QoS的区分服务体系模型及其在Linux中的实现

The System Framework of DiffServ Model of QoS And its Realization in Linux

摘 要：本文在介绍QoS技术中区分服务模型的工作原理和体系结构的基础上，重点讨论Linux操作系统对QoS的区分服务的支持。最后通过一个实际例子展示了在Linux操作系统中区分服务的实现过程。

关键词：QoS 区分服务 Linux

田 涛 （中国科学院研究生院 100039）
鲁士文 （中国科学院计算技术研究所 100080）

1 引言

随着网络多媒体业务的飞速发展，Internet上多媒体应用层出不穷，多媒体信息的数量在与日俱增。Internet已经从单一的数字传送向数字、语音、图像等多媒体信息的综合传输网络演化。不同类型的业务对服务质量的要求是不一样的。但是Internet现有的传输模式仍然是IP协议提供的无差别、尽力而为（Best-Effort）的服务。由于无法提供区分、保证的服务，致使庞大的数据流传输时造成带宽不足及数据包的延时甚至丢弃。因此，探讨如何在有限的带宽中让这些数据包传输而不受网络拥塞的影响成为下一代Internet协议研究的重点课题。本文主要探讨在服务质量保证（QoS）技术中差分型服务及其在防火墙中的应用。

2 QoS的定义

所谓的服务质量QoS，是指服务性能的聚集效应，它决定用户对特定服务的满意程度。从网络角度来说，就是通过控制一些具体的、可量化和度量的参数来提高服务质量，如传输延时、抖动、丢失率、带宽要求、吞吐量等指标。同时，服务质量不单单是网络的事情，而且是应用程序、用户终端、网络、服务器各部分的综合效应。在端到端的路径上，任何一个环节不符合QoS的要求，都会对服务质量产生影响。

3 QoS的模型

在传统的IP中，只有一种服务类型，就是尽力而为（best-effort）的服务模型。是指网络会尽最大的努力将数据包传送到目的端，但它不为任何数据流提供保证或为其事先分配资源。

现在有关网络服务质量（QoS）的研究主要有两类：综合服务模型（Integrated service，IntServ）和区分服务模型(Differentiated service，DiffServ)。

3.1 综合服务模型

综合服务的体系结构和参考框架见文献[1]。其特点在于带宽的预留。它的基本思想是对于每一个需要进行QoS处理的数据流，通过一定的信令机制，在其经过的每一个路由器上根据业务的QoS需求进行网络资源预留，从而为该数据流提供端到端的QoS保证。

资源预留协议RSVP[2]是集成服务的核心。这是一种信令协议，用来通知网络节点预留资源。如果资源预留失败，RSVP协议会向主机发回拒绝消息。

综合服务能够在IP网上提供端到端的QoS保证。但是，综合服务对路由器的要求很高，当网络中的数据流数量很大时，路由器的存储和处理能力会遇到很大的压力。如果网络拓扑结构发生了变化，就要重新考虑预留。因此，综合服务可扩展性很差，难以在Internet核心网络实施，目前业界普遍认为综合服务主要应用于较小的网络中。

3.2 区分服务模型

为了解决骨干网络上的QoS问题，提出了区分服务（Diff-serv）模型。它与综合服务模型的本质区别在于不是针对于每一个业务流进行网络资源的分配和QoS参数的配置，而是将具有相似要求的一组业务归为一类，对这一类业务采取一致的处理方式。

3.2.1 工作原理

区分服务的基本思想是将用户的数据流按照服务质量要求来划分等级，任何用户的数据流都可以自由进入网络，但是当网络出现拥塞时，级别高的数据流在排队和占用资源时比级别低的数据流有更高的优先权。区分服务只承诺相对的服务质量，而不对任何用户承诺具体的服务质量指标。IETF定义了区分服务的体系结构[3] [4]。

区分服务的体系结构基于这样一个模型之上，其主要机制是流量调节和基于PHB（每一跳行为）[4]的转发。图1表示了区分服务的分组处理过程和内部组成元素所执行的功能。业务量通过边缘路由器进入网络，进入网络的业务量在网络边缘处进行分类和可能的调节，然后被分配到不同的集合中去。每一个行为集合由唯一的DS编码点标识。在网络核心处，数据包根据DS编码点对应的PHB（每一跳行为）进行转发。这样，对分组进行的复杂处理被推到了网络的边缘，核心网的主要任务只是对其采取相应的转发措施。在区分服务模型中由于不需要网络中间结点管理每个正在工作的流状态，因此具有良好的可缩放性。

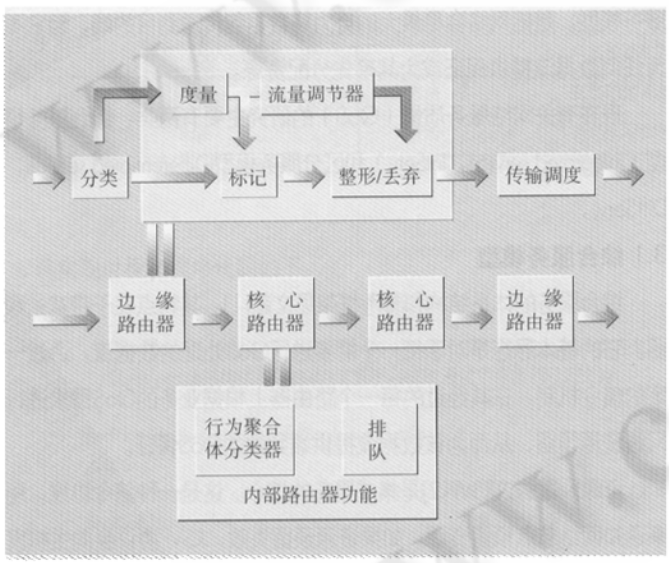

图 1　区分服务模型原理图

3.2.2 组成部件

（1）DS字段。DS字段是用来取代先前已经存在的IPV4中的TOS字节和IPV6中Traffic Class Octet的一种IP标志位。它指示了分组在网络转发时应接受的服务，即为PHB（每一跳行为）。DS字段的结构如图2所示：

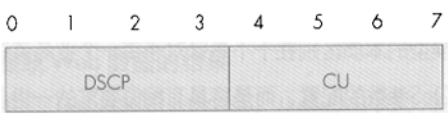

图 2　DS 域的结构

DSCP：区分服务码值，即DS标记值。

CU：未定义。

（2）DS边缘结点 DSCP。边缘结点工作在DS域的边缘或连接两个DS域。边界结点对进入DS域的业务流进行分类，使穿过DS域的分组被标记并选择一个PHB。分类器的配置（检查分组首部的那些字段）以及递送服务所必需的其它功能（如丢弃不符合令牌桶过滤器的分组）取决于网络管理员的配置策略。边界结点的功能可在路由器、防火墙或主机中实现。根据业务流通过DS域的方向可将边界结点分为入口结点（Ingress Node）和出口结点(Egress Node).

边缘路由器由多字节分类器（MC）、策略器、分组标记器、队列管理 / 调度器组成。

① MC可以检查分组中的一个或多个字段，包括IP头和传输层的头等，甚至包括净荷；

② 策略器可以对分组进行整形或丢弃，使其和描述所传输流量服务的文件相符合；

③ 分组标记器标记DS字节中的一个或多个比特，使这些分组与某个特定类或描述文件相关；

④ 队列管理 / 调度器管理队列的长度，调度分组的发送。

3.2.3 内部结点

DS域内部结点可以是核心交换机或路由器，它们可以根据DS字节的内容提供每一跳的服务。内部结点通常采用队列管理和调度原理来控制队列的深度（如随机早期检测RED），并在每个服务基础上采用一定策略对出口的发送过程进行调度（如加权公平排队WFQ、基于类的排队或优先权排队等）。

核心路由器的功能较少，也较简单，即：

（1）DS字节分类器检查IP头内的DS字节；

（2）队列管理 / 调度器与边缘路由器中的相同。

3.2.4 策略管理

对服务策略进行管理，如在网络和终端用户之间指定或强制某些服务级别的协定，使得某些用户在受到更好的服务的同时，也需要花费更多的金钱。

内部结点和边缘结点共同构成了DS域。这些结点按照通用的服务配置策略进行工作且在每个结点上实现了多个PHB组。DS域通常包括一个统一管理机构下的一个或多个网络。连续的一个或多个区分服务域构成区分服务区域（DS Region）。DS区域扩展了区分服务的范围。

4 QoS 在 Linux 中的实现

Linux内核网络协议栈从2.2.x开始，为了支持QoS，在发送数据包的代码中加入了支持模块，具体的代码位于net/sched/目录。在Linux里面，对这个功能模块的称呼是Traffic Control ,简称TC。它可以对数

据包进行分类、管理、检测拥塞和处理拥塞。

TC 包括三个基本的构成块：队列（queuing discipline）、类（class）和分类器（Classifiers）。队列可以看作设备的流量/数据包管理器。在Linux内核中支持很多队列，如Class Based Queue（类基队列），FIFO Queuing（先入先出队列），PQ（优先队列），CQ（定制队列），WFQ（加权公平队列），等等。这些队列管理机制都可以绑定到类上。以下我们讨论的队列与分类都是基于CBQ(Class Based Queue)的。

类由设备队列来管理。类由若干规则（rule）构成，这些规则用以管理类所拥有的数据。一般来说，类队列规定管理该类的数据和队列，决定延迟、丢掉或者重新分类它管理的包。

分类器用来描述包，并且把它们映射到队列规定所管理的类。分类器通常提供简单的描述语言，指定选择包、把包映射到类的方法。目前，TC可以使用的分类器有：fwmark分类器，u32分类器，基于路由的分类器和RSVP分类器（分别用于IPV6、IPV4）等；其中，fwmark分类器允许我们使用 Linux netfilter 代码选择流量，而u32分类器允许我们选择基于 ANY 头的流量。所有的防火墙都有分类器，例如，ipchains，都能用来分类包。

图3是一个数据发送队列管理机制的模型图。其中的QoS策略可以是各种不同的拥塞处理机制。我们可以把这一种策略看成是一个类，称为策略类。在实现中，这个类有很多的实例对象，即策略对象。使用者可以分别采用不同的对象来管理数据包。策略类有很多的方法，如入队列（en-queue），出队列（de-queue），重新入队列(re-queue)，初始化(init)，撤销(destroy)等。

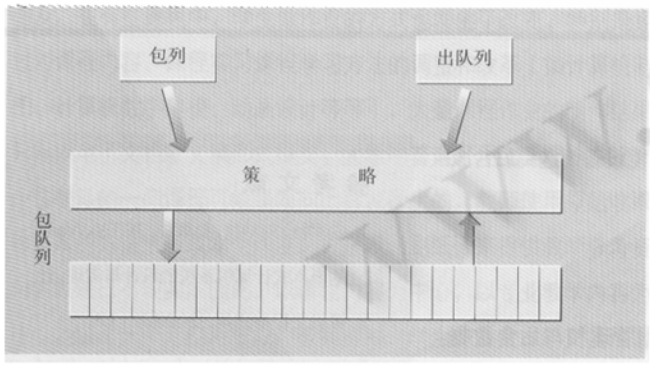

图3　QoS策略处理数据包

5 具体实例

配置和使用流量控制器TC，主要分以下几个方面：分别为建立队列、建立分类、建立过滤器和建立路由，另外还需要对现有的队列、分类、过滤器和路由进行监视。基本使用步骤为：

（1）针对网络物理设备绑定一个CBQ队列；

（2）在该队列上建立分类；

（3）为每一分类建立一个基于路由的过滤器；

（4）最后与过滤器相配合，建立特定的路由表。

下面以一个具体例子来解释LINUX中对QoS的实现过程。假设一个简单的环境：

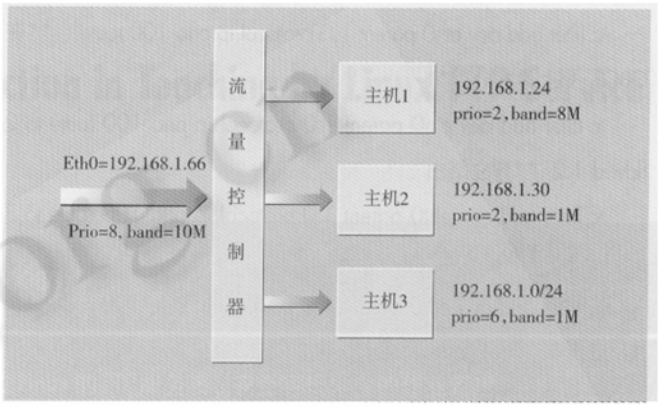

图4

其中，流量控制器上的以太网卡(eth0)的IP地址为192.168.1.66，在其上建立一个CBQ队列。假设包的平均大小为1000字节，包间隔发送单元的大小为8字节，可接收冲突的发送最长包数目为20字节。

假设有三种类型的流量需要控制：

（1）是发往主机1的，其IP地址为192.168.1.24。其流量带宽控制在8Mbit，优先级为2；

（2）是发往主机2的，其IP地址为192.168.1.30。其流量带宽控制在1Mbit，优先级为2；

（3）是发往子网1的，其子网号为192.168.1.0，子网掩码为255.255.255.0。流量带宽控制在1Mbit，优先级为6。

* * 建立队列(在eht0上绑定一个CBQ队列)

tc qdisc add dev eth0 root handle 1: cbq bandwidth 10Mbit avpkt 1000 cell 8 mpu 64

* * 建立分类

tc class add dev eth0 parent 1:0 classid 1:1 cbq bandwidth 10Mbit rate 10Mbit maxburst 20 allot 1514 prio 8 avpkt 1000 cell 8 weight 1Mbit

tc class add dev eth0 parent 1:1 classid 1:2 cbq bandwidth 10Mbit rate 8Mbit maxburst 20 allot 1514 prio 2 avpkt 1000 cell 8 weight 800kbit split 1:0 bounded

tc class add dev eth0 parent 1:1 classid 1:3 cbq bandwidth 10Mbit rate 1Mbit maxburst 20 allot 1514 prio 1 avpkt 1000 cell 8 weight

100kbit split 1:0

tc class add dev eth0 parent 1:1 classid 1:4 cbq bandwidth 10Mbit rate 1Mbit maxburst 20 allot 1514 prio 6 avpkt 1000 cell 8 weight 100kbit split 1:0

* * 建立过滤器：过滤器服务于分类，一般只针对根分类提供一个过滤器，然后为每个子分类提供路由映射。

tc filter add dev eth0 parent 1:0 protocol ip prio 100 route **在CBQ队列上建立分类**

tc filter add dev eth0 parent 1:0 protocol ip prio 100 route to 2 fldwid 1:2 **建立路由

tc filter add dev eht0 parent 1:0 protocol ip prio 100 route to 3 fldwid 1:3 映射分类**

tc filter add dev eth0 parent 1:0 protocol ip prio 100 route to 4 fldwid 1:4

* * 建立路由：与前面建立的路由映射一一对应。

ip route add 192.168.1.24 dev eth0 via 192.168.1.66 realm 2

ip route add 192.168.1.30 dev eth0 via 192.168.1.66 realm 3

ip route add 192.168.1.0/24 dev eth0 via 192.168.1.66 realm 4

经过上述步骤，就能够对通过网卡eth0的流量进行控制和管理。同时还应该对现有的队列、分类、过滤器和路由状况进行监视和维护。由于篇幅关系就不详细解释了。

6 结束语

随着互联网的深入发展，网络新业务的层出不穷，使得QoS技术成为当今的一个热门技术。几乎所有的高端网络设备都支持QoS功能。目前，Linux所提供的QoS是所有操作系统中比较复杂和完善的。以上是作者在实际工作中的一点体会和初步的见解，还有更深入的工作等待以后进一步的研究和探讨。

参考文献

1 Braden, R., Clark, D. and S. Shenker, "Integrated Services in the Internet Architecture: An Overview", RFC1633, July 1994.

2 RFC 2205 Resource ReSerVation Protocol (RSVP) – Version 1

3 Nichols, K., Blake, S., Baker, F. and D. Black, "Definition of the Differentiated Services Field (DSField) in the IPv4 and IPv6 Headers", RFC 2474, December 1998.

4 Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z. and W. Weiss, "An Architecture for Differentiated Services", RFC 2475, December 1998.

5 Weighted Fair Queueing (WFQ), http://www.cisco.com/warp/public/732/tech/wfq

6 祝顺民，Linux 2.4.x 网络协议栈 QoS 模块(TC)的设计与实现。http://www-900.ibm.com/developerWorks/cn/linux/kernel/hqos/index.shtml