第 37 卷　增　刊　计　算　机　工　程　2011 年 12 月
Vol.37　Supplementary Issue　Computer Engineering　December 2011

·网络与通信·　文章编号：1000—3428(2011)增刊—0098—03　文献标识码：A　中图分类号：TP393

# OpenDPI 报文识别分析

魏　永，周云峰，郭利超

(太原卫星发射中心，太原 036301)

**摘　要：**传统数据流分析方法不能实现对数据流的精细化管理。为此，介绍开源深度报文检测程序 OpenDPI，研究其源代码的设计结构和功能，分析报文识别的原理和过程，解决数据流会话的有效期和多报文处理问题。分析结果表明，OpenDPI 可实现对报文协议规则的动态添加与更新。

**关键词：**OpenDPI 源代码；网络管理；数据流；深度报文检测；报文识别

# Analysis of Message Identification for OpenDPI

**WEI Yong, ZHOU Yun-feng, GUO Li-chao**

(Taiyuan Satellite Launch Center, Taiyuan 036301, China)

**【Abstract】**The traditional Internet traffic analysis can not longer meet the requirements in the elaborate management of Internet traffic. Therefore, the OpenDPI program of open source Deep Packet Inspection(DPI) is introduced. This paper studies the design structure and the function of the OpenDPI code, and analyzes the principle and the procedure of its packet identification, to solve the validity of the data flow session and the problem of the multi-message processing. Analysis result shows that OpenDPI enables message protocol rules to dynamically add and update.

**【Key words】**OpenDPI source code; network management; data stream; Deep Packet Inspection(DPI); message identification

## 1　概述

随着网络的普及，网络上的数据急剧增加，面对海量的数据，需要区分有效数据和不良数据；网络上的应用也日趋多样化，不同的应用需要不同的服务质量(Quality of Service, QoS)，如语音的实时性要求远高于邮件；面对越来越多的 P2P 业务，占用了大量带宽，需要限制其所用带宽，以保证其他业务的运行；对于运营商，部分业务也需要根据流量进行收费。以上需求对网络管理提出了更高的要求，如何识别网络上的数据流成为网络管理的基础，传统的数据流分析方法仅仅分析 IP 包 4 层以下的内容，包括源地址、目的地址、源端口、目的端口以及传输协议类型。但针对网上越来越多的数据报文，采用传统的识别方式已不能有效的识别，如 P2P 通信一般采用是随机端口，迅雷的数据传输复用在 HTTP 业务之上等，而且这些数据所占比重越来越大，于是产生了一种新的协议识别方式，即深度报文检测(Deep Packet Inspection, DPI)[1]，它在传统的分析基础上，增加了对应用层有效负载的分析，可识别出各种应用业务。本文主要分析了开源深度报文检测程序 OpenDPI，从其设计结构、原理、实现过程等多方面进行了深入的分析。

## 2　OpenDPI 简介

OpenDPI[2]作为一个开源的深度报文检测系统，自从 2009.09 发布 1.0.0 版本，得到了业界的广泛认同，目前的最新版本是 2011.06 发布的 1.3.0 版本，新版本中增加了对 IPv6 的支持，实现了更多的协议识别，现能分析 118 种协议。它根据网络协议对数据包进行分类计数，得到各协议数据包的数量、大小、协议类型等信息。OpenDPI 源自商业的 PACE，相对于 PACE 引擎，OpenDPI 不支持检测加密协议，也不使用任何分类启发式和行为分析。

OpenDPI 相对于其他众多的深度报文检测软件，有自己的优点，它是开源的，代码比较精简，运行稳定、错误检测率更低，它作为一个协议分析引擎，允许第三方开发者在不同的商业解决方案上编写 DPI 应用程序。因此，可以根据需要，修改其代码，并部署在服务器上，满足特定的需求，因而研究它具有理论和现实的意义。

### 2.1　OpenDPI 的设计结构

OpenDPI 设计结构主要分为 3 个部分：

(1)对用户输入的命令行进行处理，OpenDPI 有 2 个参数 -e 和-f，-e 用于打印 DEBUG 信息，-f 用于制定待处理的包文件，处理完命令行以后还要对待处理的协议进行设置，OpenDPI 用 IPOQUE_PROTOCOL_BITMASK 记录需要分析的协议类型，对需要分析的相应位进行设置。

(2)包文件处理，OpenDPI 调用函数 pcap_loop 对包文件循环处理，主要处理过程是在函数 packet_processing 中实现的，函数分析每个报文的应用层协议类型，记录协议类型数量和大小。

(3)分析结果输出，打印分析结果，包括 IP 协议包数量，大小，需要检测的各应用层协议包数量、大小。

### 2.2　核心数据结构

本文介绍核心的数据结构，它是程序运行的基础。

(1)ipoque_id_struct：记录协议标识的结构体，用于区分当前流或数据包属于 skype 或者别的协议，其中还包括少许与协议密切相关的数据结构。

(2)ipoque_flow_struct：记录协议相关的流状态和信息。

(3)ipq_call_function_struct：在初始化使用，设置需要解析协议的条件和函数入口地址，结构定义如下：

```
typedef struct ipq_call_function_struct {
  IPOQUE_PROTOCOL_BITMASK detection_bitmask;
```

**作者简介：**魏　永(1981－)，男，工程师、硕士，主研方向：网络安全；周云峰，高级工程师、硕士；郭利超，工程师、硕士

**收稿日期：**2011-11-30　**E-mail：**wwweiyong@163.com

?1994-2015 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

```
    IPOQUE_PROTOCOL_BITMASK excluded_protocol_bitmask;
    IPQ_SELECTION_BITMASK_PROTOCOL_SIZE
ipq_selection_bitmask;
    void (*func) (struct ipoque_detection_module_struct *);
  } ipq_call_function_struct_t;
```

其中，detection_bitmask 表示当前已经检测到的协议类型，比如 Yahoo 的传输是承载在 HTTP 业务上，则该 detection_bitmask 就要设置为 HTTP，表明当一个数据流检测出是 HTTP 协议时，还需要进一步的分析；如果需要识别的协议承载在 TCP 或 UDP 上，则设置为 IPOQUE_PROTOCOL_UNKNOWN；excluded_protocol_bitmask 表示数据流已经排除的协议类型；ipq_selection_bitmask 表示数据包的下 4 层协议信息，如 IPv4、IPv6、TCP、UDP 是否有负载等情况；func 函数表示解析函数入口地址。

(4)ipoque_packet_struct：数据包结构，定义了分析数据包时的各种数据包头结构，如 IP 报文头、TCP 报文头、UDP 报文头、HTTP 报文头和需要用到的变量。

(5)osdpi_flow 结构：osdpi_flow 定义了一个数据流以及与其相关的流状态和信息。

(6)osdpi_id 结构：它记录了所有的 IP 地址及其相关信息。

(7)ipoque_detection_module_struct 结构：这是全局的数据结构，存储报文处理过程中的所有变量和参数。

## 3 报文解析的实现

### 3.1 初始化

所有的初始化工作都在函数 setupDetection()中进行，其流程如图 1 所示。

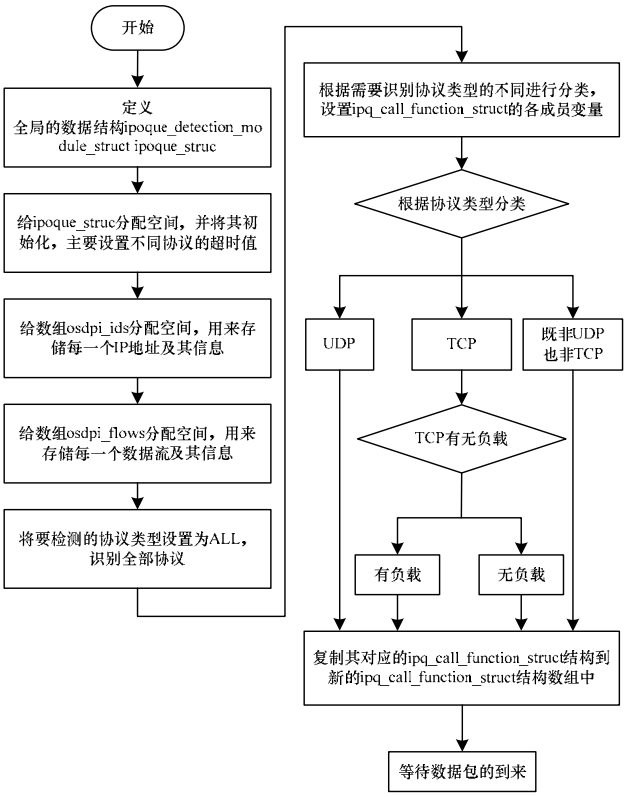


**图 1 OpenDPI 初始化流程**

(1)ipoque_struct 的初始化

定义全局数据结构并将其初始化，初始化时第 1 个参数为时间戳的值，不同系统采用的值也不同，范围为 1 ms~10ms，这里定义为 1 000，表示 1 s；第 2 参数为 ipoque_struct 结构分配存储空间；最后 1 个参数为打印调试信息，方便用来进行调试。具体函数如下：

```
static struct ipoque_detection_module_struct *ipoque_struct=
NULL;
ipoque_struct=ipoque_init_detection_module(detection_tick_resol
ution, malloc_wrapper, debug_printf);
```

(2)为了程序的稳定，在运行时不进行任何储存空间的分配，初始化时为数组 osdpi_ids 和 osdpi_flows 分配空间。osdpi_ids 用来存储每一个 IP 地址及其信息；osdpi_flows 用来储存每一个数据流及其信息。

(3)设置要检查的协议，设置检查全部协议，实现函数如下：

```
IPOQUE_PROTOCOL_BITMASK all;
IPOQUE_BITMASK_SET_ALL(all);
```

(4)设置调用协议规则库时的初始条件和函数入口地址，这里主要设置 ipq_call_function_struct 结构，主要实现函数如下：

```
ipoque_set_protocol_detection_bitmask2(ipoque_struct, &all);
```

以检测 DHCP 协议为例，其中 detection_bitmask 的值表示要检查的协议类型，这里设置为全 1，表示检查全部协议类型，说明如下：

```
#ifdef IPOQUE_PROTOCOL_DHCP
  if
(IPOQUE_COMPARE_PROTOCOL_TO_BITMASK(*detection_bitmask, IPOQUE_PROTOCOL_DHCP) != 0) {
      ipoque_struct->callback_buffer[a].func = ipoque_ search_dhcp_udp;
/*分析 dhcp 协议所用到的函数入口地址，函数中指出了识别 dhcp 的特征码*/
  ipoque_struct->callback_buffer[a].ipq_selection_bitmask=IPQ_SELECTION_BITMASK_PROTOCOL_UDP_WITH_PAYLOAD;
/*说明 dhcp 使用的是 UDP 协议，并且必须有负载*/
  IPOQUE_SAVE_AS_BITMASK(ipoque_struct->callback_buffer[a].detection_bitmask, IPOQUE_PROTOCOL_UNKNOWN);
/*将 ipoque_struct->callback_buffer[a].detection_bitmask 设置为未识别状态，作为判断是否调用 dhcp 函数时的一个条件*/
  IPOQUE_SAVE_AS_BITMASK(ipoque_struct->callback_buffer[a].excluded_protocol_bitmask, IPOQUE_PROTOCOL_DHCP);
/*excluded_protocol_bitmask 为排除位，将其设置为 DHCP 协议类型，和数据流 flow 的 excluded_protocol_bitmask 做按位与，如果不为 0，表示一个数据流检测过 dhcp 的识别函数，但该数据流不属于 DHCP，用来判断是否调用 dhcp 解析函数的一个条件*/
  a++;
  }
#endif
```

(5)将所有待分析协议分类划分。根据 TCP/IP 协议栈的下 4 层结构，将需要识别的协议进行分类，这样可以减少调用协议识别函数的数目，比如协议 DHCP，它采用 UDP 进行封装，就只需要调用承载在 UDP 上的协议，不需要调用所有协议识别函数，这样可以提高识别速度。

### 3.2 协议识别过程

按 TCP/IP 协议[3-4]的不同层次将报文进行解析，按由下至上进行处理，主要是对 IP、TCP、UDP 层进行分层解析，最后调用协议规则库进行应用层协议识别。

这里分析的是 Wireshark 捕获的离线数据包，当数据包传来时先解析捕获数据包时所加的帧头，记录捕获的时间，如果其报文长度符合要求，且为 Ethernet 数据包，则继续分

?1994-2015 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

析它的上层协议 IP 的数据包头。

从 IP 数据包头中解析出源地址 SRC 和目的地址 DST，在数组 osdpi_id 查找 SRC 和 DST，如果没有该 IP 地址，则将其加入数组中，用来记录协议识别过程中，所有的 IP 地址及其信息，实现函数如下：

```
src = get_id((u8 *) & iph->saddr);
dst = get_id((u8 *) & iph->daddr);
```

在数组 osdpi_flow 中查找当前的数据流，如果不存在，则加入数组中，用来唯一标识一条数据流，实现函数如下：

```
flow=get_osdpi_flow(iph, ipsize);
```

继续分析 IP 的上层协议，按 TCP、UDP、既不是 TCP 也不是 UDP 进行分类解析，针对 TCP 还需要进行流的判断，如果该数据流已经被检测且未被识别出来，并且当前这个流的数据包是进行 TCP 连接的第一次会话，则将该流的信息重置，表示一条新的数据流。实现函数为：

```
ipq_init_packet_header(ipoque_struct, packetlen);
```

接着分析数据流的连接信息，判断数据流是上行还是下行，针对 TCP 还需要设置流的发送序号 seq 和确认序号 ack，并判断是否进行了重传，实现函数为：

```
ipoque_connection_tracking(ipoque_struct);
```

将所有的中间分析结果，全部保存到 ipoque_struct 结构中，根据 packet 包的分析结果设置 ipq_selection_packet 变量，用来表示该数据包下 4 层协议类型，比如 IPv6、IPv4、TCP、UDP、其他、TCP 有无负载等信息。调用相应的协议识别函数进行协议的识别，调用协议识别函数需要判断 3 个条件：

(1)当前数据包的 ipq_selection_packet 值包含于初始化时回调函数结构对应的 ipq_selection_bitmask 的值，表示当前数据包需要调用哪些协议识别函数。

(2)当前数据流没有检查过这个协议分析函数。

(3)当前数据流没有被识别出来，或者该协议虽然被识别，但可能在该协议之上封装了其他协议，比如 1 条数据流被识别为 HTTP，但其上可能封装了迅雷，这种情况需进一步识别。

最后依次调用对应的协议分析函数，直到识别出数据流的协议类型。

### 3.3 协议规则库

不同的协议，其具体识别的方法不同，其匹配规则也不同，但大体识别步骤相同，首先通过传入的全局结构 ipoque_struct，得到数据包 packet 的信息，数据流 flow 的信息，然后调用相应的协议规则库函数进行协议的识别，其中 XXX 表示协议类型，具体函数如下：

```
ipoque_search_xxx_tcp(ipoque_struct)
{
        If(符合协议 xxx 的匹配规则)
        {
                ipoque_int_xxx_add_connection(ipoque_struct);
/*协议 XXX 被识别*/
                return；
        }
 IPOQUE_ADD_PROTOCOL_TO_BITMASK(flow->excluded_protocol_bitmask, IPOQUE_PROTOCOL_XXX);
/*添加该数据流的排除位，表示该数据流不可能时协议 XXX*/
}
```

## 4 报文解析的细节问题

### 4.1 数据流会话的有效期

对于部分协议，数据流的连接有一定的时限，超过时限，该数据流就会断开，需要重新识别，所以，要进行数据流的有效期处理。

对于有超时设置的协议，当其被识别出来后，需要记录当前源地址和目的地址的时间值，当下次该数据流到来时，得到这个数据包的时间值，判断这 2 个时间差是否大于该协议的超时值，如果大于，则需要重新识别该协议；否则，直接返回该数据流的协议类型，并重新记录源地址 src 和目的地址 dst 的时间值，如图 2 所示。

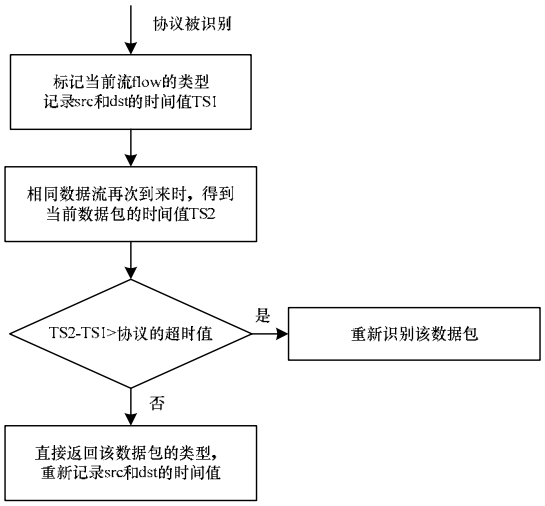


**图 2 数据流的有效期处理过程**

### 4.2 多报文处理

OpenDPI 支持多报文处理，当第一个报文到来时，提取其应用层数据进行匹配，如果只有部分匹配，但不能唯一确定一个数据流的协议类型时，则将当前的匹配结果记录到这个数据流相关信息中，等待这个数据流的下一个数据报文的到来，结合上次的匹配情况继续进行匹配，直到匹配成功或不能识别为止，如图 3 所示。

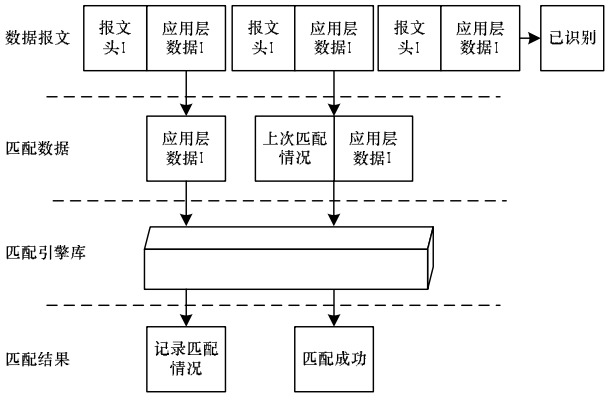


**图 3 多报文处理过程**

## 5 结束语

本文通过对 OpenDPI 源码的分析，可以将其部署在局域网中，提高网络的管理能力并节省开支。此外，DPI 行业作为相对年轻的市场，还面临着很多挑战，没有一个统一的标准，但目前各大厂家的设备中都包含了 DPI 技术，主要应用在入侵检测、网络安全、业务控制、流量分析中，成熟的软件包括思科 IPS、思科 FPM、华为 SIG(深度业务监控网关)、波音 SMIS、Sandvine、Shenick 等。OpenDPI 与这些商业化的深度报文检测系统相比，它的缺陷也是很明显的，如目前不支持加密和行为分析，没有设置对 IP 分片报文的处理功能

(下转第 103 页)

?1994-2015 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

行误差的修正。修正的消息格式如图 8 所示。

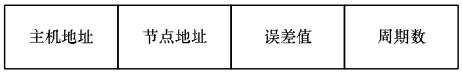


**图 8 修正的消息格式**

### 3.3 TBEC 算法的实现

硬件环境采用的是 Sun SPOT 无线传感器实验平台，该平台采用德州仪器公司的 CC2420 无线通信模块，集成了 802.15.4 无线通信协议。计时器为系统自带时钟，其精度为 ±1 ms。一套设备中有 2 个普通节点和一个基站。

软件环境采用的是 Java Me 开发框架，通过向设备中烧写 Java Me 程序，调用传感器、时钟、射频等控制接口进行实验。

在实验程序中，需要设定的变量有：基站发送的信标消息的周期 $T$(单位为 s)，为降低随机误差而设定的同步区间数量 $m$。当 $T$=5 s、$m$=30 个时，取 $N_2$ 节点的 20 个同步周期，TBEC 算法使用前后的时钟误差如图 9 所示。

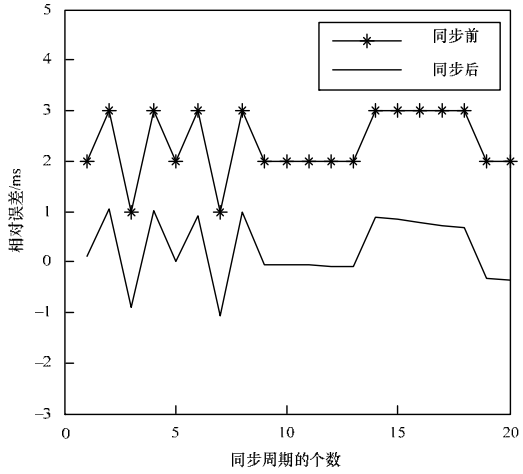


**图 9 TBEC 算法使用前后的时钟误差**

选不同的 $T$ 和 $m$ 值，节点 $N_2$ 在 30 个同步周期中，TBEC 算法使用前后误差的均值 $\mu$ 与标准差 $\sigma$。同步前后误差的均值与标准差如表 1 所示。

**表 1 同步前后误差的均值与标准差**

| 同步周期/s | 同步区间数量 | 同步前均值/ms | 同步后均值/ms | 同步前标准差/ms | 同步后标准差/ms |
|---|---|---|---|---|---|
| 5 | 50 | 2.50 | 0.78 | 0.73 | 0.70 |
| 5 | 30 | 2.50 | 0.27 | 0.73 | 0.69 |
| 30 | 50 | 4.30 | 0.16 | 0.79 | 0.82 |
| 30 | 30 | 4.30 | 0.02 | 0.79 | 0.78 |

由表 1 可知，在 TBEC 算法使用前后，节点间误差的标准差 $\sigma$ 变化不大，而误差平均值 $\mu$ 显著减小，达到较好的时间同步效果。

在 WSN 中，通常需要一个或多个基站，而基站一般会提供充足的电源和具有较强的计算能力。所以，将计算和能量的消耗集中到基站端，可以降低普通节点的能量消耗，从而提高全网的生命周期。TBEC 算法利用 WSN 的这个特点，并结合无线信道的广播特性，通过集中式管理方式将每个周期的同步消息数控制在 $1+2\times n$ 内，并达到较好的时间同步效果。但对于非集中式管理的 WSN，因为普通节点没有较强的计算能力和电源供给，所以该算法将不能够适用。

## 4 结束语

本文提出一种能量消耗较低的时间同步算法——TBEC 算法。利用大多 WSN 的集中式管理特点，并在 Sun SPOT 无线传感器网络实验平台上进行实验验证。实验结果表明，该算法能有效地降低同步过程中的能量消耗，达到较好的同步效果。

### 参考文献

[1] 任丰原, 黄海宁, 林 闯. 无线传感器网络[J]. 软件学报, 2003, 14(7): 1282-1290.

[2] Elson J, Griod L, Estrin D. Fine-grained Network Time Synchronization Using Reference Broadcasts[C]//Proc. of the 5th Symposium on Operating Systems Design and Implementation. New York, USA: [s. n.], 2002.

[3] Su Ping. Delay Measurement Time Synchronization for Wireless Sensor Networks[EB/OL]. (2010-11-21). http://sites.powercam.cc/board.php?courseID=124&f=doc&cid=7756.

[4] Ganeriwal S, Kumar R, Srivastava M B. Timing-sync Protocol for Sensor Networks[C]//Proc. of the 1st International Conference on Embedded Networked Sensor Systems. New York, USA: [s. n.], 2003.

[5] 周贤伟, 韦 炜, 覃伯平. 无线传感器网络的时间同步算法研究[J]. 传感技术学报, 2006, 19(1): 20-25.

[6] 徐朝农, 徐勇军. 无线传感器网络同步误差测量方法[J]. 计算机工程, 2011, 37(5): 115-117.

[7] 黄天戍, 陈苒菁. 石英晶振频率测试系统的研究与开发[J]. 中国仪器仪表, 2005, (10): 83-85.

[8] Ye Qing, Zhang Yuecheng, Cheng Liang. A Study on the Optimal Time Synchronization Accuracy in Wireless Sensor Networks[J]. Computer Networks, 2005, 48(4): 549-566.

(上接第 100 页)

等。此外，相对于 Internet 的应用，OpenDPI 支持的协议类型较少，对网络上的许多私有协议不能识别，因此，下一步还需要针对这些问题对系统进行改进。

### 参考文献

[1] 叶文晨, 汪 敏, 陈云寰, 等. 一种联合 DPI 和 DFI 的网络流量检测方法[J]. 计算机工程, 2011, 37(10): 102-104.

[2] OpenDPI Org. OpenDPI-manual[EB/OL]. (2011-09-04). http://opendpi.googlecode.com/files/OpenDPI-Manual.pdf.

[3] 杨忠宝, 杨宏宇. Snort 报文嗅探和报文解析实现的剖析[J]. 计算机工程, 2005, 31(13): 104-106.

[4] Stevens W R. TCP/IP 详解卷 1: 协议[M]. 范建华, 胥光辉, 张 涛, 等, 译. 北京: 机械工业出版社, 2006.

?1994-2015 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net