RocketTC: 一个基于 FPGA 的高性能网络流量分类架构

付文亮[1)], 嵩天[1)], 周舟[2)]

[1)](北京理工大学计算机学院北京市海量语言信息处理与云计算应用工程技术研究中心 北京 100081)

[2)](中国科学院信息工程研究所信息内容安全技术国家工程实验室 北京 100093)

摘 要 基于深包检测技术的流量分类方法可以达到 95%以上的识别率和准确率.然而,由于计算复杂性高、存储消耗大等原因,主流软件方法只能提供百兆(线速率)处理能力,而且不能处理大量流并发的情况.本文提出一个基于深包检测技术的芯片级流量分类架构 RocketTC,通过对应用层协议特征、匹配引擎和流管理策略进行优化,使其支持万兆级数据吞吐率.RocketTC 具有两个核心模块:基于 FPGA 的流管理器和动态可重构的分类引擎阵列,前者实现万兆吞吐率下的流表管理,后者快速检测流量特征并支持动态协议特征更新特性.本文提出的分类方法使用轻量级 DPI 方法,通过缩小检测范围和特征长度进一步减少计算复杂度和存储消耗.我们使用 Xilinx Virtex-5 FPGA 对上述设计进行实现与在线流量测试,结果表明 RocketTC 可以对 92 种网络协议进行识别,准确率达到 97%,而且稳定提供 20 Gbps 线速处理能力.

关键词 架构设计;流量分类;FPGA;多级流水;部分动态可重构(PDR)
中图法分类号 TP393 DOI 号

RocketTC: A High Throughput Traffic Classification Architecture on FPGA

FU Wen-liang[1)], SONG Tian[1)], ZHOU Zhou[2)]

[1)](Beijing Engineering Research Center of Massive Language Information Processing and Cloud Computing Application, School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081)

[2)](National Engineering Laboratory for Information Security Technologies, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093)

Abstract Deep packet inspection (DPI) based traffic classification methods could achieve more than 95% accuracy and recognition rate. However, due to considerable computation and storage expenditures, existing software-based solutions could not offer sufficient processing capability for massive deployed high speed networks with massive concurrent streams. This paper proposes RocketTC, a scalable FPGA-based architecture to accelerate traffic classification while maintaining high accuracy. Specifically, we introduce two key elements to meet our goals: (1)an efficient flow management scheme using only on-chip BRAMs for storing the flow table, and (2)a parallel and pipelined classification engine array supporting partial dynamic reconfiguration (PDR). We have implemented and evaluated RocketTC on Virtex-5 FPGA based platform. Our results show a sustained throughput of over 20 Gbps for minimum packet size, and high accuracy above 97% for lightweight classifying near a hundred popular applications when regarding L7-filter as the ground truth. Additionally, it is easy for RocketTC to update for the purpose of classifying more applications.

本课题得到国家自然科学基金项目(No. 61272510, No.60803002, No.61070198, No. 61379145)资助.付文亮,男,1984年生,博士研究生,主要研究方向为网络安全、节能网络技术.E-mail:fuwenl@bit.edu.cn. 嵩天(通信作者),男,1980年生,博士,副教授,主要研究领域为网络内容安全和下一代网络体系结构.E-mail:songtian@bit.edu.cn. 周舟(通信作者),男,1983年生,博士,助理研究员,主要研究方向为网络安全及高性能网络.Email:zhouzhou@iie.ac.cn.
联系方式：13811714604, E-mail:fuwenl@gmail.com, 中国计算机学会会员

Key words architecture; classification; FPGA; multi-stage pipeline; Partial Dynamic Reconfiguration(PDR)

1 引 言

网络流量分类是网络管理、流量工程和网络安全行为分析等网络研究和应用的基础,也是网络研究的重点内容之一.同时,准确的从应用层角度了解网络也是下一代互联网设计的基础.

网络流量分类研究发展至今,衍生出多种分类方法,从不同角度对网络数据流的特征进行分析与识别.这些方法主要有:基于端口号、基于有效载荷、基于主机和基于机器学习等.

基于端口号的分类方法通过检测数据包端口号进行流量识别,实现简单、速度快、延迟低.然而,由于大量使用非标准或随机端口号通信,该方法仅能提供不足 70%的识别准确率[1].基于有效载荷的分类方法[1-6],又称深包检测方法(DPI),主要通过检测数据包中协议特征字符串进行分类.这种方法能够达到较高的识别率和准确率,但是计算复杂性高,存储开销大,且不能识别加密数据流.基于主机的分类方法[7-9]通过分析数据流的社会特征进行分类.由于不涉及有效载荷且计算复杂度不高,这种方法不仅可以识别加密流,还能提供较高的数据吞吐率.基于机器学习的方法[10-11]将网络中的数据流统计特征(如平均到达时间、数据包的个数)进行抽取,形成网络协议特征集作为流量分类的标准.其中,后两者的分类准确率远低于基于有效载荷的分类方法.

总的来看,在良好的协议特征支持下,基于有效载荷的分类方法可以提供最佳的识别率和识别准确率.此外,虽然 DPI 方法需要较高的计算和存储开销,但其实现简单、鲁棒且更符合实时性的要求.

基于深包检测的分类方法,本文设计并实现了一个基于 FPGA 的流量分类架构,其主要具有以下几个特点:

1)高吞吐率:在高速网络环境中,实时流量分类设备必须具备相应的处理能力.例如,OC-192(10Gbps)网络满负荷工作时,要求接入的实时设备每秒处理 3125 万个最小包(以太网包长 64 字节).如果在线接入设备处理能力不足,就会出现丢包等现象.RocketTC 提供数据吞吐率高达 20Gbps,满足万兆级高性能网络的要求.

2)芯片级设计:受到计算和存储资源的限制,现有涉及芯片的流量识别方法中,不是使用软硬件相结合的方式[12],就是仅支持特定应用类型数据流的硬件方法[13-15] (如:多媒体数据流).为了达到万兆级吞吐率,我们针对 FPGA 芯片的特性设计了基于片上存储器(Block RAM)的高效流表管理策略和具有部分动态可重构特性[16] (PDR)的分类引擎阵列,使用硬件设计解决流量分类问题,减少系统瓶颈.

3)轻量级 DPI 方法: 为了提高检测效率、降低资源使用率,RocketTC 采用简化的 DPI 方法,仅针对部分有效载荷(前 32 字节)进行特征检测.理论上,这种轻量级深包检测方法可覆盖 95%以上的协议特征字段[12].

4)精简协议特征集:良好的协议特征是保证 DPI 方法高准确率的关键.为了达到这个目的,我们设计了一套针对部分有效载荷(前 32 字节)的协议特征集.实验表明,在这套协议特征集的支持下,RocketTC 的分类准确率可达 97%.

5)在线更新特性:RocketTC 使用了动态可重构和菊花链式引擎配置的方式实现匹配单元和协议特征的在线更新,在整个动态更新过程中,系统始终处于正常工作状态.

本文组织如下:第二节介绍相关研究情况;第三节对 RocketTC 整体架构进行描述;第四、五节介绍流管理单元和流分类引擎阵列的详细设计;第六节实验结果及相关分析;第七节总结全文.

2 相关研究

近年来,很多研究者尝试将 FPGA 等可编程芯片引入流量分类系统以解决系统在吞吐率方面的瓶颈.罗等将 C4.5 决策树作为突破点,在 FPGA 上实现决策树快速查找,理论上大大提高了系统吞吐率[13].然而,他们并没有将算法进行硬件实现,无法对引入 FPGA 的影响做全面评估.

Canini 等人提出了一个软硬件结合方案[12].他们将流量分类系统中计算复杂性高、内存消耗大、实时性较低的协议分类模块用软件实现,实时性高的流表查找模块用硬件实现.在 NetFPGA 平台上的实验表明,系统吞吐率最高可达 8Mpps,基本达到千兆线速率的要求.然而,其软件部分仍成为系统的瓶颈,不仅增加了整个系统的延迟,还不支持大量数据流并发的情况.

W.Jiang 等设计了一个基于 FPGA 的多媒体流量分类系统[14].这个系统使用 FPGA 实现近似 K 最近邻算法以增加吞吐率.最终测试表明,在保证高准

确率的前提下,系统吞吐率可以达到约 80 Gbps(64 字节数据包).但是,这个硬件方法只能处理多媒体数据流,具有较大限制.

此外,Khan 等人提出一个基于 FPGA 的流量疏导图(traffic dispersion graphs)方法[15].测试表明,系统平均吞吐率可达 7.4Mpps,平均每个疏导图包涵 10K 个数据流.

综上可见,目前涉及芯片的流量分类方法,不是基于软硬件相结合的思想,就是仅针对某种特殊应用的硬件分类方法.目前还没有一个协议覆盖较为全面的芯片级解决方案.

3 RocketTC 架构

RocketTC 是一个芯片级高吞吐率流分类系统架构,如图 1 所示.其中,网络数据由以太网控制器送入帧聚合模块.

帧聚合模块(Frame Aggregation, FA)将以太网帧进行拆分,并以固定位宽将 IP 数据包送入包头及有效载荷提取模块(Header and Payload Extraction, HPE).HPE 提取数据包五元组,将其转发到流管理模块(Flow Management, FM)并等待处理结果.如果该数据包未被识别,则将其信息送入流分类引擎阵列(Traffic Classification Engine Array ,TCEA);否则不对该数据包做进一步处理(该包所属数据流已识别).

FM 管理数据流应用信息, 快速过滤已识别数据包.其对数据包五元组做哈希运算,并由所得哈希值索引流表.根据命中情况, FM 单元维护流表状态,并将查找结果反馈到 HPE 单元(数据流由五元组区分).这部分内容将在第四节详细描述.

TCEA 模块通过特征字符串匹配的方式对数据包有效载荷进行检测.如果识别成功,TCEA 将结果反馈给 FM 模块,并促使后者记录相关数据流信息.这部分将在第五节详细介绍.

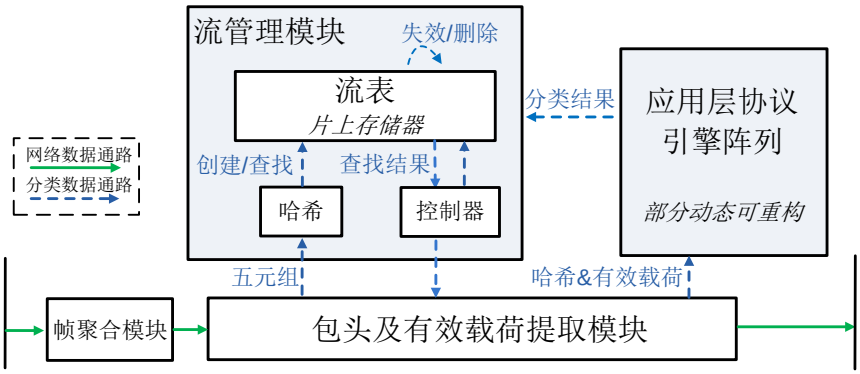

图 1. RocketTC 架构及数据通路

4 流表管理策略

流量分类结果（流信息和应用标识）以条目的形式存储在流表,其高性能组织是影响在线设备性能的主要瓶颈之一.我们设计了基于片上存储器的流表管理策略.功能上看,流表管理主要包括如下内容 1)查询流表,2)添加条目和 3)维护条目有效性.

FM 模块负责维护系统流表,储存已识别的数据流信息和分类结果.通过查询流表,RocketTC 可以快速过滤已被识别的数据包,极大的提高了系统效率.为了达到条目管理高性能,FM 模块采用片上存储器和哈希索引的方式快速定位流表条目,并通过分块的方式减少哈希冲突率.

针对成功识别的流信息和应用标识,FM 模块为其添加条目并存储到流表中.同传统的 “先添加,后识别”方法[12]不同,RocketTC 采用“先识别,后添加”的设计,不仅减少了流表数量,降低了流表维护的难度,还避免了可能针对流表的 DDOS 攻击.换句话说,系统只关心那些已识别的数据流.

当一个流表条目失效 (其标记的数据流生命周期结束),FM 模块应尽快将其删除以减少误报.传统遍历流表的方法不适用于高吞吐率的情况（以 32768 个条目,存储器存取周期 8 纳秒为例,一次遍历需要约 524 微秒,在万兆吞吐率下造成至多 5000 个数据包丢失）.因此,我们了设计一个高效的流表管理策略,其中包含两个重要内容:高性能流表数据结构和高效的条目管理机制.

4.1 流表设计

合理的数据结构是实现高速查找表的关键.FM 模块使用片上存储器(on-chip block RAM)存储流表条目,并通过哈希索引的方式进行快速查找.流表条目由部分五元组哈希值(流特征)、流类型(应用层协议编号)和时间标记组成,如图 2 所示.

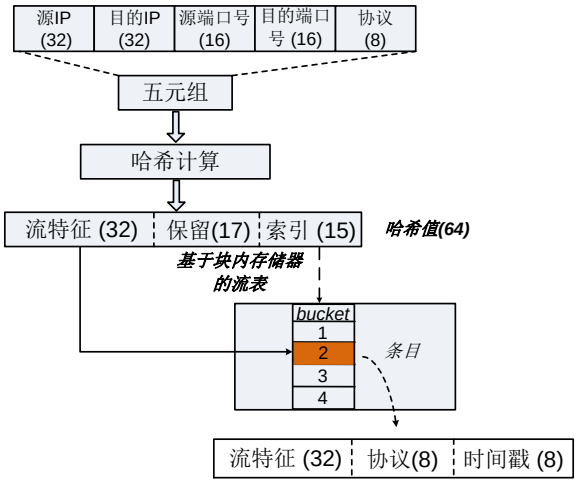

图 2. 流表结构(括号中的数字为数据位宽)

为了实现高速查找,我们使用哈希函数将一个 M 位(此处为数据包五元组的数据位宽,共 104 位)数值均匀映射到 N 位(64 位)索引值.具体到 RocketTC 中,我们使用循环冗余检验算法(CRC)作为哈希函数,因为 CRC 算法具有相对较好的唯一性[17],而且具有很高的计算性能[1].

如图 2 所示,我们将流表存储在 FPGA 的片上存储器,由五元组哈希值的低 15 位进行索引,最多可存 2^{15} 个块 (bucket).此外,为了降低冲突率,每个块可以存放 4 个条目(具有相同的哈希索引值),整个流表最多可存储 2^{17} 个条目.具体来说,每个条目存储 48 位数据,包括 32 位流特征(signature),8 位协议号(自定义)和 8 位的时间戳(由高位到低位).其中 32 位流特征为五元组哈希值的高 32 位 ,与索引流表的哈希值低 15 位并不重复.进行索引时,FM 模块将哈希索引的一个块数据(四个条目)取出进行比对,一个时钟周期得到结果.

由于采用了哈希索引,流表管理中可能产生冲突问题(两个不同的数据流得到相同的哈希值),这种情况我们称为假阳性错误.Prodanoff 等人证明[18],最好情况下,哈希冲概率突问题符合经典的生日悖论模型.基于此结论,我们由以下公式估算冲突概率:

$$P_{collision}=1-\frac{m!}{m^n(m-n)!}\approx 1-e^{-\frac{n^2}{2\times m}}$$

其中, m 是哈希值总数量, n 是已经被占用的条目数量.

表 I 不同参数下的哈希冲突率

使用率	100%	50%	25%	12.5%
冲突概率	6.1×10^{-5}	1.5×10^{-5}	3.8×10^{-6}	9.5×10^{-7}

为了更清晰的展示和评估冲突率可能产生的影响,我们将流表使用率作为参数,计算得到哈希冲突率.由表 I 所示:当流表占用率为 50%时,添加新条目的冲突概率为 1.5×10^{-5};当流表全部被占满时,冲突概率依然保持在 6.1×10^{-5}.

流表条目中,除了 32 位的流特征外,还包括 8 位协议号和 8 位时间戳.协议号标明该数据流属于哪个网络应用.由于硬件资源限制,RocketTC 支持 92 个应用,我们使用 8 位对其进行标记.条目中的时间戳记录了条目添加或最后访问的时间,是流生命周期管理机制的关键标记,在 4.2 中详细介绍.

流表查找和添加操作针对具体条目进行.数据包进入系统后,FM 模块根据其五元组哈希值索引流表.如果条目已存在,将返回已识别信号并结束查找;如果不存在,则指示 HPE 将数据包相关信息送到 TCEA 模块进行识别.TCEA 模块将识别成功的分类信息反馈给 FM 模块,并促使后者添加流表条目.

4.2 生命周期机制

由于数据流具有时效性,流管理模块必须定期将过期条目进行标记和删除(常用数据流有效期为 60 秒). 为了达到这个目的,FM 模块维护一个全局时钟,每隔一定时间递增(10 秒).由于系统资源的限制,我们将计数器位宽设为 8 位,这个颗粒度为 10 秒的全局时钟经过 2560 秒(10×2^8)循环一个周期.添加条目时,流管理模块将系统时间作为标记插入条目.当索引到条目时,系统通过比较当前系统时钟与条目建立时间的方式验证条目实效性.

流管理模块采用触发检查和定期清理两种方法维护流表实效性.系统索引到某条目时,首先判断其实效性,如果过期,则删除.这种触发式的删除方法可能会出现某些记录长时间没有被访问,一个全局时钟周期后仍有效的假阳性错误.针对这种情况,FM 模块采用了定时清空机制,在每个全局时钟周期固定时间清空流表条目,以减少假阳性错误.系

1 Xilinx, "Vertex-5 Cyclic Redundancy Check (CRC) Wizard", http://www.xilinx.com/support/documentation/ipcommunicationnetwork_errorcorrect_crcwizard.htm.

统流表共有 2^{15} 个块,清理 1 个块需要 1 个时钟周期,共需 2^{15} 个周期完成清理操作.在芯片时钟周期为 8 纳秒的情况下,整个清理操作耗时约 262 微秒.

```
1: IF GlobeTimer = 0 , THEN
2:     clear AllFlowRecord
3:     forward Hash&Payload to TCEA
4: ELSE
5:     IF FlowRecord_5-tupleHash exists THEN
6:         λ ← | Time_current - Time_timestamp |
7:         IF λ ≥ θ THEN
8:             delete FlowRecord_5-tupleHash
9:             forward Hash&Payload to TCEA
10:        ELSE
11:            update Time_timestamp
12:            report hit
13:        ENDIF
14:    ELSE
15:        forward Hash&Payload to TCEA
16:    ENDIF
17: ENDIF
```

图 3. 超时算法伪代码

FM 模块判断条目有效性和定期清空机制算法如图 3 所示.查找开始后,系统先检查全局时钟是否循环了一个周期(是否重新开始计数).如果是,则清空所有流表条目开始新循环,并将当前数据包送入识别引擎进行识别;否则进行条目查找.

系统找到相应流表条目后,计算其时间戳与全局时钟的差 λ,并与阈值 θ 比较:如果 λ 大于 θ,则表明此流表已过期,删除此子条目,并将相应数据转到 TCEA;否则返回查找成功信号.在 RocketTC 中,θ 的默认值设为 6(流表条目生命周期为 60 秒)与通常使用的数据流生命周期一致[7].

5 流分类引擎阵列

协议识别在流量分类引擎阵列（TCEA）模块完成.TCEA 模块的设计主要使用轻量级深包检测方法,具有高准确率,低延迟及高可扩展性等特点.

5.1 .轻量级分类方法

为了达到高准确率,TCEA 模块采用 DPI 方法对数据包进行分类.然而,传统 DPI 方法(如 Aho-Corasick 算法)将数据包的全部有效载荷作为检测对象,FPGA 有限的硬件资源无法承担由此带来的高计算复杂度和存储消耗.因此,我们设计了一个轻量级 DPI 方法,在保证高准确率的基础上,大幅度缩小数据检测的范围,从而在检测准确率与资源消耗之间达到平衡.

轻量级 DPI 检测方法要求数据检测更具针对性.Aceto 等对当前流行的正则表达式分类器 L7-filter 做了深入的分析发现,几乎所有成功识别的

表 II 部分轻量级协议特征

网络应用	特征字符串
HTTP	^HTTP/1.1.200
QQ2010	^0x02 0x1b 0x55 0x00
FTP	^220.*ftp
CVS	^BEGIN (AUTH|VERIFICATION|GSSAPI) REQUEST\x0a
POP3	^(\+ok |-err)
SSH	^0x53 0x53 0x48 0x2d
Tonghuashun	^fdfdfdfd

^表示特征从用户有效载荷的第一个字节开始

数据包中,协议特征字符串大多开始(99.98%)并结束(90.77%)于有效载荷的前 32 个字节[3].由此可知,轻量级 DPI 方法的准确率理论上可以达到 90%.

我们采用的轻量级分类方法具有以下几个特征:首先,轻量级分类方法仅检测数据包有效载荷的前 32 个字节;其次,模式集由针对固定位置的协议特征构成,而不是复杂的正则表达式.通过对协议规范及当前流行分类方法进行分析,我们总结了可支持 92 个协议的特征集,涵盖了包括网页浏览、p2p 下载(如电驴)、FTP、即时通讯(QQ、MSN 等)、流媒体、VOIP、网络游戏等主流网络应用,见表 II.

5.2 流分类引擎阵列

凭借 FPGA 的高并行性和部分动态可重构特性,我们设计了一个高性能、支持在线更新的 TCEA 架构,如图 4 所示,整个模块由配置与控制单元(Configure & Control ,CC)和并行排列的多条分类引

擎流水线组成.

1)配置与控制单元(CC): CC 单元主要负责两部分工作:首先,配置和管理静、动态引擎单元,保证流水线正常工作;其次,处理 CE 返回的匹配结果,并将其反馈到流管理单元.

CC 单元支持在线配置动态引擎协议特征,且在配置过程中不影响其它模块工作.此外,TCEA 的分级流水引擎只有在本单元没有匹配成功的情况下,才将数据转到下一个单元,具有固定的优先级.

当不同流水线上相同优先级的引擎单元报告匹配时,CC 单元中内置的仲裁器可以根据协议优先级做出最终裁决,保证得到最优匹配结果.

2)分类引擎(CE): CE 单元以多流水方式配置:每个引擎单元独立识别一个网络应用特征,多个 CE 单元串连起来组成流水线,多条流水并行组成 TCEA 的数据通路.当有数据进入 TCEA 模块时,CC 单元将待匹配数据同时发送到所有并行的流水线.由于仅针对部分有效载荷进行处理,分类引擎单元采用掩码加特征字符串的方式进行识别,前者标识特征位置,后者存储特征内容.

TCEA 分为静态和动态两个区域.静态区域由系统固化的 CE 单元组成,这些静态引擎不支持动态可重构技术,但可以通过读写寄存器的形式对协议特征和掩码进行修改,从而改变所识别协议特征.与静态引擎不同,动态区域的 CE 单元可以在系统工作的任何时候进行重构(重新分配资源,更有识别针对性),且在更新过程中不会影响其它流水线工作.动态可重构功能是 RocketTC 的一个重要特点,它使 RocketTC 具有在线配置芯片资源的特性.此外,如果用户配置的单元超过最大可配置单元数,TCEA 模块会按照优先级挑选用户最关心的协议.

RocketTC 系统支持 92 个网络应用的识别,分散在各个引擎单元中.如何合理的部署引擎是一个值得研究的问题.例如,一个数据流可能既属于 Gnutella 又属于 HTTP,当两者的 CE 单元同时报告匹配成果时,CC 单元根据优先级做出裁决,将高优先

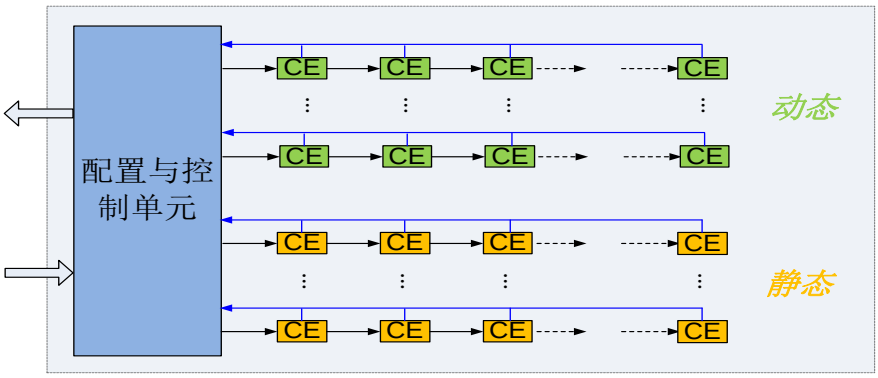

图 4 流量分类引擎阵列.

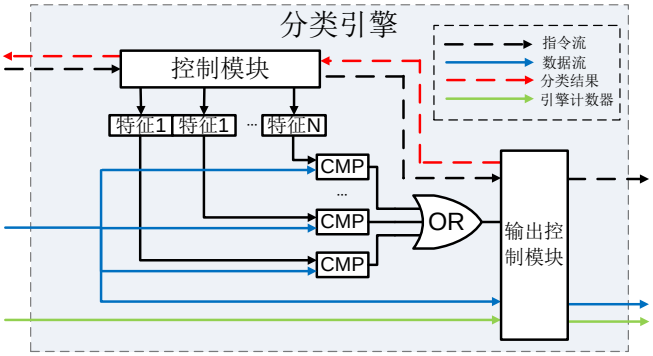

图 5 分类引擎

级的匹配结果反馈给 FM. 我们在设计 TCEA 单元时,将协议优先级与协议号相对应,简化设计.

数据通路和控制通贯穿整个 CE 流水线,如图 5 所示.数据通路负责传递数据流的哈希值和有效载荷(前 32 字节),控制通路负责传递控制信号.

如果某 CE 单元匹配成功,则将本单元对应协议号和五元组哈希值反馈到 CC 单元,且不再转发数据(到下个 CE 单元);如果匹配不成功,则将待测流数据和相关控制信号传递给本流水线下个 CE 继续检测.流水线式的设计使 CE 单元功能相对独立,从而获得高扩展性.在 CE 单元的设计中,我们使用引擎计数器判断流水线的终端.引擎计数器在数据进入流水线时被赋值,并随着流水线进程递减.当变量为 0 时,表明数据已到当前流水线的终点.

综上所述,TCEA 是一个快速、轻量级、高准确率且可扩展的结构化设计.当前实现的原型系统具有 4 条流水线,支持多达 92 个协议的识别.

6 性能评估

我们基于赛灵思 Virtex-5 FX200T FPGA 的可编程芯片平台实现了 RocketTC 原型系统.本节主要就 RocketTC 的具体性能进行评估与分析,所有测试均在实际网络环境中进行.

表 III 测试数据集

Date	Duration	Unique IP	Flows	Bytes
2010-5-10	30 minutes	17,095	251,306	27.49 G

6.1 实验设置

1) 性能指标:由于 RocketTC 属于芯片级设计,我们的考核指标不仅包括准确率等软件分类器的性能指标,还对芯片资源消耗等参数进行测量.具体性能指标如下:

a) 准确率:正确识别流量与总流量的比例.

b) 吞吐率:每秒处理数据包的数量(pps).这个参数与数据包大小相乘便得到每秒传输的数据位(bps).

c) 延迟:处理一个数据包的平均时间.

2) 数据集: 本文采用某高校网络出口采集的 30 分钟网络数据集进行评测.这个学校的有效用户约千人,主要由研究人员与学生组成.数据集包含了所有进出数据流, 具体描述入表 III 所示.

3) 方法:为了在真实环境下测试 RocketTC 的各个性能参数,我们使用 Tcpreplay 软件重放测试数据集,在线部署原型系统.本实验采用 L7-filter 软件测试结果作为流量分类结果的真值.

6.2 系统吞吐率和资源使用率

首先,我们对 RocketTC 进行吞吐率测试.在 TCEA 模块中,4 条流水线并行工作,每个流水线 23 级流水(CE),可识别 92 个应用.同时,我们还将两条流水线划入动态区域,测试系统的动态部分可重构性能.整个系统的实现情况如表 IV 所示.Virtex-5 FX200T 芯片共有 16Mb 片内随机存储器,分为 912 个块(block).我们使用了其中 452 个存储块,实现了一个可存储 13.1 万条目的流表.由于引入双口随机存储器(dual-port RAM)和多级流水等特性,RocketTC 最高工作频率为 220 MHz,系统吞吐率可达 70 Mpps(或 20 Gbps,包大小为 64 字节).

TCEA 可以通过动态重构支持更多的协议.换句话说,只要有足够的硬件资源,TCEA 可以通过动态更新的方式更改分类引擎中的协议特征.实际上,CE 单元的 LUT 使用量与特征字符串的复杂度有关,协议的特征字符串越复杂,耗费的资源就越多.

表 IV. 资源使用情况

资源	已使用	剩余	使用率
# Slices	11,179	30,720	36%
# BRAMs	452	912	49%
块内存储器(Mb)	7.6	16	47%

在资源使用方面,CE 单元所耗费的 LUT 数量从 58(5 个特征字符串)到 252(16 个特征字符串)不等,平均为 135.从这个趋势来看,如果使用百万 LUT 级别的新型 FPGA,RocketTC 可以支持最多达 7000 个网络协议,完全满足未来几年人们对流量识别系统的要求.另外,充足的 FPGA 资源还可以为 RocketTC 添加数据测量和网络应用管理等更复杂的功能.

6.3 系统延迟

在平均处理延迟的问题上,我们将 RocketTC 与其他分类方法进行对比.本实验以如下几种方法作为参考:1)纯软件解决方案,包括 L7-filter 和 PortLoad [3]和 2)结合硬件的实现方案 AtoZ [12]和基于 LSH 的多媒体分类器[14].运行 L7-filter 的计算机采用 2.83 GHz 的 Intel Core 2 Quad CPU,装配 4 GB 内存.我们以最快速度响应数据捕获以测量实际延迟.

结果归纳如表 V,在整个测试过程中, RocketTC 始终保持很小的延迟(≤450 ns),且没有明显的丢包

表 V. 处理延迟对比

分类器	平均延迟 (μsec)	方差 (μsec^2)	加速比 (与 L7-filter 比较)
L7-filter	206.92	28532.58	1.0
RocketTC	0.45	0.23	458.8
PortLoad [3]	6.99	0.88	28.6
AtoZ [12]	17	2	11.2
LSH [14]	0.11	N/A	1880.1

现象.这个结果表明,同 L7-filter 相比,RocketTC 的延迟要比前者低两个数量级.需要注意的是,在这些分类方法中,只有 LSH 的性能稍高于 RocketTC,主要有下面两个原因.首先,LSH 实现方法并没有集成网络的扩展性,适合当前复杂的高速网络环境.由于使用了深包检测的方法,RocketTC 可以提供比其他方法更佳的识别准确率和颗粒度.同时,由于使用了轻量级深包检测检测方法,RocketTC 流量分类架构更适

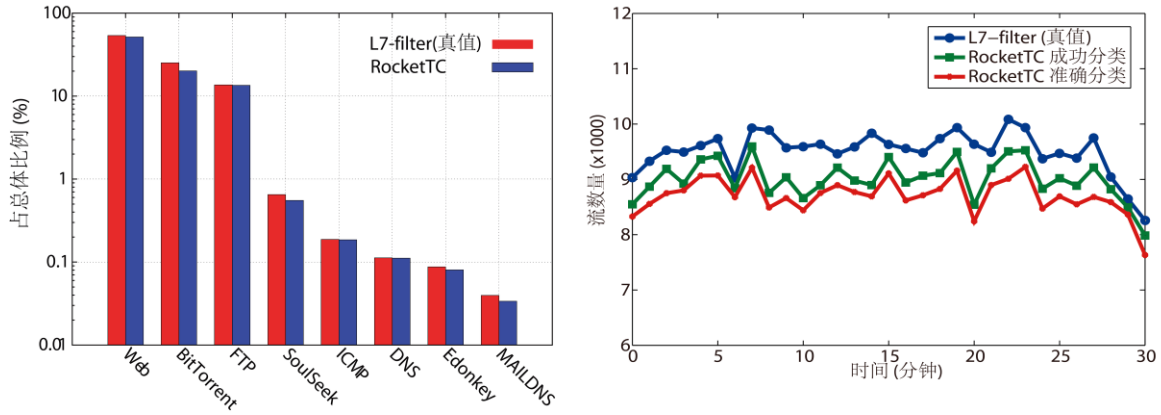

图 6 Rocket 原型系统的协议识别率（左）与准确率(右)

接口,因此减少了数据进出网口的延迟.其次,LSH 使用机器学习的方法,针对多媒体三大分类进行处理,其计算复杂度和识别协议数量远低于基于 DPI 方法的 RocketTC.此外,同 RocketTC 识别 90%以上的通用流量分类器不同,LSH 仅识别多媒体数据流.

6.4 识别准确率

为了强化对 RocketTC 的测试,我们以 1 分钟为跨度对准确率进行测量,如图 6 所示.从测试结果来看,RocketTC 具有很强的鲁棒性,成功识别超过 95%的数据流,且准确率高于 97%.由分析可知,RocketTC 不能识别或错误识别数据流的原因主要有以下几点:1.计算五元组哈希值产生哈希冲突(即不同的五元组得到相同的哈希值),流表管理模块使用冲突的哈希值索引流表导致假阳性错误;2.协议特征出现在检测范围之外,RocketTC 不能对这些特征的识别,产生伪阴性错误;3.RocketTC 所使用的 DPI 方法不能识别加密数据流.

7 结束语

针对当前数据流分类系统的不足,我们提出一个基于深包检测的芯片级架构,其在满足实时流量分类系统高吞吐率和高识别率的要求下,提供较强合硬件实现.

另外,由于使用了高效的流管理策略和高并行的分类引擎阵列,RocketTC 理论上可以提供高达 20Gbps 的数据吞吐能力.实验表明,以 L7-filter 作为参考,RocketTC 支持近百个协议,识别了超过 95%的网络流量,且识别率不低于 97%.同时我们认为,即将上市的百万 LUT 级 FPGA 必然可以为 RocketTC 的性能带来较大提升.

后续的研究可以从以下方面展开.首先,通过重构,RocketTC 可以支持更多应用协议.其次,将 RocketTC 集成到网络基础设备中,进行更贴近实际的测试.最后,我们还准备对 RocketTC 架构做修改,使其成为具有针对性的网络测量和流量管理系统.

参 考 文 献

[1] A. Moore and K. Papagiannaki. Toward the accurate adentification of network applications. International Passive and Active Measurement Workshop (PAM), Boston, MA, USA, April 2005: 41-54.

[2] S. Sen, O. Spatscheck, and D. Wang. Accurate, scalable in-network identification of P2P traffic using application signatures. International World Wide Web Conference (WWW), New York, NY, USA, May 2004: 512-521.

[3] G. Aceto, A. Dainotti, W. Donato, and A. Pescape. Portload: taking the best of two worlds in traffic classification. IEEE International Conference on Computer Communications (INFOCOM) Workshops, San Diego, CA, USA, March 2010: 1-5.

[4] P. Haffner, S. Sen, O. Spatscheck, and D. Wang. ACAS: automated construction of application signatures. SIGCOMM MineNet Workshops, Philadelphia, PA, USA, August 2005: 197-202.

[5] J. Ma, K. Levchenko, C. Kreibich, S. Savage, and G. Voelker. Unexpected means of protocol inference. ACM Internet Measurement Conference (IMC), Rio de Janeiro, Brazil, October 2006: 313-326.

[6] M. Ye, J. Wu, K. Xu, and D. Chiu. Identify P2P traffic by inspecting data transfer behavior. IPIF Networking, Aachen, Germany, May 2009, 33(10): 1141-1150.

[7] T. Karagiannis, K. Papagiannaki, and M. Faloutsos. BLINC: multilevel traffic classification in the dark. ACM SIGCOMM, Philadelphia, PA, USA, August 2005, 35(4): 229-240.

[8] M. Iliofotou, M. Faloutsos, and M. Mitzenmacher. Exploiting dynamicity in graph-based traffic analysis: techniques and applications. ACM CoNEXT, Rome, Italy, December, 2009:241-252.

[9] B. Gallagher, M. Iliofotou, T. Eliassi-Rad, and M. Faloutsos. Homophily in application layer and its usage in traffic classification. IEEE International Conference on Computer Communications (INFOCOM), San Diego, CA, USA, March 2010:1-5.

[10] T. T. Nguyen and G. Armitage. A survey of techniques for internet traffic classification using machine learning. IEEE Communications Surveys & Tutorials, 2008, vol. 10(no. 4): 56-76.

[11] H. Kim, K. Claffy, M. Fomenkov, D. Barman, M. Faloutsos, and K. Lee. Internet traffic classification demystified: myths, caveats, and the best practices. ACM CoNEXT, Madrid, December 2008, 11.

[12] M. Canini, W. Li, M. Zadnik, and A. Moore. Experience with high speed automated application identification for network management. ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS), Princeton, New Jersey, USA, October 2009:209-218.

[13] Y. Luo, K. Xiang, and S. Li. Acceleration of decision tree searching for IP traffic classification. ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS), San Jose, California, USA, November 2008:40-49.

[14] W. Jiang and M. Gokhale. Real-time classification of multimedia traffic using FPGA. IEEE International Conference on Field Programmable Logic and Applications (FPL), Milano, Italy, September 2010:56-63.

[15] F. Khan, M. Gokhale, and C. Chuah. FPGA based network traffic analysis using traffic dispersion patterns. IEEE International Conference on Field Programmable Logic and Applications (FPL), Milano, Italy, September 2010:519-524.

[16] M. Santambrogio and D. Sciuto. Design methodology for partial dynamic reconfiguration: a new degree of freedom in the HW/SW codesign. IEEE International Symposium on Parallel & Distributed Processing (IPDPS), Miami, FL, USA, April 2008: 1-8.

[17] Z. Cao, Z. Wang, and E. Zegura. Performance of hashing-based schemes for internet load balancing. IEEE International Conference on Computer Communications (INFOCOM), Tel Aviv, Israel, March 2000, 1:332-341.

[18] Z. Prodanoff and K. Christensen. Managing routing tables for URL routers in content distribution networks. International Joural of Nework Management, vol.14, March 2004:177-192.

Wenliang Fu, born in 1984, now is a PH.D candidate of Beijing Institute of Technology. E-mail: fuwenl@bit.edu.cn. Currently, his interests include green networking technologies and future Internet architecture.

Tian Song, born in 1980, PH.D, now is an associate professor of Beijing Institute of Technology, E-mail:songtian@bit.edu.cn.. Currently, his interests include future Internet architecture, network security and computer architecture.

Zhou Zhou, born in 1983, associate professor of Institute of Information Engineering, Chinese Academy of Sciences. Email: zhouzhou@iie.ac.cn. His interests include network security and high performance networking.

Background

Application layer traffic classification is one of the important components and performance bottlenecks of network management, traffic engineering and network security system. However, current traffic classification methods are incapable of offering sufficient throughout for modern high speed network environment while maintaining high classification accuracy.

Among current traffic classification methodologies, payload-based scheme can provide high accuracy according to delicately designed signatures, and offer the finest classification granularity than other methods. But, due to considerable computation and storage expenditures, existing software-based solutions could not offer sufficient processing capability for massive deployed high speed networks with massive concurrent streams.

In this paper, we propose a high throughput traffic classification architecture on FPGA, namely RocketTC, and implement it on a Virtex-5 Pro board as prototype. This architecture consists of two elaborate FPGA based components: an efficient flow management scheme and a parallel and pipelined traffic classification engine array, which are combinational optimized for high accuracy and throughput. Extensive experimental results show that RocketTC can achieve over 97%. Besides, using a single FPGA device, RocketTC is able to work at over 20 Gbps throughput. The

proposed research is aimed at providing efficient methods and hardware based architectures to achieve enough performance for application layer protocol analysis in next 5 to 10 years.

Our work is supported by National Science Foundation of China under grant No. 61272510, No.60803002, No.61070198 and No. 61379145. These research topics are focused on high performance pattern matching architectures and systems for 10-40 Gbps network environments, evoked by actual requirements of network security and audit system. The high performance pattern matching system allows us to: 1) inspect traffic data for sensitive signatures, malwares, attacks and violence messages; 2) administrate our network based on link usage offered by the traffic classification system, and develop advanced network administration strategies; 3) collect public sentiments by examining key words for further analysis. In 2010, we developed a hardware acceleration network auditing system and successfully launched it to market. Currently, we are working on FPGA based pattern matching system with DFAs method, and extending it to other applications such as traffic classification system, IDS and IPS.

This paper is focused on providing a high speed FPGA based traffic classifier for network auditing and security analyzing, which could ensure accurate analysis of network security situation for follow-up analysis of the Internet.