一个嵌入式网络流量识别系统的设计与实现

An Embeded Network Traffic Identification System

杨　铮　李国元　左敏　（重庆大学 计算机学院 重庆 400044）

摘　要： 文章介绍一个在嵌入式环境下的网络流量识别系统设计并实现，该系统具有可扩展的特征。采用流量特征配置、Linux 内核模块等技术灵活扩展系统识别能力。本文剖析了流量识别、流量分类和系统扩展接口等高效的实现方法。通过测试验证了该系统能正确有效的识别企业内部网的网络流量。

关键词： 嵌入式系统 linux 内核模块 流量识别 流量分类

1　引言

随着网络技术的发展，企业已经由管理网络转化到管理应用，由管理连接转化到管理生产力。在互联网上的应用也从过去简单的 Ftp，E－mail 等应用发展到当前的 P2P、MSN、视频会议以及一些较大型的数据库应用，如 Oracle，SQL 等，传统的网络限制太多或者缺乏可视性且带宽消耗严重。难以管理的繁重流量负载会减损应用性能，数据库、ERP 系统、视频会议等关键应用效能受损。企业的 CIO 都在为企业网络资源实际分配的认知和控制而努力。IP 网络必须能够提供增强的服务，例如，提供各种不同级别的 QoS（Quality of Service）服务、分布式防火墙、IP 安全网关，按流量计费等。所有这些增强的网络管理服务的提供都依赖于对网络承载的各种流量的识别和分类。因此，研究实时、在线的网络流量识别方法，提供快速、准确、高效的流量识别，对于了解当前网络流量特征、提高网络运维水平，实现科学的网络运行维护和管理具有非常重要的意义。

2　系统开发平台

系统采用嵌入式体系结构，基于先进的内置两个千兆网口的专用网络处理器 MPC8349 设计。MPC8349E PowerQUICC Ⅱ Pro 处理器集成了增强的 e300 PowerPC 内核及 DDR 内存控制器，支持双 10/100/1000 Mbps 以太网控制器、双 32 位/单 64 位 PCI 控制器、集成安全引擎、高速 USB 2.0 主机和设备控制器、4 频道 DMA、DUART、串行外围设备、通用 I/O 和系统定时器等。操作系统采用嵌入式 Linux，具有高稳定性、安全性和良好的可移植性等优点，对网络通信协议支持丰富、完善，可满足多种网络通讯、网络服务的要求。

3　系统框架设计

系统框架模型如图 1 所示，由数据采集引擎、流量识别引擎、流量分类引擎、控制与整形引擎、流量统计引擎、用户空间通信引擎和 Web 管理组成。

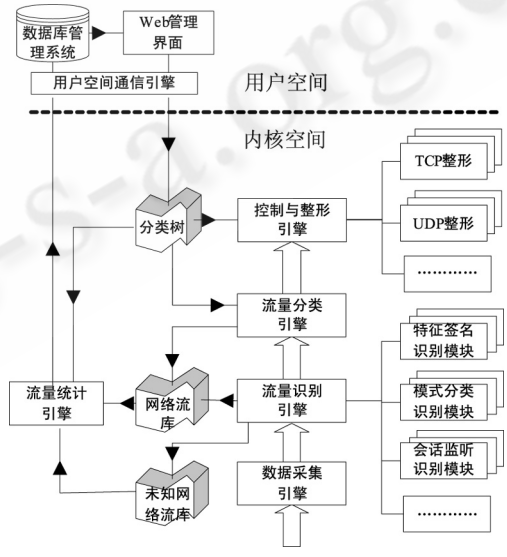

图 1　系统框架模型图

每块可以在运行时添加到内核的代码，被称为一个模块，这意味着可以在系统正在运行着的时候增加或删除内核的功能。Linux 内核提供了对许多模块类型的支持，包括但不限于设备驱动。每个模块由目标

代码组成,可以动态连接到运行中的内核中。本系统包括一个主模块和若干子功能模块组成,主程序模块负责主要流程的顺利执行和各个子功能模块的合理调度。各个子功能模块与主程序模块相对独立,主程序模块与各个子模块之间通过约定好的接口进行访问。整个流量识别系统没有固定需要的模块,一切视实际的系统环境和要识别的网络流量的需求来确定。在该框架模型中,系统的大部分功能都依赖于插入到系统中的模块所提供的服务。

4 系统实现

4.1 数据包采集

数据包采集引擎工作在 Linux 透明网桥[1] 模式下,主要负责实时收集以太网网络数据包并送交识别引擎。常用的抓包方法都是将内核网络协议栈中数据包复制到用户空间如 libpcap,这样做增加了内核空间与用户空间的切换次数,降低了系统运行效率和识别的实时性,而且不能对网络流进行速率控制与整形。本系统将数据包采集引擎入口以钩子(hook)函数形式嵌入到 Linux 网络协议栈中,网卡设备驱动收到一个帧首先交给 handle_bridge 处理,它会调用 br_handle_frame 钩子函数。该帧被传输到 br_handle_frame_finish 之前会调用数据采集引擎的入口函数。该数据包采集引擎独立于 Netfilter[2] 框架,能够有效保证数据包采集的完整性。如图 2 所示:

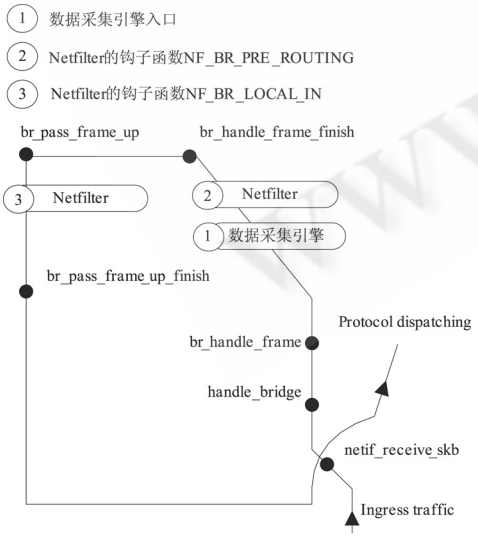

图 2 网桥模式下数据包处理流程

4.2 可扩展性

可扩展性设计是整个系统设计的关键,它不仅是本系统适应不断变化发展的网络应用的需要,而且是扩展自身功能的关键。各个扩展接口由一系列描述接口的结构体和注册函数组成。本文以流量识别引擎的扩展接口设计为例进行详细说明,其它扩展接口采用类似设计。流量识别引擎包括普通流量识别接口和未知流量识别接口。普通流量识别接口在每收到一个数据包时顺序遍历调用各个识别模块入口函数。未知流量识别接口主要是为了对未知流量进行聚类分析和提取特征,它通过内核线程的方式在后台运行,触发时机为未知网络流集达到一定数量或者某个时间间隔后。注册函数需要在主程序模块通过调用 EXPORT_SYMBOL(name)输出到内核符号表中,使注册函数在其它模块中可以调用。各个识别模块必须定义及初始化至少一个 identify_ops 的结构体实例,并通过相应的注册函数按照指定优先级从高到低的顺序挂载到主模块对应的识别引擎接口上。识别引擎接口基本数据结构如下所示:

```
typedef struct protocolNode * identify_fn(const struct sk_buff * );//识别模块入口函数类定义
struct identify_ops {
    struct list_head list;//Linux 内核双向链表结构体
    identify_fn * identify;//识别模块入口函数
    enum PRIORITY priority;//0 -7 识别模块优先级
    struct app_protocol * protolisthead;//该识别模块可以识别的应用协议链表
    ……
};//接口数据结构
int register_identifyops(struct identify_ops *);//普通流量识别模块接口注册函数原型
int register_identifyunkonwnops(struct identify_ops *);//未知流量识别模块接口注册函数原型
```

4.3 流量识别

流量识别引擎是整个系统的核心模块,其调用各种识别模块对各种网络应用进行识别,识别的结果以网络流的形式进行存储,识别模块无法识别的网络流放入未知网络流库,并缓存应用层数据。

定义 1 一条网络流标识某种协议所创建的一个

连接会话，一条网络流定义为六元组：NFlow ::= < SIP，DIP，SPort，DPort，TProtocol，AProtocol >。

其中，SIP，DIP，SPort，DPort，Tprotocol，AProtocol 分别表示该网络流的源 IP 地址、目的 IP 地址、源端口号、目的端口号和传输层协议号和应用层协议号，以上六元组段可以唯一地确定任意一个数据包属于哪个特定的网络流。

对于网络流的存储采用链式哈希表，其便于网络流频繁的插入和删除。哈希值的计算选用 Linux 内核提供的 jhash_3words 函数，该函数已经在 Linux 内核数据结构中广泛使用，被证明具有较高的效率和稳定性。网络流库的基本数据结构：

```
struct NFlow {//网络流数据结构
    struct list_head list ;//用于嵌入到网络流库中
    struct timer_list timeout ;//某段时间内该流没有数据传输则释放。
    struct list_head classify ;//用于链接分类结果
    struct flow_tuple tuple[FLOW_DIR_MAX] ;//该网络流的六元组信息
    ……
    } ;
struct NFlowSet {//网络流库
    atomic_t numFlows ;//当前规则集的数量
    rwlock_t flowset_lock ;//读写锁，用于网络流库的互斥访问
    struct list_head * flowset_hash ; //哈希表头数组。
    ……
    } ;
```

由于流量特征的复杂性，要准确识别出各种网络流量，往往根据流量特征的不同，需要采用不同的技术。典型的识别技术有以下几种：①基于应用层特征签名（Signature）的识别[4,7]。针对应用层报文进行深度内容检查，判断其中是否含有符合特定应用特征的签名。②基于会话（Session）的分类。应用层协议的特征签名有时只存在于一个会话的开始的某些报文中。该会话的其他报文不包含特征签名。③基于流状态的统计识别。多数网络应用在进行实际的数据传输之前都要进行一些控制信息的交互，而不同的网络应用进行信息交互的方式都有各自的特点，因此可以在 IP 层通过流量统计特征的方式识别不同的流量，这种方法常用来识别一些加密或非公开协议的应用所产生的流量。

基于应用层特征签名的识别，由于许多协议特征表现为在应用层数据部分的一个特征签名（signature）[5]，对于这类协议可以采用统一的识别接口进行处理。首先应该进行应用层协议的分析，对于公开协议的应用，通过分析其协议标准来提取特征 对于非公开协议的应用，采取抓包加逆向分析的方法，提取出协议特征。由于提取到的协议特征往往不具有唯一性，因此本系统采用正则表达式来描述协议特征。一个应用层协议对应一个特征配置文件，存放在指定路径下。

定义 2　每个特征配置文件内容定义为：

ASF ::= < AN，DL，TS >

其中，AN，DL，TS 分别表示协议名、数据包长度、特征字符串。用户可以构造自己的协议特征文件，存放在特征配置文件目录下进行动态扩展。

识别模块在初始化时首先读取各个特征配置文件中的特征字符串，并调用正则表达式编译函数 regcomp 将特征字符串编译成正表达式数据结构，最后通过调用正则表达式执行函数 regexec 对应用层数据进行正则匹配。在识别过程中需要对应用层数据包进行缓存，因为应用层特征有可能分布在多个数据包中。表 1 为特征配置文件内容示例（其中 Paket Length 为‘ * ’表示长度不固定）：

表 1　特征配置文件内容示例

AN	DL	TS
ntp	48	^([\x13 \x1b \x23 \xd3 \xdb \xe3] | [\x14 \x1c $].……？？？？？？？？？[\xc6 – \xff])
smb	*	\xffsmb[\x72 \x25]
bittorrent	*	^(\x13bittorrent protocol | azver\x01 $ | get/scrape→info_hash =) | d1 :ad2 :id20 :| \x087P\) [RP]
smtp	*	^220[\x09 – \x0d – ~] * (e？smtp | simple mail)

基于会话的分类，该类网络应用通常是由控制和数据会话两部分组成，通信双方的数据端口是在控制会话中动态协商产生的，而且控制和数据会话可能使用不同的传输层协议。常见的基于会话的应用如表 2 所示：

表 2　基于会话应用举例

应用	使用协议	控制链路	数据链路
RealOne Player and QuickTime Player	RTSP HTTP	554/TCP, 80/TCP	554/TCP, 6970 – 32000/UDP,or 80/TCP
FTP	FTP (Active, Pssive)	21/TCP or user – defined/TCP	20/TCP (active), Dynamic/TCP(passive)
H. 323 IP Phone	H. 323	1720/TCP, and dynamic/TCP	Dynamic/UDP
SIP IP Phone	SIP	5060/UDP	Dynamic/UDP
Windows Media Player	MMS HTTP	1755/TCP, 80/TCP	1755/TCP , 80/TCP, 1024 – 5000/UDP or 1 – 65000/Multicast

由于各种基于会话的网络协议的控制会话的特征和数据会话生成方式各不相同,数据端口可能需要通过计算才能获得,因此需要为各种基于会话的应用开发单独的识别模块。对于明文传输的会话应用,本文提出采用网络流状态跟踪算法 FSTC(Flow State Track Classification),该算法在基于特征签名识别方法的基础上,首先识别出控制会话并且进行缓存,然后跟踪监听通信双方的会话过程进而识别出数据会话的相关信息。具体的过程由算法 1 给出:

算法 1　网络流状态跟踪算法 FSTC

输入:数据包 skb(struct sk_buff[1] 实例)

输出:应用协议标识 APid。

算法过程:

```
1、初始化:需要返回的应用协议标识 APid = null;
   根据数据包信息初始化临时网络流 nf_tmp;
2、采用状态跟踪算法识别控制和数据会话:
   If(nf_tmp 是已缓存的控制会话流) {
      If(断开连接命令) {
         释放缓存控制会话流;goto finish;
      } Else if(数据端口通知命令) {
刷新控制流定时器;计算数据端口并缓存;
         }
} Else if(客户端数据端口连接) {
      APid = data_control;//数据会话建立;
      } Else {
         If(控制会话连接) {
            缓存该控制会话;设置定时器;
            APid = control_channel;//控制会话
         }
}
Finish:return APid;
```

图 3 为 Ftp 被动模式识别过程的有穷状态机。

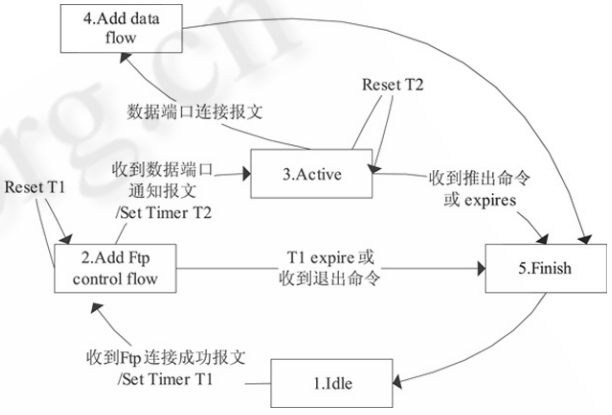

图 3　Ftp 被动模式识别有穷状态机

对于数据加密传输的数据,可以采用基于流状态的统计识别等方法。随着流量识别技术的发展,对于今后可能出现的新的流量种类和流量识别技术,可以有针对性地开发相应的识别模块,并挂载到相应流量识别接口上。

4.4　流量分类

流量分类是根据数据包本身携带的信息或与数据包有关的信息(OSI 第二层到第七层的相关信息)索引预先设置的分类树,查找匹配的规则来达到区分数据包的目的。流量分类是进行流量统计的基础,也为流量控制与整形提供依据。本系统支持两种分类方式:①基于应用的流量分类树。该分类方式在注重站点上自身应用而非为多个位置划分流量的分布式分支机构中是最通用的。②基于位置的流量分类树,根据位置、用户组、子网、主机列表或 IP 地址分类流量。并能够对每个位置进行扩展,以将其应用包含进去。该分类方式在主站点上利用不同战略为多个分支机构或部门管理流量方面是最通用的。多种分类方式决定了系统必须有两颗以上的分类树,根据用户的配置动态决定当前使用的分类树。本文采用逐行扫描算法遍历整棵分类树方法对网络流进行分类。

4.5 流量统计

流量统计引擎将收集到的流量传输给用户空间通信层,并由后者负责存储。流量统计引擎以内核线程的方式在内核后台运行,并通过 netlink [2] 套接字与用户空间通信层进行数据交换。流量收集包括两种模式:①主动模式。流量统计引擎主动去收集分类树和网络流库中的流量②被动模式。当分类树和网络流库超时后向流量引擎递交网络流量。

5 运行结果分析

系统必须放在它可查看所有您希望它管理的流量的位置,可以部署在主站点和分支机构上 WAN 链路路由器和/或因特网链路路由器之后。为验证系统识别的准确率,分别以 BitTorrent、HTTP 和 FTP 这三种分别代表 P2P、Web 浏览和文件传输的应用为例进行测试,在测试同时,使用 Sniffer 抓包工具捕获流量,根据系统识别出的应用流量与该应用实际传输数据量的对比,计算出系统对该应用的流量识别率,通过手工分析获得系统对该应用连接的识别率。系统在 Intranet 与出口路由器之间,8Mbps 出口速率下运行部分识别结果如表 3 所示:

表 3　识别结果分析

应用名	实际流量(KByte)	识别出的流量(KByte)	流量识别率	实际连接数	识别出的连接数	连接识别率
bittorrent	9250886	9002751	97.3%	5248	4940	94%
http	235197	235197	100%	864	864	100%
ftp	1240917	1240917	100%	312	312	100%

运行结果显示,本系统能够比较准确地识别出表中所列的三种流量和连接。对于 BT 流量的识别,经研究发现,为防止上文提到的基于应用层特征签名的流量识别,一些 BT 客户端软件采用了一定的技术手段,例如,BitSpirit 采用一种"扩展握手协议"的方法,其实现是在含有 BT 特征签名的 TCP 握手信息前加入了一个 HTTP 包头,BitComet 则是将握手信息加密扰乱。这两种方法可以在一定程度上突破基于 BT 协议特征签名的流量识别,所以实验数据中的 BT 流量识别率并未达到 100%。但由于这些方法局限于同一种客户端软件之间才能互联,启用该选项后,资源很少,下载速度可能会比较慢,因此使用者较少。

6 结语

如何对网络流量进行科学有效的管理和控制,保证用户关键业务的服务质量,有效管理消耗大量带宽的 P2P 文件传输、流媒体或网络游戏等非关键流量,是目前困扰很多网络运营商和企业网络管理人员的迫切需要解决的问题。流量可控性的前提是可识别,论文通过对网络中各种流量特征及流量识别技术的分析和研究,设计并实现了一个嵌入式环境下基于 Linux 的流量识别系统,能够根据需要扩充新的识别模块,对未知的新型业务流量快速识别。

参考文献

1 KLAUS WEHRLE, FRANK PAHLKE. Linux 网络体系结构: Linux 内核中网络协议的设计与实现. 北京: 清华大学出版社, 2006.

2 CHRISTIAN BENVENUTI. 深入理解 Linux 网络内幕. 南京: 东南大学出版社, 2006.

3 JONATHAN CORBET, ALESSANDRO RUBINI, GREG KROAH – HARTMAN. Linux 设备驱动程序, 第三版. 魏永明, 耿岳, 钟书毅, 译. 北京: 中国电力出版社, 2006.

4 田立勤, 林闯. 报文分类技术的研究及其应用. 计算机研究与发展, 2003, 40(6): 765 – 775.

5 余胜生, 张宁, 周敬利, 胡熠峰. 一种用于大规模规则库的快速包分类算法. 计算机工程, 2004, 30(7): 49 – 51.

6 ANDREW W. MOORE, KONSTANTINA PAPAGIANNAKI. Toward the Accurate Identification of Network Application. in Passive & Active Measurement Workshop 2005.

7 SUBHABRATA SEN, OLIVER SPATSCHECK, DONGMEI WANG. Accurate, Scalable In – Network Identification of P2P Traffic Using Application Signatures. In: Proceedings International WWW Conference, New York, USA, 2004.

8 BABOESCU F, VARGHESE G. Scalable Packet Classification. ACM SIGCOMM01, 2001.