remove

　　删除有某个句柄(handle)指定的QDisc，根QDisc(root)也能够删除。被删除QDisc上的任何子类连同附属于各个类的过滤器都会被自动删除。

　　change

　　以替代的方式修改某些条目。除了句柄(handle)和祖先不能修改以外，change命令的语法和add命令相同。换句话说，change命令不能一

　　转载tc文档2-CBQ

　　CBQ

　　名字

　　CBQ - 基于类别排队(Class Based Queueing)

　　摘要

　　tc qdisc ... dev dev (parent classid | root) [ handle major: ] cbq [ allot bytes] avpkt bytes bandwidth rate [ cell bytes] [ ewmalog ] [ mpu bytes ]

　　tc class ... dev dev parent major:[minor] [ classid major:minor] cbq allot bytes [ bandwidth rate ] [ rate rate] prio priority [ weight weight] [ minburst packets] [ maxburst packets ] [ ewma log] [ cell bytes] avpkt bytes[ mpu bytes] [ bounded isolated ] [ split handle& defmapdefmap ] [ estimator interval timeconstant]

　　简介

　　CBQ实现了一个丰富的共享连接的类别层次，既有固定(shaping)带宽的能力，也具有带宽优先级管理的能力。带宽固定是通过计算连接的空闲时间完成的。空闲时间的计算标准是数据包离队事件发生的频率和下层连接(数据链路层)的带宽。

　　带宽固定算法

　　当把一条10mbit/s的连接限制为1mbit/s时，这条连接将有90%的时间是空闲的。如果不是，它就会调低带宽以满足90%空闲时间的限制。

　　在操作过程中，空闲时间还要进行指数平滑移动平均(exponential weighted moving average,EWMA)计算，在这种计算方法中最近经过的数据包权重是前面数据包权重的指数。UNIX系统的平均负载就是使用这种算法计算的。

　　最后，由EWMA计算的值减去空闲时间，所得结果叫做avgidle。最好的情况是avgidle等于0，也就是数据包是严格地按照计算的时间间隔到达。

　　而过载连接的avgidle值是负数，如果这个负数太大，CBQ就会调低带宽，这就造成了过度限制(overlimit)的情况。

　　相反，一个空闲的连接会积累下一个很大的正数avgidle，这样可能造成经过一段空闲后，使带宽失控。为了防止出现这种情况的出现，可以使用maxidle参数限制avgidle的值。

　　如果处于过度限制(overlimit)的情况，CBQ会限制数据包通过的时间间隔严格按照计算的值。不过，由于时钟解析度的原因，这样可能不太合适。见minburst参数。

　　类别划分

　　在一个CBQ实例中，可以存在很多分类。每个类有可以包括其它的QDisc，默认情况下，是pfifo。

　　在为数据包排队时，CBQ作为根，使用不同的方法确定那个类接收数据包。

　　如果没有一些不常见的配置选项，这个过程非常简单。我们在每个节点上查询一个指令，接着按照指令的指示让数据包进入某个类。如果这个类是一个叶子节点，我们就把数据包排到这里；如果这个类还有子节点，我们就重复以上过程。

　　我们在每个节点上重复以下操作，直到数据包被发送到其它节点，或者这个过程由于其它原因中止。

　　查询附属于类上的过滤器，如果过滤器把数据包发到某个叶子节点，处理完毕；否则，开始下一个查询。

　　在defmap中查询这个数据包的优先级，优先级倚赖于TCP头的TOS域。检查这个类是否还继续分类，如果是重新开始下一个查询。

　　向defmap要求获得best effort优先级的指令，检查应答是否还存在分类，如果不是则重新开始下一个查询。

　　如果上述操作都没有返回一个指令，就在这个节点把数据包排队

　　这个算法可以保证数据包最终会有归宿，即使你正在建立流量控制的配置。

　　连接共享算法

　　在向网络设备发送数据包时，CBQ首先要决定发出哪个类的数据包。它对所有的类采用加权轮转(Weighted Round Robin)的方式处理，使每个类的数据包都有机会被送出。WRR处理首先从优先级最高(数字最小的优先级)的类开始处理，直到处理完类中所有的数据包，再接着处理优先级次之的类。

　　每个类可以从其兄弟类借带宽。一个类可以使用bounded声明其带宽不可外借；也可以使用isolated声明不原意外借带宽。

　　QDISC

　　一个CBQ QDisc类的根有如下参数：

　　parent major:minor | root

　　这个命令的参数决定这个CBQ实例所在的位置，或者是在一个网络接口的根(root)，或者位于一个现有的类里面。

　　handle major:

　　和其它的QDisc一样，CBQ QDisc也可以指定一个句柄。句柄只能包含主识别号(major number)和一个冒号。数字的选择是随意的，不过如果在这个QDisc里面会继续分类，这个数字非常有用。

allot bytes

　　这个参数用于分配的带宽，决定数据包传输的时间表。QDisc的allot参数和类的allot参数略有不同。数字bytes是任意的。缺省是一个根据avpkt得到的数字。

　　avpkt bytes

　　数据包的平均大小，它用于计算最大空闲时间(maxidle)，也用于确认allot参数的值是安全的。

　　bandwidth rate

　　为了决定空闲时间(idle time)，CBQ必须知道底层物理接口的带宽，或者父QDisc的带宽。这是一个极为重要的参数，下面会继续讨论。

　　cell time

　　time的值决定进行数据包传输时间计算的时间间隔。使用缺省值是比较明智的。

　　mpu bytes

　　一个大小为0的数据包仍然会消耗时间来传输。用这个参数设置小于bytes个字节的数据包在进行传输时间的计算忽略不计。缺省值是0。

　　ewma log

　　在CBQ需要测量平均空闲时间时，它会使用加权指数移动平均算法来平滑测量的值，得到一个移动平均值。log决定平滑因子的大小。这个数值介于0到31之间，数值越小敏感度越大。默认值是5。

　　一个CBQ QDisc只需要知道底层连接的大小。实际的带宽限制由其子类完成。

　　分类

　　有很多参数可以用于分类的配置操作：

　　parent major:minor

　　确定父QDisc的位置，把这个类加入到树状结构中。如果它是直接附属于一个QDisc，而且这个QDisc没有其它的类，minor可以被忽略。这个参数是必须的。

　　classid major:minor

　　和QDisc一样，类也可以命名。主识别号(major number)必须等于其父QDisc的主识别号。这个参数是可选的，只有它还需要再细分才需要。

　　weight weight

　　在从队列中取出数据包通过网络接口向外发送时，CBQ会采用轮转的方式轮流从队列中取出属于不同分类的数据包。weight设置每个类的权重。权重越高，在每个循环CBQ取出的数据包也就越多。一个类中所有的权重都会被换算成与rate参数设定值的百分比。

　　allot bytes

　　这个参数设置每个循环可以有多少个字节出队。这个值最小为avpkt的2/3。这个参数是必须的。

　　prio priority

　　设置类的优先级，在轮转过程中，priority数字较小(优先级高)的类的数据包优先出队。这个参数也是必须的。

　　avpkt bytes

　　参见QDisc中的相关介绍。

　　rate rate

　　设置这个类可以达到的最大速率，以及所有子类速率总和可以达到的值。这个参数是必需的。

　　bandwidth rate

　　这个参数不同于建立CBQ时的bandwidth参数。只有在决定maxidle和offtime时才有用，在设置maxburst或者minburst时，CBQ会使用maxidle和offtime进行计算。如果需要设置最大突发(006Daxburst)或者最小突发(miniburst)，这个参数是必需的。

　　maxburst

　　这个参数设置的数字用于计算maxidle，以便avgidle的值等于maxidle时，在avgidle达到0之前，允许设定的数据包突发性地通过。这个数字越大，对于突发流量的适应性越好。你不能直接设置maxidle，只能通过这个参数来设置。

　　minburst

　　上面讲过，在过度限制(overlimit)情况下，CBQ需要调低带宽。为了避免这种情况的出现，比较理想的解决方案是精确地空闲某个时间，然后放行一个数据包。然而，对于UNIX内核来说，很难对时间间隔小于10ms的事件进行调度，因此只好把等待时间放长，接着突发性地放行多个数据包。

　　上面所说的等待时间叫做offtime。minburst的值越高，在一个较长时间内进行的带宽限制越准确，但是也会导致更大的突发流量。这个参数是可选的。

　　minidle

　　如果avgidle小于0，需要等待avgidle增加到一个足够大的值才能发送一个数据包。为了避免在一个长期处于关闭状态下的连接出现突发流量，如果avgidle太低就会被复位为minidle参数设置的数值。

　　minidle的单位是负微秒，10表示avgidle不能低于-10微秒。这个参数是可选的。

　　bounded

　　表示这个类的带宽概不外借。

isolated

　　表示这个类的带宽不原意外借。

　　split major:minor & defmap bitmap[/bitmap]

　　如果附属于这个类的过滤器不能判断数据包所属类别，CBQ也可以根据数据包的优先级为它们分类。优先级共有8个，范围是0到7。

　　defmap设置这个类接受具备哪些优先级的数据包，接受的优先级使用bitmap来计算，CBQ用bitmap和数据包的优先级域进行and计算。最低有效位(Least Significant Bit)对应优先级0。split告诉CBQ需要做出决定的类，参数应该是其父类。

　　例如：tc class add ... classid 10:1 cbq .. split 10:0 defmap c0，这条命令告诉类10:0把优先级为6和7的数据包都送到子类10:1。

　　然后最好使用tc class add ... classid 10:2 cbq ... split 10:0 defmap 3f命令决定其它优先级数据包的流向，把优先级为0、1、2、3、4和5的数据包都送给10:2。

　　C0的二进制是11000000，3F的二进制是001111111，因此这两个defmap可以匹配数据包优先级域的所有位。C0匹配6和7，对应的是interactive和control位；而3F匹配余下的位。

　　estimator interval timeconstant

　　CBQ能够计量每个类使用了多少带宽，tc过滤器能够把数据包分类。CBQ使用一种简单的估算方式计算每interval微秒通过多少流量，判断自己使用的带宽。另外，还需要进行指数平滑移动平均的计算，时间常数由timeconstant设置。它决定对短期突发流量平均值的敏感程度，这个值越高敏感度越低。

　　BUGS

　　底层链路的带宽可能是无法预知的。例如，PPoE或者PPTP连接实际上是一个逻辑的通道，而不是一个物理设备。CBQ对于主要的带宽配置错误有很大的弹性，代价可能就是带宽固定(shaping)精度降低。

　　默认情况下，内核依靠粗糙的计时信息进行带宽固定。在一个较长的时间段内，可以维持很好的精度，但是在以秒为单位进行计量的时间段内，其结果就不准确了。

　　参考tc-cbq-details(8)(这个文档至今尚未完成)改进这个问题。

　　我经过测试后，以下代码完全可以限制下载速度，如下：

　　由于tc对发出包限制，我这里是 LAN:eth0 INT:eth1,限制网内下载速度

　　# tc qdisc add dev eth0 root handle 10:0 cbq bandwidth 5Mbit avpkt 1000

　　创建队列规则，添加设备eth0，root表示这是根（root）规定,其句柄 （handle）设定为 10:0'。 其类型为 CBQ。公司带宽为 5 M，平均包大小为 1000 字节。

　　下面生成根类（root class）：

　　# tc class add dev eth0 parent 10:0 classid 10:1 cbq bandwidth 5Mbit rate 5Mbit allot 1500 weight 1M prio 8 maxburst 20 avpkt 1000

　　各参数见前文档，allot 1500 我的网卡MTU值

　　下面生成支类

　　# tc class add dev eth0 parent 10:1 classid 10:10 cbq bandwidth 5Mbit rate 1Mbit allot 1500 weight 5Kbit prio 5 maxburst 20 avpkt 1000 bounded

　　# tc class add dev eth0 parent 10:1 classid 10:20 cbq bandwidth 5Mbit rate 1Mbit allot 1500 weight 100Kbit prio 5 maxburst 20 avpkt 1000 bounded

　　# tc class add dev eth0 parent 10:1 classid 10:30 cbq bandwidth 5Mbit rate 3Mbit allot 1500 weight 900Kbit prio 5 maxburst 20 avpkt 1000 bounded

　　我们已经向内核通知了我们的类，我们还需要告诉内核如何管理队列

　　# tc qdisc add dev eth0 parent 10:10 sfq quantum 1500b perturb 15

　　# tc qdisc add dev eth0 parent 10:20 sfq quantum 1500b perturb 15

　　# tc qdisc add dev eth0 parent 10:30 sfq quantum 1500b perturb 15

　　perturb：表示时间15sec

　　告诉内核网络包和类的映射关系

　　# tc filter add dev eth0 parent 10:0 protocol ip prio 10 u32 match ip dst 192.168.1.226 flowid 10:10

　　# tc filter add dev eth0 parent 10:0 protocol ip prio 10 u32 match ip dst 192.168.1.227 flowid 10:20

　　# tc filter add dev eth0 parent 10:0 protocol ip prio 10 u32 match ip dst 192.168.1.228 flowid 10:30

　　上传速度限制类似与下载

　　不同处是 下载使用内网接口，上传使用外网接口，上传对源地址使用src参数。