面向流量控制的可扩展策略库的设计与实现*)

刘潇潇[1] 张大方[1] 谢 鲲[2] 姜玉蓉[1]
(湖南大学软件学院[1] 计算机与通信学院[2] 长沙 410082)

摘 要 Linux提供了一套功能强大的流量控制机制,成为当前实现网络QoS的重要工具。针对制定流量控制方案时带宽划分精度和粒度难以权衡,采用单一队列调度算法实施流量整形效果不尽如人意等问题,提出了基于可扩展策略库的流量控制管理方案,在Linux下设计并实现了可扩展流量控制策略库。介绍了该策略库的3个模块:策略存储实体、出错回滚监控模块和可扩展策略库管理模块。定义了策略的存储格式,将主机、服务、时间、上下行方向等元素综合管理起来。同时提出了结合优先级调度的策略添加、删除、合并的策略管理算法。通过实验分析证明该流量控制管理方案的有效性和实用性。

关键词 流量控制,流量整形,可扩展策略库,Linux

Design and Implementation of Scalable Policy Sets for Traffic Control

LIU Xiao-xiao[1] ZHANG Da-fang[1] XIE Kun[2] JIANG Yu-rong[1]
(College of Software Engineering[1], College of Computer and Communication[2], Hunan University, Changsha 410082, China)

Abstract Linux traffic control mechanism has become an important tool in assuring QoS of nowaday network. To balance precision and granularity of bandwidth division, improve unsatisfied throughput caused by using a single queue dispatching algorithm, this paper brought forward a method—scalable policy sets based traffic control management. Three highly modularized components of the policy sets were presented: policy entity, management module and exception handling module. Policy storage format was defined. Elements including host, service, time and up/down direction were integrated, and the policy management algorithm was provided. In the end, validity and practicability of the method are validated by experiment.

Keywords Traffic control, Traffic shaping, Scalable policy sets, Linux

1 引言

近年来,随着大规模存储和分布式系统的成熟,互联网上传输的不再只是简单的文本和图像。多媒体通信、能够集成文本、视频、音频、图像为一体,得到越来越广泛的应用。眼下,随着技术的发展,众多新型业务已经渗透到网络的各种主流应用之中。然而这也使得一些无关紧要的应用(如下载、在线电影、游戏等)都来与一些关键性应用争夺有限的广域网资源,使得关键性应用无法高效率运行。正是基于这种"劣币驱逐良币"日趋严峻的现实,对IP流量来说,"不了解的事物不会对自己构成伤害"的原理不再适用,对网络实施流量控制的呼声也越来越高。

网络流量控制器的主要工作在于通过监视网络流量,识别出重要和关键的业务,并根据各种应用和服务进行网络分类,增加关键应用的有效吞吐量;同时找出非法的应用予以阻止,从而提高网络利用率。目前,流量控制中许多研究工作将重点放在了对Linux流量整形算法的研究实现,及在其已有技术基础上的深入改进(代表性算法研究包括:HTB[1,2],CBQ[3,4],SFQ[5]等)和利用Netfilter/Iptables模块实现简单的流量控制功能等领域上。而对于怎样有效管理和控制网络流量方面的研究探讨却还较少。特别是随着当今业务的多元化趋势,如何针对网络运行状况,组建一个行之有效的流量控制管理机制,合理组织分配有限的网络资源,优化网络应用流量,提高关键业务的QoS,已成为当下之需。

本文在对Linux流量控制机制研究的基础上,提出一种基于可扩展策略库的流量控制管理方案。该方案构建了一个可扩展策略库,使得用户可通过定制策略将网络中的主机、服务、时间及上下行方向等元素进行有机管理,在不损害整体流控系统性能、保证关键业务畅通运行的情况下,有效地提高了网络的可管理性。同时,本文通过实验分析证明其有效性和实用性。

2 Linux流量控制机制

Linux从2.2版起就提供了一套灵活、功能强大的流量控制机制。当网络节点发送数据包时,它负责将等待发送的数据包按照一定的规则分类、排队以及调度发送出去。Linux流量控制框架如图1所示。

由图1可知,Linux流量控制分为两部分:用户空间和内核空间。内核空间存放并运行着提供特定队列操作的核心代码,而用户空间存放并运行用于人机交互的用户程序。

*)国防基础科研"十一五"项目(A1420060162),湖南省科技计划资助项目(2006GK3101)。**刘潇潇** 系统分析师,硕士研究生,主要研究领域为网络测试、软件测试;**张大方** 博士,教授,博士生导师,CCF高级会员,主要研究领域为可信系统与网络、容错计算;**谢 鲲** 博士,主要研究领域为可信系统与网络、网络测试;**姜玉蓉** 硕士研究生,主要研究领域为软件测试、网络测试。

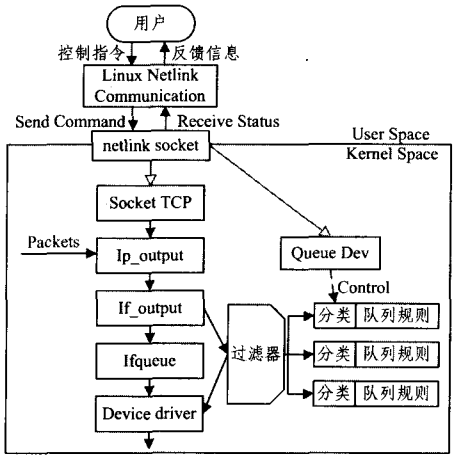


图 1　Linux 流量控制框架图

2.1　Linux 内核空间流量控制元素

Linux 内核模块将不同标签的数据包进行区分，送往不同队列进行调度整形，实现区分服务。而实现这一系列功能的元素(除去 Policies)大体上可分为如下 3 大类[6]：

- 分类(classes)
- 队列规则(queuing disciplines)
- 过滤器(filters)

其中分类机制是流量控制的一种重要手段，它通过建立树型结构的层次框架，概念上将网络设备的缓存队列分为多块进行区分管理，使得流量控制更为灵活。每个分类中都包含一个与之相关的队列规则，用以控制进入该分类的数据包发送的方式。而分类下面又可以派生出子分类，子分类分享了父类的部分带宽，但因其包含的队列规则不同，调度整形的效果也就不一样。实际上，分类和队列规则是紧密联系在一起的，不同队列之间是没有联系的，分类将多个队列组织到一起，起到了粘合剂的作用。

在实施流量控制过程中，需要根据数据包标签，将不同的包发往合适的分类，这一过程就叫做包分类。在 Linux 流量控制模块中，包分类是通过过滤器完成的。一个过滤器包含若干匹配项，如果符合匹配项，就按此过滤器分类。

2.2　用户空间和内核空间的通讯

Linux 出于安全因素，规定用户空间进程不能直接访问内核空间以及其他用户空间进程地址，内核空间同样也不能直接访问用户空间地址。当数据需要在用户空间和内核空间交互时，不能直接读取该数据，必须用一种特殊的通讯套接字 netlink[7]机制来传递数据，以保护内核的安全和稳定。在 Linux 实现中，netlink 其实就是一组宏，这组宏通过创建和访问 netlink 数据报，给用户进程和内核模块提供双向的通信连接。用户空间的接口通过 rth→fd = socket(AF_NETLINK, SOCK_RAW, protocol)创建一个类型为 AF_NETLINK 的 socket，然后使用这个 socket 与内核模块进行通信。内核模块需要使用专门的内核 API 来使用 netlink，API 位于文件〈net/core/af_netlink. c〉中。

Linux 提供的这种流量控制机制方便了对网络流量的控制，特别是分类队列规则可以使我们按各种需要灵活地对要控制的网络资源进行区分管理。然而正因为它的灵活，使我们在制定一个相对较大网络的流量控制方案时难免出现混乱。尤其是在分层、分类时，若处理不当更容易出现这种情况，从而降低带宽管理的简洁性和易用性，增加其复杂度。为此，本文提供一种已经实现的，并在应用过程中得到实践验证有效的流量管理方案。

3　可扩展策略库总体设计

可扩展策略库是在 Linux2. 6. 18 用户空间上开发的，其整体架构如图 2。对比图 1，可扩展策略库是在用户和 netlink 通讯模块之间抽象出来的一层。一方面，它接收用户为保护或限制某业务发送的期望业务流信息，为其指定相应的策略，实现流控管理；另一方面，它通过 netlink 通讯机制向内核空间发送控制命令，实施流量控制。为保证策略库具有良好的可扩展性和可维护性，整个系统架构设计遵循分层和构件化的设计理念，分为 3 个模块：存储实体、出错回滚监控模块和可扩展策略库管理模块。

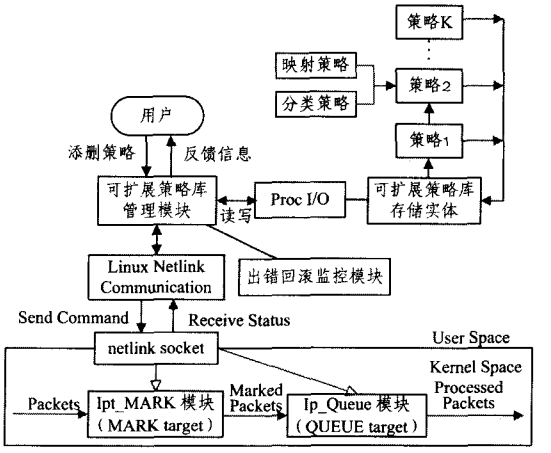


图 2　可扩展策略库整体结构图

3.1　可扩展策略库存储实体

存储实体模块主要负责处理策略数据的表现形式，它包含用于各模块间传递信息的数据集，是整个策略库的基础。

根据用途可将策略分为映射策略和分类策略两种。映射策略用于存储业务与分类之间的对应关系。而分类策略根据用户期望的业务流信息分析结果，每个策略实体将数据业务流中的主机、服务、时间、上下行方向和期望带宽等信息模型化，并根据配置的规则属性建立不同的策略实体。

3.2　出错回滚监控模块

在可扩展控制策略库中，加入出错处理——出错回滚监控模块，以保证整个策略库良好的安全性和容错性。

在添加映射策略中，添加映射策略之前必须先保证对应的分类存在。然而在这个过程中，如果添加分类策略成功返回，而在添加映射策略时，系统死机或出现其他错误导致无法添加映射策略，相应的这类数据包将无法到达指定分类，从而无法成功实现流量整形。为了避免出现这类错误，出错回滚监控模块如检测到发生上述错误，执行回滚操作，重新添加映射策略，或删除之前添加的分类策略。

删除分类策略时，同样存在类似限制，先删除与之相关的映射策略，然后才能删除分类策略。若删除映射策略成功，而删除分类策略时出错，出错回滚监控模块执行相应的回滚操作，保证内核流控状态和策略库存储的一致性。

3.3　可扩展策略库管理模块

可扩展策略库管理模块是整个策略库的核心模块，其主要职能是建立策略实体与 Linux 内核流控状态的对应关系，并规范策略定制的逻辑过程，同时通过 netlink 通讯机制向内

核发送控制指令。当用户期望的业务流信息进入管理模块时，策略管理算法逻辑会针对业务流属性特征识别出重要和关键的业务，结合已有的业务流进行资源划分，并对其配置合适的队列调度算法实施流量整形，合理利用网络资源，提高网络 QoS；同时找出非法的应用予以阻止，从而确保重要关键业务的性能，增加关键应用的有效吞吐量，提高网络利用率。

4 可扩展策略库策略管理算法

可扩展策略库管理模块最核心的部分就是策略管理算法。其中映射策略存储较为简单，本文不做介绍。下面主要讨论分类策略的管理算法。算法的基本思想是根据用户期望的业务流信息，将一定范围内不同的业务配置到不同的规则，然后将不同的规则划分到与之对应的策略中。

可扩展策略库分类策略管理算法中涉及的部分类结构如图 3 所示。

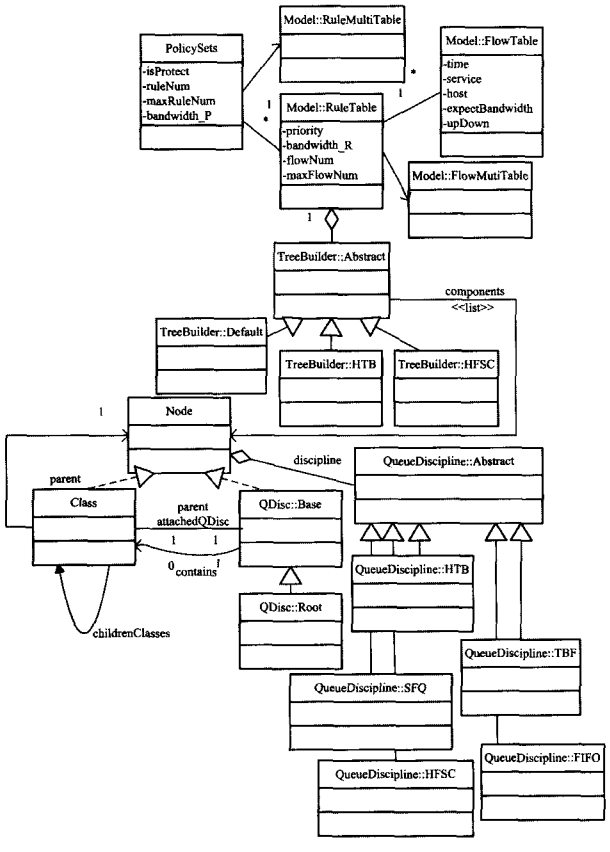


图 3 分类策略管理算法涉及的类图

其中，PolicySets 用于存储策略信息，RuleTable 用于存储规则信息，FlowTable 存储用户期望的业务流信息。PolicySets 同 RuleTable 一对多关联，表明一个策略下可包含多个规则。同样，RuleTable 和 FlowTable 一对多的关系也指出一个规则下包含了一个或多个流信息。RuleTable 还以聚合的方式，通过 TreeBuilder 与 Node 相关联，Node 中存储了业务集归属的分类信息和为该规则下的业务集指定的内核队列调度算法。通过这种分层的思想，将流量控制信息分别以策略、规则和业务流三层进行存储，修改任何一层属性都不会造成其他的变动，从而保证了流控管理的灵活性。而且这种分层存储的架构使得一个策略下挂载的流信息理论上可以无限扩展。

4.1 添加、删除策略

添加策略时，在策略下挂载规则，并根据将要挂载到下的业务流信息，设置操作相关属性字段。若当前策略下的规则无法挂载新增业务信息，遍历所有策略。如能找到一匹配条件的策略，根据其下操作，设置相关属性字段，否则采取策略合并机制。

策略合并原则：当无法根据条件挂载新增业务信息，遍历所有受限策略中的规则，通过最佳匹配算法将新来的业务信息挂载到可提供和该业务期望带宽最相近带宽的规则中去。

删除策略时，取消对业务的流控，受限策略回收释放的差额带宽，并重新设置挂载该业务流信息的规则属性字段。

添加删除策略逻辑会针对业务流信息识别出关键业务，结合已存储的策略库信息进行资源划分，并对业务流配置合适的队列调度算法实施流量整形，保证关键业务的畅通运行；同时找出非法的应用予以阻止，合理利用网络带宽，实现网络利用最大化。

4.2 优先级调度

后期受限规则下业务可能无限扩展，导致所有受限规则可用带宽趋于零，此时最佳匹配算法已无法起到作用。为了避免所有后来的业务都聚集到一个规则中，影响队列调度的公平性，后期引入了优先级的概念。初始时，设置所有受限规则优先级；最佳匹配算法失效时，启动优先级调度模块，更改规则优先级。

启动优先级调度后，新来的业务流信息通过优先级设置，选取合适的规则进行挂载。每次挂载完新到业务流信息后，都须重新计算规则优先级。

考虑到在实际网络状况中，需要保护的关键业务流只会是少数，为保证整体流量控制的高效性（策略越多，内核在执行匹配过滤操作所花费的时间也就越多，因而流控整体性能就会下降），受保护策略下规则最多只能挂载一个业务流信息。而需要予以管制的非关键业务，尤其是影响网络利用率的有害流量种类繁多，为确保一一限制，受限策略下规则可无限挂载业务流信息。但基于性能的考虑，算法限定同时支持流量控制的业务数只能到 300。整个算法具有良好的扩展性，同时将流量控制信息分别以策略、规则和业务流三层进行存储，保证了管理的灵活性。

5 实验验证与分析

本文在实验室内部网络搭建实验环境。如图 4 所示。

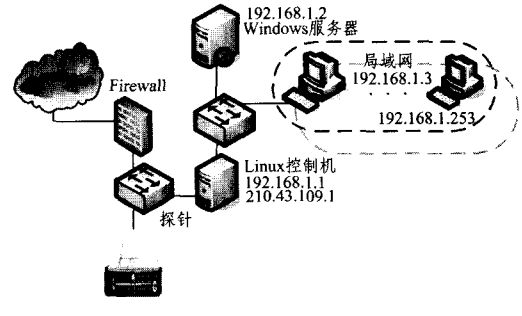


图 4 实验环境图

实验将 Linux 控制机（Intel Core 2 Duo E6400、2GRAM、双千兆网卡）以串联的方式接入网络，局域网中其他机器均通过 Linux 控制机与外界联网。在局域网出口路由器通过端口镜像的方式并联一个数据包采集探针，监测所有出入境流量。实验所采用的探针使用基于网络处理器的采集板，保证了采集数据的精度；同时以并联方式接入，不会对现有网络造成任

何影响。

为了测试整个策略库的有效性和实用性，实验从单策略运行和策略合并两方面来进行评估。

(1)单策略运行

2007 年 10 月 20 日 14:20 到 2007 年 10 月 20 日 16:00 之间，运行单策略限制整个局域网的入境流量，实验效果图如图 5 所示。

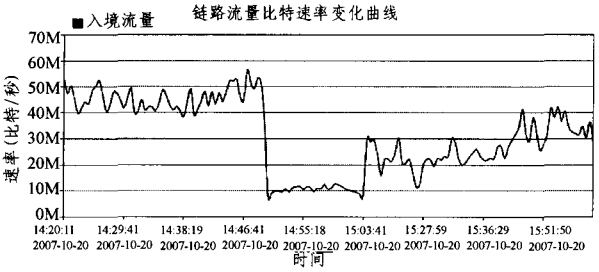
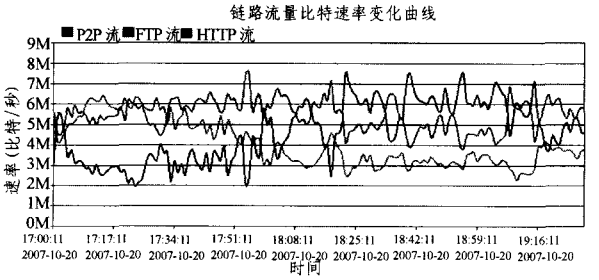


图 5 单策略运行网络状况图

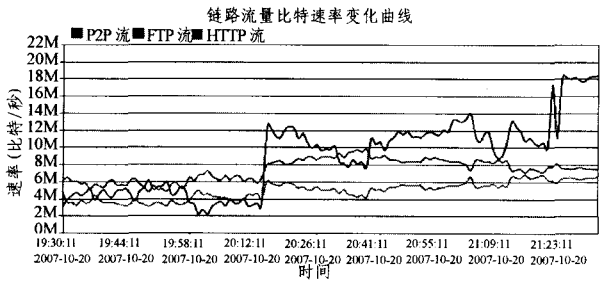
初始，链路流量速率始终介于 40～55MB/s 之间。14:48 对其限流，速率明显降至 12MB/s 以下。后于 15:00 取消策略限制，流量恢复。

(2)多策略运行

2007 年 10 月 20 日 17:00 到 2007 年 10 月 20 日 21:30 之间，对整个局域网中 P2P 流、FTP 流和 HTTP 流实施策略合并管制，实验效果如图 6 所示。



(a) 策略合并网络状况图



(b) 删除对 HTTP 限制后网络状况图

图 6 多策略运行网络状况图

在 17:00 到 21:15 之间，P2P 流、FTP 流和 HTTP 流此消彼长，共同处于一个 punish 策略环境管制中。21:15 删除对 HTTP 流的策略限制，HTTP 流量逐渐恢复。同时值得注意的是，FTP 流和 P2P 流速率有所提高，它们占用了先前 HTTP 流的带宽，但总体仍处于策略合并的受控状态。

通过上述实验结果分析，网络流量始终处于期望的受控状态，整个策略库表现出良好的管理精度。由于整个策略设计采取分而治之的思想，避免了在对制定一个较大网络的流量控制方案时容易出现的混乱状况，有效提高了带宽管理的简洁性和易用性。

结束语 本文通过深入分析 Linux 流量控制机制的具体实现，提出了一种基于可扩展策略库的流量控制管理方案，有效提高了流量控制管理的方便性。文中策略库设计遵循分层和构件化的设计理念，保证了流量管理的灵活性，并就策略管理提出了策略管理算法。算法根据用户期望的业务流信息，将一定范围内不同的业务配置到不同的规则，然后将不同的规则划分到与之对应的策略中。在保证流量控制有效性的情况下，该算法还通过采用策略合并和优先级调度机制，扩展了流量控制的业务数。基于可扩展策略库管理方案的良好特性在最后实验部分也得到证实。

参 考 文 献

[1] Devera M. Hierarchical Token Bucket Theory[EB/OL]. http://luxik.cdi.cz/~devik/qos/htb,2003

[2] Ivancic D, Hadjina N, Basch D. Analysis of precision of the HTB packet scheduler[C]// Electromagnetics and Communications, 2005. ICECom 2005. 18th International Conference. 2005:1-4

[3] Floyd S. Note on Class——Based Queueing: Seting Parameters [EB/OL]. http://www.icir.org/floyd/cbq.html

[4] Magaña E, Izkue E, Villadangos J. Analysis of protocols: Review of traffic scheduler features on general purpose platforms[J]. ACM SIGCOMM Computer Communication Review, 2001, 31(2)

[5] Semenov V K, Filippov T V, Polyakov Yu A, et al. SFQ balanced comparators at a finite sampling rate[J]. Superconductivity, IEEE Transactions, 1997, 7(2):3617 - 3621

[6] Almesberger W. Linux Network Traffic Control-Implementation Overview[R]. Technical Report, SSC/1998/037. November 1998

[7] He K K. Why and How to Use Netlink Socket[J]. Linux Journal, 2005(130)

[8] Hubert B. Linux Advanced Routing and Traffic Control HOWTO[EB/OL]. http://lartc.org/howto/

[9] Braun T, Scheidegger M, Einsiedler H, et al. A Linux Implementation of a Differentiated Services Router[J]. Network and Services for Information Society INTERWORKING 2000, 2000(1938):1611-3349

[10] Love R. Linux 内核设计与实现[M]. 北京：机械工业出版社，2006

[11] Stevens W R, Rago S A. UNIX 环境高级编程[M]. 北京：人民邮电出版社，2006

[12] Zhang L, Clock V. A New Traffic Control Algorithm for Packet—Switched Networks[J]. ACM Transaction on Computer Systems, 2005, 9(2):101-124

[13] The XML C parser and toolkit of Gnome [EB/OL]. http://www.xmlsoft.org/