基于 Netfilter 的灵巧网关的设计与实现*)

宋舜宏　杨寿保　张焕杰

(中国科技大学计算机系　合肥230026)

摘　要　网络资源需要安全有效地使用与管理,为此本文提出以 Linux 操作系统中的 netfilter/iptables 框架为基础的对网络资源进行有效控制的方法,介绍如何设计并实现一个功能合理、性能稳定、扩展性好、方便管理、易于升级的灵巧网关。

关键词　防火墙,灵巧网关,网络过滤,连接跟踪

The Design and Implement of the Smart-gateway Based on Netfilter

SONG Shun-Hong　YANG Shou-Bao　ZHANG Huan-Jie

(Department of Computer Science, University of Science and Technology of China, Hefei 230026)

Abstract　The resources of network must be used and managed securely and efficiently. So this paper prompts a method based on netfilter/iptables frame in Linux OS to control the network resources, and introduces how to design and implement a Smart-gateway with perfect function, stable performance, high scalability, easy to manage and update.

Keywords　Firewall, Smart-gateway, Netfilter, Conntrack

1　引言

目前校园网发展迅速,为了能够充分发挥网络优势,让学生能够更加便利地使用网络,同时加强网络管理,需要在很多局域网的出口设置网关,对本局域网的计算机上网进行管理,包括动态配置 IP 地址、流量限制,出口限制等,同时为了便于对这些设备的控制与管理,需要提供一个远程管理的接口。本文首先介绍了有关防火墙的知识及 netfilter 架构,接着从校园网发展需求出发,讲述如何利用专用防火墙硬件设备,在 Linux 的 netfilter/iptables 架构下,实现一个能够稳定工作、性价比高的网关,最后对系统的性能进行了分析。

2　防火墙与网关

防火墙是设置在被保护网络和外部网络之间的一道屏障,以防止发生不可预测的、潜在破坏性的侵入。是设置在不同网络(如可信任的企业内部网和不可信的公共网)或网络安全域之间的一系列部件的组合。它是不同网络或网络安全域之间信息的唯一出入口,能根据企业的安全政策控制(允许、拒绝、监测)出入网络的信息流,且本身具有较强的抗攻击能力。它是提供信息安全服务,实现网络和信息安全的基础设施。防火墙的功能主要有,强化网络安全策略;对网络存取和访问进行监控审计;防止内部信息的外泄。灵巧网关是取代路由器作为整个网络的出口设备,但基本上具有防火墙的功能,是在防火墙的基础上增加了路由方面的功能。

3　netfilter 原理

3.1　netfilter/iptables 框架结构

netfilter[1]是一种内核中用于扩展各种网络服务的结构化底层框架,其设计思想是生成一个模块结构使之能够比较容易扩展。新的特性加入到内核中并不需要重新启动内核。这样,可以通过简单地构造一个内核模块来实现网络新特性的扩展。给底层的网络特性扩展带来了极大便利,使更多从事网络底层研发的开发人员能够集中精力实现新的网络特性。netfilter 包过滤增加了许多新的功能:比如基于状态的防火墙,基于任何 TCP 标记和 MAC 地址的包过滤,更灵活的配置和记录功能,通过速度限制实现防御 DoS 攻击。

3.1.1　*钩子函数*　netfilter 为每种网络协议(IPv4、IPv6等)定义一套钩子函数(IPv4定义了5个钩子函数),这些钩子函数的功能有包过滤、NAT、mangle、conntrack 等,也可以是用户自己定义的功能。这些钩子函数均为独立模块,完美地集成到由 netfileter 提供的框架中。钩子函数在数据报流过协议栈的几个关键点被调用(如图1所示)。在这几个点中,协议栈将把数据报及钩子函数标号作为参数调用 netfilter 框架。当一个数据包进入 netfilter 时,首先进入 NF_IP_PRE_ROUTING,当通过此钩子后,包要么被路由至本机,要么被转发至其它主机,当然也可能被丢弃。如果包被路由到本机,进入 NF_IP_LOCAL_IN,如果被转发到其它主机,包将进入 NF_IP_FORWARD,在被发送到网络设备之前进入,若包是由本机产生的,则先进入 NF_IP_LOCAL_OUT,在被路由之后,进入 NF_IP_POST_ROUTING,然后发送至网络设备。

3.1.2　*钩子函数的功能*　内核的任何模块可以对每种协议的一个或多个钩子进行注册,实现挂接,这样当某个数据包被传递给 netfilter 框架时,内核检测是否有模块对该协议和钩子函数进行了注册。若注册了,则调用该模块的注册时使用的回调函数,这样这些模块就有机会检查(修改)该数据包、丢弃该数据包或指示 netfilter 将该数据包传入用户空间的队列,那些排队的数据包是被传递给用户空间异步地进行处理。一个用户进程能检查数据包,修改数据包,甚至可以重新将该

*)本文受国家自然科学基金项目(编号90104030)和安徽省"十五"科技攻关项目(编号01012013)支持。宋舜宏　硕士研究生,研究方向:网络安全。杨寿保　教授,博士生导师,研究方向:网格计算、密码学和网络安全。张焕杰　博士研究生,研究方向:网络安全,网格计算。

数据包通过离开内核的同一个钩子函数注入到内核中。这样 netfilter 开发者可以构建相当复杂的包操作。

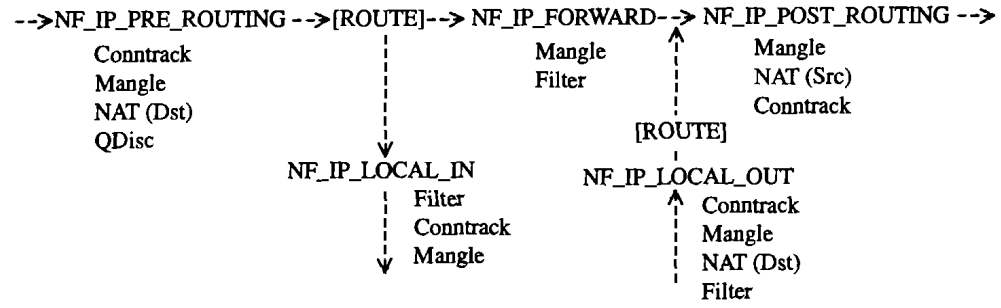

图1 netfilter 结构框架

3.1.3 *规则表的作用* 内核模块可以通过使用 iptables 注册一个新的规则表(table),并要求数据报流经指定的规则表,进行数据报选择。这种数据报选择用于实现数据报过滤(filter 表)、连接跟踪(conntrack 表)、网络地址转换(NAT 表)及数据报处理(mangle 表)。Linux2.4内核提供的这几种数据报处理功能都基于 netfilter 的钩子函数和规则表。

包过滤 filter 表[2]不会对数据报进行修改,而只对数据报进行过滤。它是通过钩子函数 NF_IP_LOCAL_IN,NF_IP_FORWARD 及 NF_IP_LOCAL_OUT 接入 netfiler 框架的。因此对于任何一个数据报只有一个地方对其进行过滤。

连接跟踪将在3.2节中作重点介绍。

NAT 表[4]监听三个 netfilter 钩子函数:NF_IP_PRE_ROUTING、NF_IP_POST_ROUTING 及 NF_IP_LOCAL_OUT,其实现是以 filter 表及 conntrack 为基础的。NF_IP_PRE_ROUTING 实现对需要转发的数据报的目的地址进行转换而 NF_IP_POST_ROUTING 则对需要转发的数据包的源地址进行转换。对于本地数据报的目的地址的转换则由 NF_IP_LOCAL_OUT 来实现。NAT 表不同于 filter 表,因为只有新连接的第一个数据报将遍历表,而随后的数据报将根据第一个数据报的结果进行同样的转换处理。NAT 表被用在 SNAT、DNAT、伪装(SNAT 的一个特例)及透明代理(DNAT 的一个特例)。

mangle 表[1]在全部五个钩子中进行注册,它是建立在 conntrack 的基础上的。使用 mangle 表,可以实现对数据报的修改或给数据报附上一些带外数据。

3.2 conntrack 原理

在 netfilter 中,conntrack[5]作为一个独立的模块来实现,它允许对包过滤代码进行扩展以使之能够简单、清晰地使用连接跟踪。可以根据包的内容,记录各个连接的状态,是状态包过滤防火墙和 NAT 必需的功能。

3.2.1 *关键的数据结构* conntrack 在 netfilter 中的位置如图1所示。其数据结构 ip_conntrack[6,7]由以下几部分组成:引用计数、原方向 tuple、应答方向 tuple、状态、超时时间、其它信息等。tuple 的结构 ip_conntrack_tuple[6,7]是用来唯一标识一个连接的,其组成如图2所示。

协议	源地址	目的地址	源端口号(类型)	目的端口号(类型)

图2 tuple 结构

每个 conntrack 记录由原方向的 tuple 和对应的反方向的 tuple 组成,而每个 tuple 分别由五个字段组成,不同协议其字段意义不同。反方向的 tuple 由原方向的 tuple 生成,其生成过程为,调换 srcip 与 dstip 的位置,然后调用协议相关的 invert_tuple 处理,其中对于 tcp 及 udp 协议是调换 sport 与 dport 的位置,icmp 是对 type 和 code 进行处理,如8/0(eacho request)的 invert 是0/0。

3.2.2 HASH conntrack 中两个方向的 tuple 被 HASH 后,根据 HASH 结果放在对应的链表中,方便快速查找。查找 tuple 是否存在,只要对 tuple 做 HASH,到对应的链表中查找,基复杂度一般为 O(1),如果构造 tuple 使得 HASH 出现冲突,查找最坏变成 O(N)复杂度,可以用来进行 DOS 攻击。

3.2.3 *conntrack 处理过程* 对经过路由进入包(PRE_ROUTING,LOCAL_OUT)的处理,是根据包的内容生成 tuple。查找系统中是否有该 tuple 的记录,如果不存在,新建立一个 conntrack 记录,并生成反向的 tuple 记录。调用对应的协议的数据包处理程序,如为 TCP 协议,则进行状态跟踪;如为 UDP 协议,则延长超时时间;如果是 ICMP 协议,两个方向的 ICMP 包相等,将直接删除 conntrack 信息。

对经过路由离开包(POST_ROUTING,LOCAL_IN)的处理,如果 conntrack 中的两个 tuple 不在链表中,则增加到链表中。整个处理过程中被 DROP 的包,不会走到这里,对应的 conntrack 的引用数为0时 conntrack 信息被删除。如果超时,则删除 conntrack 记录。

其它过程如 ALTER_REPLY_TUPLE 修改反向包时,NAT 处理代码将通知 conntrack 处理代码,将来应答的 tuple 会改变;HELPER 程序将对特定的应用数据的特殊处理,如 FTP 等;EXPECT 是对相关连接处理,如 FTP 的 CMD 和 DATA 连接是相关的。

3.2.4 *conntrack 扩展* 利用日志模块可将一个连接的如下信息记到日志服务器:连接开始、结束时间、协议、srcip、dstip、sport、dport、地址转换后的 newsrcip、newdstip、newsport、newdport、发送包数、发送字节数、接收包数、接收字节数、TCP80端口的 URL 等。这些信息是在超时处理过程中,使用 UDP 协议发送给日志服务器。连接开始时间与结束时间及统计数在连接过程中设置或修改。

如果增加一个对 TCP80端口的 HELPER 程序,对 TCP 的数据包进行处理,将能恢复其中的 URL,但特意将 URL 分到不同包中,或者使用 KEEP_ALIVE,会使记录不全。

4 系统的设计与实现

4.1 功能设计

netfilter 实现了完整的基于连接跟踪的包过滤防火墙,支持包过滤,双向地址转换,连接跟踪的处理与 NAT 处理分离,更加模块化。考虑网络管理的应用需求,利用 netfilter 的

灵活、可扩展的特点，加载相应的模块，应实现如下的功能：

灵活的地址转换(NAT)支持　支持SNAT、DNAT两种工作方式。通过使用这两种方式，除了可以完成地址转换外，还可以提供端口重定向等更灵活的服务。

完善的IP包过滤支持　支持基于协议、地址、端口、连接跟踪、速率、时间的包过滤，并支持各个规则的嵌套，方便定义IP包过滤规则。

日志功能　连接日志记录：能将每个连接的以下信息发送到日志服务器：连接开始、结束时间、相关协议信息，统计信息、TCP80端口的URL等。

告警功能　可通过网络进行及时告警，有电子邮件方式或向与入侵检测系统的中控器报警。

方便的控制界面　通过Web方式，在通过严格的身份论证后授权访问，查看相关信息，进行命令控制，及时对系统进行网上升级。可对系统进行远程的实时监控，查看流量等。

IP地址和MAC地址的绑定　支持IP地址和MAC地址的绑定，以解决盗用IP地址问题。并可以通过定期发送ARP广播，提供端口重定向等更灵活的服务。

IP隧道支持　支持IP和GRE两种方式封装的IP隧道，方便多个专网之间通过公共网络使用IP隧道高速互连。

策略路由支持　可以根据源地址、接收接口、目的地址来决定路由，适合复杂的路由决策环境。

带宽管理支持　提供灵活的带宽管理机制，使得网络管理员能根据需求分配网络带宽的使用。

实时的流量使用显示　精确到秒来显示各个网络接口的使用情况，网络的带宽使用一目了然。

TCP连接数限制　限制一个IP占用的连接总数和最多处理确认状态的总数，仅对出口的连接限制。

4.2　平台的选择

目前主要有三种实现网关的方案，一是软件型，基于通用硬件和通用操作系统之上，安装相应的防火墙软件；二是软硬一体化型，基于定制的硬件和定制的操作系统，开发专用的软件，数据包的处理主要由CPU完成；三是ASIC型，基于专用的硬件，数据包的处理由专门的ASIC或NPU完成，优点是性能非常高。

由于采用软硬一体化型技术路线，开发周期短，成本低，性能满足一般需求。考虑到容易实现、性价比较高，有一个公开原代码并具有强大的netfilter架构的Linux，因此我们采用了第二种方案。硬件采用专为防火墙设计的硬件平台，硬件配置为PIII900CPU、128M RAM、DISK ON Module IDE接口电子盘(32M)、集成3个Intel 10/100M网卡，对外提供：电源、3个RJ-45接口、COM口。

4.3　设计实现

4.3.1　设计思想　基于LINUX内核，利用成熟系统，以减少工作量，并且容易维护。我们在Linux kernel 2.4中的netfilter/iptables的基础上，实现基于状态的包过滤及双向地址转换等功能，采用WWW管理界面，使用方便，运行中的系统可以现场进行软件升级，方便可靠。

4.3.2　文件结构的设计　文件包括三部分，一部分为DOS系统的文件，用于启动Linux，之所以从DOS启动Linux系统，是为了方便升级。另一部分为Linux系统文件，包括Linux内核与根文件系统。内核需要根据系统配置进行编译[10]，注意需要将netfilter相关部分编译进去，而不要作为模块加载。根文件系统包括除Kernel以外的所有需要的文件，启动时将系统读入RAM，作为ramdisk。其中内核为0.7M，initrd文件大小为1.5M。第三部分为配置文件，为文本格式文件，用以启动要执行的命令。

启动过程为，首先启动到DOS操作系统，通过autoexec.bat执行loadlin，加载kernel和initrd文件，进入Linux后，用mount命令加载各所需硬件设备，并把配置文件拷贝到linux系统的/etc目录下，按照配置文件对系统进行设置，配置文件是作为script执行的。

4.3.3　文件系统内容　首先需要装glibc[10]，为busybox[11]提供支持库。还需要busybox提供Unix的utilites，busybox具有常用的Linux命令，而且非常小，不到800k。用thttpd[12]作为WWW服务器，同样是非常小，而且支持CGI。iptables是netfilter的用户接口程序。dhcpd[13]为一个动态主机配置服务器程序。

4.3.4　编写规则　根据应用的不同可以进行相应的配置并编写规则。下面以灵巧网关策略路由应用为例说明如何编写规则。其它规则的编写可参考iptables的使用说明[2]。

对于有多个出口的应用，如连接了公网的教育网，如果内部的服务器希望被外面访问，则需要使用策略路由技术。策略路由是指在决定一个IP包的下一跳转发地址时，不是简单地根据目的IP地址决定，而是综合考虑多种因素决定。一般的路由器策略路由实现只能根据单个IP包中的源地址进行判断，在使用时并不方便。应该能够根据更多的因素进行决定，如可以根据原来数据包的信息来决定以后的路由，在使用时更加灵活。

如果网络内部地址为192.168.0.*，两个出口的地址分别是202.38.64.*和218.22.21.*，其中202.38.64.10和218.22.21.10都对应到内部的服务器192.168.0.10。这时，需要如下设置，让所有发送给202.38.64.10的应答包经过教育网出去，所有发送给218.22.21.10的应答包经过电信网出去，这样才能保证网络通讯正常：

```
modprobe ipt_conntrack
iptables -t mangle -F
iptables -t mangle -A PREROUTING -j MARK --set-mark 1 -m conntrack --ctorigdst 202.38.64.0/24
iptablest - mangleA - PREROUTING -j MARK --set-mark 2 -m conntrack --ctorigdst 218.22.21.0/24
ip rule add from 0/0 table main pref 90
ip rule add fwmark 1 table 100 pref 100
ip rule add fwmark 2 table 120 pref 100
ip rule add from 0/0 table 130 pref 200
ip route add 0/0 via G1 table 130
ip route add x/y via G2 table 130
ip route add 0/0 via 202.38.64.x table 100
ip route add 0/0 via 218.22.21.x table 120
```

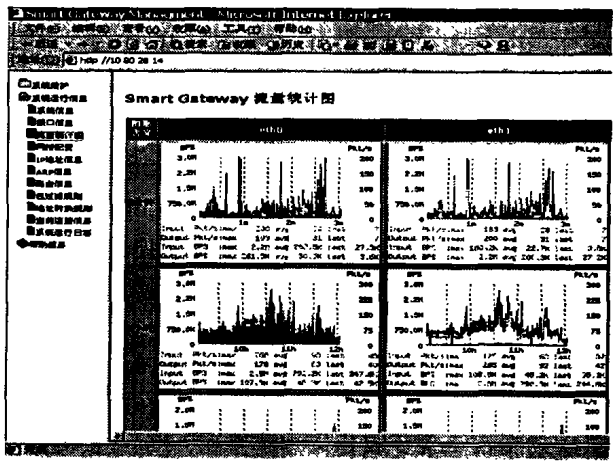

图3　流量统计图

4.3.5 管理界面的设计 管理所需数据通过两个方面得到，一是充分利用 proc[9]文件系统的相关数据，二是使用 iptables 命令进行一定内容过滤后获得相应信息，在此基础上编写 CGI[14]程序，将所需信息通过 Web 服务传送。下面给出了 Web 管理功能部分界面，如图3与图4所示。

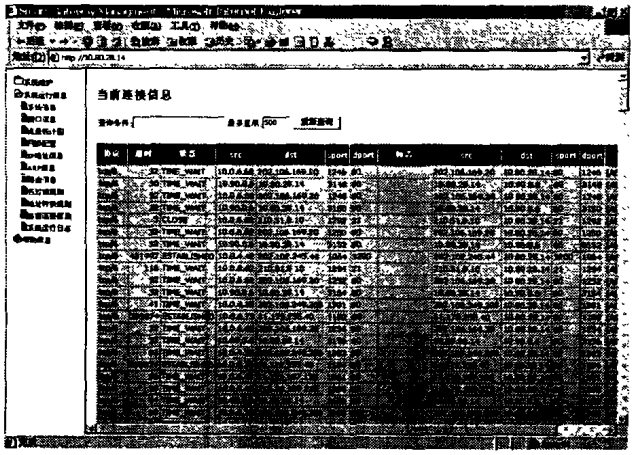

图4 当前连接信息

4.3.6 系统升级 升级前，将原来的文件进行备份，更换 Linux 文件系统的两个文件，kernel 文件和 initrd 文件后，重新启动即可，如果升级失败则可重新使用老版本，使得升级安全、方便。

5 系统性能分析

灵巧网关提供3个10/100M 自适应以太网接口，最多可以扩展到5个，适应当前网络速度不断提高，带宽不断增加的应用环境。灵巧网关的软件经过特别优化，足以应付各个接口满负荷工作。在设计时就将高可靠性作为首要特性考虑，在硬件、软件层协同采用了多项技术保证系统的可靠性。经过实际运行，系统稳定，适合长期工作，最长连续工作纪录达到450天。

由于每个 conntrack 需要占用一定内存，如果出现恶意攻击，使连接数过多时，可能导致系统无法正常工作，可以通过超时设置来防止这种情况发生。由于对 conntrack 是用 HASH 方法进行查找，如果恶意伪造数据包的报头，可能会使 HASH 出现冲突，导致系统性能下降。

结束语 灵巧网关是一个设计合理，功能齐全，性能稳定的系统，完全满足校园网的应用需求。同时由于其升级与管理非常方便，因此具有很强的实用性。根据配置的不同，灵巧网关可以作为路由器、地址转换设备、带宽分配器、包过滤器等来使用，具有高度的灵活性。还可以根据客户需求，在灵巧网关上开发新的功能。

参 考 文 献

1 The netfilter framework in Linux 2.4. http://www.gnumonks.org/papers/netfilter-lk2000/presentation.html
2 http://www.netfilter.org/documentation/HOWTO//packet-filtering-HOWTO.html
3 http://www.netfilter.org/documentation/HOWTO/de/netfilter-hacking-HOWTO.html
4 http://www.netfilter.org/documentation/HOWTO//NAT-HOWTO.html
5 http://gnumonks.org/ftp/pub/doc/conntrack+nat.html
6 http://cvs.gnumonks.org/netfilter-tools/doc/conntrack.sgml?rev=1.2
7 http://pc1.peanuts.gr.jp/~kei/Kernel-Snapshot/linux/net/ipv4/netfilter/
8 毛德操，胡希明. Linux 内核源代码情景分析(上、下). 浙江大学出版社，2001
9 陈莉君. 深入分析 Linux 内核源代码. 人民邮电出版社，2002
10 Shah S. Linux 管理员指南. 机械工业出版社，2001
11 http://www.busybox.net/
12 http://www.acme.com/software/thttpd/
13 http://rpm.pbone.net/index.php3/stat/4/idpl/44124/com/dhcptd-2.1.i686.rpm.html
14 Guelich S. CGI 编程-使用 Perl(第二版). 中国电力出版社，2001

(上接第39页)

到传送最上级节点和从接收最上级节点到目的节点的路径固定，最上级节点之间的通路在小范围内根据 QoS 路由建立虚通道，沿路所需资源全部进行预留，因而能够完全满足实时通信在延迟、抖动等方面的 QoS 要求。

(2)主干节点管理操作简单，适用于大规模网络。在建立实时通信连接时，可以根据所在级别实际通信统计情况设置虚通道的大小。如在最低级别上每次连接建立的虚通道可供1个实时通信应用使用，但在图2的 D2和 B2之间则可以考虑每次连接建立的虚通道可供10000个实时通信应用使用，由于虚通道的大小设置是基于实际应用统计，其实际利用率应该非常高；而且 D2下属节点和 B2下属节点间的实时通信都可以使用该通道，而 D2和 B2只需要管理1个而不是10000个虚通道，管理操作大为简化。

(3)保留分组转发机制灵活利用网络资源的特点。同级内部可以根据路由协议选择路由，能够充分利用网络资源。

(4)路由协议简单。路由局限在同级数百或者数千个节点的范围之内，可以采用最简单的路由协议，QOSPF，甚至增添 QoS 参数的 RIP 协议都可以满足要求。

(5)网络设备低成本高效率。非终端节点可以根据仅根据目的地址确定大部分分组的下一转发节点，仅当目的地址为同级其它节点的下属节点时才需要查找路由表选择转发路径。同级节点数目有限，各级路由表表项都可以控制在很小规模以内，操作简单，转发效率大大提高，成本大幅下降。

(6)地址可无限扩展。前文已详细介绍，在此不再赘述。

小结 区域路由网络综合了电路交换和分组交换方法的优点，在保证实时应用服务质量的同时还可以灵活利用网络资源，同时能够解决实时应用下综合服务中主干节点负载过重问题和区分服务中不能完全保证实时服务 QoS 问题。其网络设备低成本高效率，特别适用于大规模网络，是一种适用于互联网的、值得进行深入研究和尝试的 QoS 实现方案。

参 考 文 献

1 McDysan D. QoS & traffic management in IP & ATM networks [M]. New York: The McGraw-Hill Companies，2000. 32～54
2 顾尚杰，薛质. 计算机通信网基础[M]. 北京：电子工业出版社，2000. 27～28
3 RFC1633. Integrated Services in the Internet Architecture: an Overview[S]. 1994
4 RFC2475. An Architecture for Differentiated Services[S]. 1998.
5 翁惠玉，刘芳，陈志英，等. Intserv/RSVP 的现状及存在的问题[J]. 数据通信，1999(4)：4～7
6 李玮，卢燕飞. 区分服务 Diffserv 体系结构及其工作机制剖析[J]. 电信快报，2002(6)：15～18
7 唐宝民. 电信网技术基础[M]. 北京：人民邮电出版社，2001. 18～20