基于特征串的应用层协议识别

陈 亮 龚 俭 徐 选
(东南大学计算机系江苏省计算机网络技术重点实验室,南京 210096)
E-mail: lchen@njnet.edu.cn

摘 要 随着各种 P2P 协议的广泛应用以及逃避防火墙检测的需要,传统的基于常用端口识别应用层协议的方法已经出现问题。文章通过分析可用的文档和实际报文 TRACE,分别为七种应用层协议找出其实际交互过程中必须出现且出现频率最高的固定字段,并将这些固定字段作为协议的特征串来识别这七种协议。实验结果表明,相较于端口方法,使用特征串方法识别这七种应用层协议具有更高的准确性,并且时间消耗的增长不会超过 2‰。

关键词 网络流量 应用层协议识别 特征串

文章编号 1002-8331-(2006)24-0016-04 **文献标识码** A **中图分类号** TP393.08

Identification of Application-Level Protocols Using Characteristic

Chen Liang Gong Jian Xu Xuan
(Jiangsu Province Key Laboratory of Computer Networking Technology, Department of Computer Science, Southeast University, Nanjing 210096)

Abstract: Along with the emergence of many P2P protocols and the need of circumventing firewalls, traditional methods of application-level protocol identification such as using default server port become more and more inaccurate.The characteristic for each of seven application-level protocols is defined by analyzing some available documentations and packet-level traces in this paper.The characteristic of a protocol is a necessary part of actual communication, and it is more frequent to be used than any other necessary parts.These characteristics then are utilized to identify the seven protocols.The measurements show that the approach has higher accuracy than traditional port-based approach, and the time consumption increment do not exceed 2%.

Keywords: network traffic, application-level protocol identification, characteristic string

1 引言

无论是对网络规划、网络问题检测、网络使用情况报告,还是对提高网络服务质量、检测异常流量来说,流量的识别都是最基本的前提。近年来,网络流量在形式与种类上都较过去更加复杂,新的应用协议不断涌现[1]。虽然原则上可以使用在 IANA[2]中注册的端口号识别各种应用层协议,但是由于下述原因的存在,端口识别协议的方法正在越来越多的受到限制:

(1)不是所有的协议都在 IANA 中注册使用的端口。例如,BT 等 P2P 协议。

(2)有些应用程序可能使用其常用端口以外的端口,以绕过操作系统的访问限制。例如,某些没有特权的用户可能在非 80 端口上运行 WWW 服务器,因为大多数操作系统通常将 80 端口限制为只允许某些特定的用户使用。

(3)有些注册的端口号被多个应用程序所使用。例如,端口 888 同时被 accessbuider 和 CDDBP 所使用。

(4)在某些情况下,服务器的端口是动态分配的。例如,FTP 被动模式下的数据传输端口就是在控制流中协商的。

(5)由于防火墙等访问控制技术封堵了某些未授权的端口,很多协议改变为使用常用端口以绕开防火墙的封堵。例如,80 端口被很多非 Web 应用程序所使用以绕开那些不过滤 80 端口流量的防火墙。实际上,以隧道方式在 HTTP 协议上使用 IP 协议可以允许所有的应用程序通过 TCP 的 80 端口。

(6)木马和其它的网络攻击(如 DoS)所产生的大量流量也不能归结为其使用的端口所代表的协议。

为了解决端口识别协议的方法所出现的问题,近年来很多的研究工作都致力于开发新的方法来识别应用层协议。2004 年,Myung-Sup Kim 等人提出一种启发式方法识别应用层协议[3],但是这种方法归根结底还是依赖于协议既定的端口,如果协议的实现不依赖于某些固定的端口而是完全随机的选择监听端口,该方法就不能正确地给出协议名称。同样在 2004 年,Subhabrata Sen 使用基于协议签名的方法识别应用层协议[4],但是其特征串定义仅限于 P2P 协议的范围,并且往往要检查全报文以匹配多个特征串,效率较低。因此,本文的目标是为每种协议定义唯一的特征串,并将特征串的定义扩展至传统的应用层协议范围。定义特征串的原则为:选择该协议特有的,交互过程中必须出现且实际环境中出现频率最高的字段作为协议的特征串。同时,本文以 7 种国内常用的应用层协议为例,具体分析其交互过程,为其定义了能够标识各自的特征串,并通过实

基金项目:国家 973 重点基础研究发展规划项目资助(编号:2003CB314804);教育部科学技术重点研究项目(编号:105084);江苏省网络与信息安全重点实验室资助(编号:BM2003201)

作者简介:陈亮(1981-),男,南京东南大学博士生,主要研究方向为网络行为学。龚俭(1957-),男,工学博士,东南大学计算机系教授、博导,主要研究方向包括网络管理、网络安全、分布式系统等。

表 1　实验数据描述

	开始时间	结束时间	采集长度	报文总数	TCP 流数	平均速率
TRACE1	2005-09-28 00:00	2005-09-28 01:00	60Byte	1.0G	30.1M	1.38Gbps
TRACE2	2005-11-08 10:00	2005-11-08 11:00	80Byte	2.0G	48.8M	2.72Gbps

验证明,相较于端口识别协议的方法,特征串方法在准确性上要高很多,并且识别单一协议的时间消耗增长不超过 2%。

本文实验所使用的两个报文 TRACE 均来自于 CERNET 江苏省省网主干,由实验室设计构造的高速网络采包器 watcher1.1 所采集。采集器采用 Intel XEON (TM) 2.40GHz*2 的处理器和 2GB 物理内存的硬件配置,操作系统平台采用 Linux 2.4.26 内核,采集器基于 Ring-Buffer 零拷贝技术[5],能支持 3 条 GE 信道,丢包率小于 0.5%。对 TRACE 的详细描述见表 1(本文中所指的流均为双向流)。当协议分析得到多个固定字段时,TRACE1 被使用来进行字段统计,以得出使用频率最高的字段作为特征串。TRACE1 和 TRACE2 共同被使用来进行端口和特征串识别协议方法效果的对比。

2　定义协议特征串

本章将对七种常用的应用层协议进行分析,包括 BT、eDonkey、MSN、SMTP、POP3、FTP 和 HTTP。由于这七种协议都是开放协议,从而使得我们可以分析协议本身,采用如下的原则定义特征串:通过分析可用的文档找出协议交互过程中特有的且必须出现的固定字段。若固定字段唯一,则使用该字段作为协议的特征串;若存在多个固定字段,则在实际的报文 TRACE 中进行统计,找出其中出现频率最高的字段作为协议的特征串。因为所选择的特征串一定会出现在一个完整的协议交互过程中,所以该方法完全可以标识出这些协议。但是如果交互不完整,该方法可能会漏报。本章所使用的实验数据为引言中所描述的 TRACE1,下面分别对每种协议进行分析。

2.1　BitTorrent

BT 协议的交互过程分两个步骤[6]:(1)客户端使用 HTTP 协议和 Tracker 服务器交互,发布或者得到种子信息;(2)如果用户需要上传或下载文件,则使用 BT 定义的 peer 连线协议和对等客户端交互。因为文件传输占据了 BT 流量的绝大部分,所以我们的目的是识别客户端之间文件传输的流量。peer 连线协议由一次握手开始跟着不断的数据流,握手报文以字符 0x13 开始,跟着是字符串 "BitTorrent protocol"。因此可以选取 BT 握手阶段 TCP 首部之后的 '0x13BitTorren protocol' 作为识别该协议的特征串。鉴于该特征串较长,识别过程中会有较大的系统开销,故将特征串逐渐减短,在 TRACE1 中进行匹配分析,结果见表 2。

表 2　选取的特征串与相应的匹配报文数

特征串长度	特征串	匹配的报文数
4	0x13Bit	782 766
6	0x13BitTo	782 766
7	0x13BitTor	774 930
20	0x13BitTorrent protocol	774 930

由表 2 可见,匹配 20 字节特征串和 7 字节特征串的报文数相同,匹配 6 字节特征串和 4 字节特征串的报文数相同。因此可以肯定,匹配 7 字节特征串的报文为 BT 报文,但目前我们仍未能确定那些匹配 6 字节特征串而不匹配 7 字节特征串的报文属于何种应用层协议。因此本文选取 '0x13BitTor' 作为 BT 协议的特征串,长度为 7 字节。

2.2　eDonkey

和 BT 协议类似,eDonkey 也是近年新出现的 P2P 协议,交互过程同样是由两部分组成。所不同的是 eDonkey 不利用任何已存在的协议,与服务器和对等客户端的交互都使用自己定义的协议[7]。协议的消息由 6 字节的消息头以及之后的消息体构成,消息头包含了 1 字节的协议类型(0xE3 为 eDonkey,0xC5 为 eMule),4 字节的消息体长度和 1 字节的消息类型。由于只匹配 1 字节的协议类型偶然性较大,故同时对的协议类型和消息类型字段进行统计。和 BT 协议相同,我们着重考虑协议中的数据传输部分,eDonkey(和 eMule)包含消息类型 53 个,在 TRACE1 中,匹配到任一 eDonkey 命令的报文总数为 767 974,其中匹配到 E3+46 的报文(协议类型 0xE3,消息类型为 0x46,即发送数据)个数为 203 719,匹配到 C5+40(发送压缩数据)的报文个数为 96 819,两者总共占总命令数的 40%,而匹配其它任一命令的报文数不超过总命令数的 3%。这样的结果说明:从报文数的角度,相较于控制信息的流量,文件传输流量占据了 P2P 协议流量的绝大部分。更因为这是 eDonkey 传输文件过程中必须的两个消息,所以本文选取这样的字符串作为 eDonkey 协议的特征串:有效载荷的第 1 个字节为 0xE3,第 6 个字节为 0x46;或者第 1 个字节为 0xC5,第 6 个字节为 0x40,长度为 6 字节。

2.3　MSN

以下四种协议(MSN、SMTP、POP3、FTP)的消息格式均由命令和参数组成。我们首先分析协议本身,找出协议交互过程中特有的且必须出现的命令,然后在 TRACE1 中对这些命令进行统计,找出其中出现频率最高的命令作为协议的特征串。

现行的 MSNP10(MSN Protocol version 10)包括 23 个命令[8,9],其中 MSG(发送消息)命令为协议的核心。TRACE1 统计表明:匹配 "MSG" 的报文数占匹配任一命令的报文总数的 38.6%,而匹配其它任一命令的报文数在总数中所占的比重不超过 8%。因此,本文选择 'MSG' 作为 MSN 协议的特征串,长度为 3 字节。

2.4　SMTP

RFC2821 定义的 SMTP 包括了 10 个命令[10],其中 5 个命令是交互过程必须的:HELO(EHLO)、MAIL、RCPT、DATA 和 QUIT。但是 QUIT 命令也使用于 FTP、POP3 等协议中,故不作考虑。TRACE1 统计表明:匹配任一 SMTP 命令的报文总数为:354 752,其中匹配 'RCPT TO' 特征串的报文个数为 187 678,占总数的 53.0%,因此本文选取 'RCPT TO' 作为 SMTP 协议的特征串,长度为 7 字节。

2.5　POP3 和 FTP

将这两个协议放在一起讨论是因为二者有很多相同的命令,单独分析会造成误报。RFC959[11]中定义的 FTP 包含命令 33 个,RFC1939[12]中定义的 POP3 包含命令 12 个,二者有 8 个命令的名称相同,且 POP3 交互过程的必需命令全部包含在这 8 个命令中。因此,对 POP3 协议的识别不能够采用匹配命令的方式。但是 POP3 规定对所有命令的响应均必须以字符串

“+OK”(成功)或“-ERR”(失败)以及之后的原因构成。因此,我们可以选择“+OK”和“-ERR”作为 POP3 协议的特征串。

为了分析相同命令对识别 FTP 协议的影响,首先观察 FTP 命令的使用情况。TRACE1 中,匹配 FTP 命令的报文总数为 200 774,其中排前三位的命令如表 3 所示。

表 3 FTP 命令使用频率前三位

命令	匹配报文数	占总报文数的比例	宿端口使用情况(括号中的百分数代表该端口的报文在该命令报文数中所占的比例)
USER	53 023	26.4%	21:42657(80.4%);110:985(1.9%) 80:649(1.2%);>1024:8167(15.4%)
PASS	39 438	19.7%	21:33194(84.2%);110:941(2.4%) 80:399(1.0%);>1024:4610(8.7%)
QUIT	36 022	17.9%	25:23206(64.4%);21:2077(5.8%) 110:855(2.4%);>1024:9684(26.9%)

由表 3 可见,三个命令占 FTP 总命令数的 64%,并且每个命令都不仅仅只被 FTP 协议所使用(例如还被 110 端口所代表的 POP3 协议所使用)。特别是 QUIT 命令,由于 SMTP、POP3、FTP 均使用该命令作为结束命令,所以不能以其作为 FTP 的特征串。综上考虑,本文采用如下的方法识别 FTP 和 POP3 协议:使用“USER”作为 FTP 协议的特征串,长度为 4 字节;使用“+OK”和“-ERR”作为 POP3 协议的特征串,长度为 4 字节。若匹配到“USER”,将流标记为 FTP;若在该流中继续匹配到“+OK”或“-ERR”,则将流标记改为 POP3。反之,若先匹配到“+OK”或“-ERR”,将流标记为 POP3;若在该流中继续匹配到“USER”,则流标记不变,仍为 POP3。

但是,该方法只能标记 FTP 的控制流,数据流的情况较复杂:在主动模式下可以简单的将 20 端口的流标识为 FTP 数据流;而在被动模式下需要检查 PASV 命令的响应信息,其包含了服务器的 IP 地址和监听数据连接的端口号,然后用该端口号标识 FTP 数据流。

2.6 HTTP

HTTP 协议是基于请求/响应范式的[13]。服务器接到客户端的请求后,给予相应的响应信息。响应总是以协议版本号开始,当前版本下该字符串为“HTTP/1.1”,考虑到兼容其它的版本,本文采用特征串“HTTP”,长度为 4 字节。TRACE1 统计表明,匹配“HTTP”和匹配“HTTP/1.”的报文数相等。

在过去的网络环境中,HTTP 协议只为 Web 所使用,所以识别出的协议流量即代表 Web 流量。但是,近年来出现了很多完全遵照 HTTP 规范的协议,例如 Gnutella 和 Kazaa 两种 P2P 协议就完全使用 HTTP 规范进行文件的传输[4]。因此本文的方法只能识别出广义上的 HTTP 协议,即使用了 HTTP 规范的协议,而不能更进一步的将那些使用了 HTTP 规范的协议区分开来。

3 端口和特征串方法的比较

实验的数据为文章引言中所描述的 TRACE1 和 TRACE2。实验分别利用表 4 中所列出的端口号以及文章第 2 部分中定义的特征串识别相应的七种协议,统计每种协议的流数、报文数和字节数;以及同时匹配端口号和特征串的流数、报文数和字节数,记录在表 5 和表 6 中。最后比较分析两种识别方法的效率。

表 4 协议及相应端口

协议	BT	eDonkey	MSN	SMTP	POP3	FTP(control)	HTTP
端口	6881-6889 16881[6]	4661 4662[7]	1863[2]	25[2]	110[2]	21[2]	80[2]

3.1 实验结果和分析

除了由于采集的时间不同所造成的流量上的差异,表 5 和表 6 中各种方法的识别结果在流数、报文数和字节数的比例上是相近的,因此可表明这两种识别协议的方法都具有稳定性。进一步分析两表中的数据可以得出以下结论:

(1)对于 SMTP 和 POP3,两种方法识别的结果较相近,特征串识别结果较端口识别结果少的情况是由那些不完整的流(例如只有握手阶段而没有内容)所造成的。并且,两种方法识别结果的交集(同时匹配端口和特征串)和单独使用一种方法识别的结果也非常相近,这证明了端口方法和特征串方法识别的是同样的流。这样的结果说明了 SMTP 和 POP3 所使用的端

表 5 端口识别与特征串识别方法的比较(TRACE1)

协议	匹配端口			匹配特征串			同时匹配端口和特征串		
	流数	报文数	字节数	流数	报文数	字节数	流数	报文数	字节数
BitTorrent	126.4K	24.3M	17.4G	465.4K	325.8M	232.4G	5.3K	12.0M	9.1G
eDonkey	12.3K	2.8M	1.6G	5.5K	148.9M	107.5G	68	2.7M	1.5G
MSN	4.1K	269.0K	187.1M	105	411.8K	289.0M	67	4.9K	758.9K
HTTP	21.0M	204.5M	96.3G	1.2M	285.8M	195.8G	1.1M	112.7M	82.1G
SMTP	230.8K	2.6M	941.3M	214.3K	2.3M	913.3M	210.8K	2.2M	903.5M
POP3	10.5K	207.1K	35.5M	10.2K	206.8K	34.8M	10.1K	206.3K	34.7M
FTP	2.0M	44.2M	5.3G	3.3M	76.4M	13.4G	1.2M	21.6M	3.8G

表 6 端口识别与特征串识别方法的比较(TRACE2)

协议	匹配端口			匹配特征串			同时匹配端口和特征串		
	流数	报文数	字节数	流数	报文数	字节数	流数	报文数	字节数
BitTorrent	160.8K	26.5M	20.2G	925.4K	456.3M	328.1G	12.4K	14.3M	10.9G
eDonkey	24.3K	3.6M	2.5G	14.2K	256.4M	194.4G	122	3.4M	2.4G
MSN	31.1K	2.0M	1.3G	2.2K	5.3M	4.1G	1.8K	212.1K	54.5M
HTTP	18.1M	418.4M	214.4G	6.1M	668.4M	479.6G	5.9M	258.8M	186.4G
SMTP	409.3K	6.4M	2.8G	393.4K	6.1M	2.7G	387.8K	5.8M	2.6G
POP3	23.2K	409.0K	220.7M	21.9K	390.9K	211.2M	20.8K	365.9K	201.4M
FTP	777.9K	3.0M	448.7M	1.1M	37.6M	2.0G	69.6K	1.0M	63.6M

口尚未被普遍更改，也说明了特征串方法识别这两种协议的可信性。

(2)对于 FTP 和 MSN，特征串方法识别的结果较端口方法有些差异，并且二者交集很小，这表明两种方法识别出了不同的流。分别抽样检查两种方法得到的结果：在特征串识别的结果中但是不在端口识别结果中的流还包含了 FTP(或 MSN)的其它命令，因此其也均为 FTP(或 MSN)流量。然而，在端口识别结果中(除去只有握手阶段的短流)但是不在特征串识别结果中的流并不包含这些命令。由此可以看出 FTP 和 MSN 协议已经不遵守其各自的端口约定，也证明了特征串方法较端口方法识别这两种协议更为准确。

(3)对于 BT、eDonkey，特征串方法识别的结果较端口方法多一个数量级(从报文数和字节数的角度，特征串识别的结果与端口识别的结果的比例在 TRACE1 和 TRACE2 中的值分别为 15 和 70)。并且，特征串方法识别的结果更符合 P2P 协议传输大文件的特征：长流(流中的报文数多)、大报文(报文中的字节数多)。因此可以相信，特征串方法识别的结果确实属于该协议的流量。端口识别这两种协议的方法已经不可行。

(4)对于 HTTP，有两点值得注意：①单从流的数量上看，端口方法识别的数量比特征串方法多，而二者结合的方法识别的结果和特征串识别的结果较相近。抽样检查在端口识别结果中但不在特征串识别结果中的流，发现其不符合 HTTP 规范，可以确定这些流量不是 HTTP 流量。特征串方法识别 HTTP 协议流量的结果是正确的。造成这一结果的原因在文章的引言中已经说明：为了逃避放火墙的封堵，很多非 HTTP 协议的实现采用防火墙可以放行的 80 端口作为监听端口，造成了 80 端口并不被 HTTP 所专用的情况；②虽然特征串方法识别的流数较端口方法少很多，但是其识别的报文数和字节数却远高于端口方法，甚至和表中两种 P2P 协议的特征相近(长流、大报文)。如果只观察特征串识别的结果中非 80 端口的情况，这种现象更为明显(以 TRACE2 为例，流数：0.2M，报文数：409.6M，字节数：293.2G)。造成这种结果的原因我们相信是由于 Gnutella 和 Kazaa 两种完全遵照 HTTP 规范的 P2P 协议的存在。但是由于采集长度的限制，我们还不能验证这一推论。实验的结果证明了本文引言及第 2.6 节中的分析：使用 80 端口识别 HTTP 流量的方法已经不可行，而本文所提出的特征串方法也只能识别广义上的 HTTP 流量，还不能细分出各种使用 HTTP 规范或者 HTTP 隧道的协议。

综上所述，除了 SMTP 和 POP3 协议，端口方法在识别其它五种协议时已经非常不准确，本文提出的特征串方法具有更高的准确性。

3.2 实验效率比较

实验环境为引言部分所描述的 watcher1.1 采集器，效率比较见表 7 所示。

表 7 端口识别与特征串识别方法的效率比较

	TRACE1		TRACE2	
	时间消耗	内存消耗	时间消耗	内存消耗
端口方法	30min	605MB	75min	605MB
特征串方法	33min	605MB	82min	605MB

特征串方法在空间及时间消耗上均高于端口方法。特征串方法在内存消耗上的增长是用来存储各种协议的特征串和各条流的协议标志，内存消耗增长小于 1MB。而且，由于采用系统自身的内存管理，内存消耗维持在一个稳定的范围内，不随实验 TRACE 的不同而改变。时间消耗与实验所用的 TRACE 和采用的方法有关。TRACE 中报文采集长度越长，流量越大，系统所消耗的时间就越多。对于同一 TRACE，特征串方法在时间消耗上的增长主要是用来匹配特征串。因为文中所提及的协议特征串均有固定的位置和固定的长度，所以匹配这些特征串不会造成太大的时间开销。由表 7 可见，即使同时识别七种协议，特征串方法较端口方法的时间消耗增长仍在 10%范围内，而只识别一种协议的时间增长不会超过 2%。

4 总结

在当今高速大容量的 Internet 环境中，大量的应用层协议为了避免识别，逃避防火墙的检查，不使用固定的端口进行通信，这不仅包括众多近年新出现的 P2P 协议，而且包括了越来越多的传统协议。使用随机端口进行通信对流量的识别与分析造成很大的困难，因为传统的基于常用端口的流量识别方法已经不能适应于当前的 Internet 环境。

本文分析了七种应用层协议的交互过程，选择协议所特有的、交互过程中必需的且实际环境中使用频率最高的固定字段或者命令作为协议的特征字段，并使用这些特征字段识别这七种协议。为了检验基于特征串识别协议的准确性，文章对两个从 CERNET 江苏省省网主干采集的 TRACE，分别使用端口识别和特征串识别协议的方法统计属于该七种协议的流数、报文数和字节数。结果表明：STMP 和 POP3 协议的实现很好地遵守着端口的约定，两种方法识别的结果很相近。FTP、MSN、BT 和 eDonkey 协议的实现并不遵守严格的遵守默认端口的约定，特征串方法较端口方法具有更高的准确性。对于 HTTP 协议，由于存在其它众多的协议利用其 80 端口进行通信以逃避放火墙的检查，造成了端口识别协议方法的失败，特征串方法更为准确。但是另一方面，本文所提出的特征串只能识别广义上的 HTTP 协议，并不能细分出 Web、Gnutella 等完全遵照 HTTP 规范的协议。

相较于端口方法，使用特征串方法同时识别七种协议将在时间消耗上增长约 10%。随着更多的特征串被加入到系统中以识别更多的协议，时间消耗还会继续增长。但是，如果只识别单一的协议，时间消耗的增长不会超过 2%，这对于研究单一协议的行为具有重要的意义。

随着端口识别协议的方法越来越不准确，特征串识别协议的方法是一个值得继续研究的方向。如何为更多的协议定义特征串，以及当特征串较多时如何提高系统的效率，是下一步研究的重点。(收稿日期：2006 年 5 月)

参考文献

1.Subhabrata Sen, Jia Wang. Analyzing Peer-to-Peer Traffic across Large Networks[C].In: IEEE/ACM Transactions on Networking, NJ: IEEE Press, 2004: 219~232

2.IANA[S].http://www.iana.org/assignments/port-numbers

3.Myung-Sup Kim, Young J Won, James Won-Ki Hong.Application-Level Traffic Monitoring and an Analysis on IP Networks[J].ETRI Journal, 2005; 27(11): 22~42

4.Subhabrata Sen, Oliver Spatscheck, Dongmei Wang.Accurate, Scalable In-Network Identification of P2P Traffic Using Application Signatures[C].

(下转 86 页)

化元件模块，每个操作段使用一个轮函数模块，两个操作段需要 2 个轮函数模块。

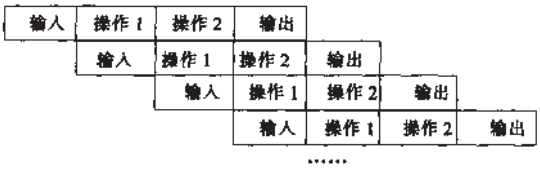

图 2　两级流水线

3.5　控制电路设计

控制电路用于加密数据在 FPGA 外围电路与 FPGA 密码算法模块之间的传输，并尽可能与算法运行过程相匹配，在提高算法硬件实现的效率和数据传输速度方面起着重要作用。控制电路通过使用状态机实现加/解密数据传输。本文设计 FPGA 数据 I/O 宽度 32bit，4 个时钟输入/输出一个 128bit 分组数据。为了匹配每一级流水线 5 轮迭代过程，在输入/输出分组数据的第 5n 个时钟时状态机执行了一轮空操作，避免因判断算法是否处理结束产生的延时，使算法进程与控制电路可以无缝连接，并节省硬件资源。在得到第一个分组结果后，每 5 个时钟就会产生一个分组结果，从外部看起来，完成一个分组仅需要 5 个时钟。流程如图 3 所示。

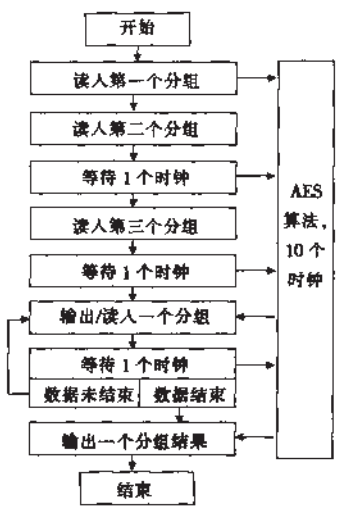

图 3　控制电路设计流程

控制电路还包括与外围电路之间各种协议信息及其它控制信号的处理，需要根据具体硬件系统设计要求而定，本文不做进一步说明。

4　实验结果与性能分析

图 4 所示为加密仿真波形，仿真最高频率 78.38MHz，完全可以满足较低全局时钟频率的要求。由于读入三个分组数据后，第一个分组数据才处理结束，因此处理第一个分组数据需要约 15 个时钟，在数据较少的情况下对提高吞吐量会有所影响，但是当数据量很大时，影响非常小。本文采用 33MHz 时钟频率，实验测试表明吞吐量可以达到 810Mbps。如果能提高全局时钟频率，则吞吐量将超过 1Gbps。

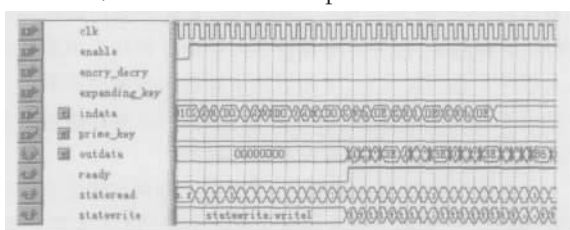

图 4　仿真波形

5　结束语

给出了 AES 算法 IP 核的快速实现和配合流水线技术的算法控制电路设计方案，根据 AES 算法的特点及硬件加密系统的特点，采用流水线技术和算法优化设计，提高了数据吞吐量，使加密算法的 FPGA 实现过程不再是传输速度的瓶颈，整个设计具有很强的实用性，运行稳定，且效果良好。对于 AES 算法分组长度和密钥长度为 192bit 和 256bit 的情况，因为执行的轮数增加，要实现流水线操作，在资源使用和吞吐量方面达到较好的效果还需要进一步优化，这也是我们今后研究的方向。（收稿日期：2006 年 3 月）

参考文献

1.Standaert et al.Efficient Implementation of Rijndael Encryption in Reconfigurable Hardware: Improvements and Design Tradeoffs[C].In: CHES 2003, LNCS 2779, 2003: 334~350
2.Saggese et al.An FPGA-Based Performance Analysis of the Unrolling, Tiling, and Pipelining of the AESAlgorithm[C].In: FPL 2003, LNCS 2778, 2003: 292~302
3.Kris Gaj, Pawel Chodowie.Comparison of the hardware performance of the AES candidates using reconfigurable hardware
4.Joan Danmen, Vincent Riijmen.AES Proposal: Rijndael.AES algorithm submission, AES home page: http://www.nist.gov/aes, 1999-09

（上接 19 页）

In: Proceedings of the 13th international conference on World Wide Web, NY: ACM Press, 2004: 512~521
5.Luca Deri, NETikos, SPA.Improving passive packet capture: beyond device polling.http://jake.unipi.it/Ring.pdf , 2004-10/2005-11
6.BitTorrent.http://www.bittorrent.com/protocol.html
7.Yoram Kulbak, Danny Bickson.The eMule Protocol Specification.http://ftp.citkit.ru/pub/sourceforge/e/em/emule/protocol_guide.pdf, 2005-01/2005-11
8.MSN Messenger Protocol.http://www.hypothetic.org/docs/msn/index.php
9.R Movva, W Lai.MSN Messenger Service 1.0 Protocol.http://www.hypothetic.org/docs/msn/sitev1/index.php, 2003-09-/2005-11
10.Simple Mail Transfer Protocol[S].RFC 2821
11.File Transfer Protocol[S].RFC 959
12.Post Office Protocol-Version 3[S].RFC 1939
13.Hypertext Transfer Protocol——HTTP/1.1[S].RFC 2616