软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2023,34(11):5330−5354 [doi: 10.13328/j.cnki.jos.006669]
©中国科学院软件研究所版权所有.
E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

基于 FPGA 的高性能可编程数据平面研究综述*

赵 鹏[1,2], 程 光[1,2], 赵德宇[1,2]

[1](东南大学 网络空间安全学院, 江苏 南京 211189)

[2](教育部计算机网络和信息集成重点实验室 (东南大学), 江苏 南京 211189)

通信作者: 程光, E-mail: gcheng@njnet.edu.cn

摘　要: 可编程数据平面 (PDP) 一方面支持网络应用的卸载与加速, 给网络应用带来了革命性的发展机遇; 另一方面支持新协议、新服务的快速实现和部署, 促进了网络创新和演进, 是近年来网络领域的研究热点. FPGA 因其通用的计算架构、丰富的片内资源和扩展接口提供了多种可编程数据平面的具体实现, 支持更广范围的应用场景. 同时, FPGA 还为探索更通用的可编程数据平面抽象提供了可能. 因此, 基于 FPGA 的可编程数据平面受到了学术界与产业界的广泛关注. 首先分类别阐述基于 FPGA 的可编程数据平面 (F-PDP) 抽象. 接着, 介绍基于 F-PDP 快速构建网络应用的关键技术的研究进展. 之后, 介绍基于 F-PDP 的新型可编程网络设备. 此外, 从提升网络性能、构建网络测量框架以及部署网络安全应用这 3 个方面, 详细梳理近年来基于 F-PDP 的应用研究成果. 最后, 探讨 F-PDP 未来可能的研究趋势.

关键词: 可编程数据平面; 现场可编程门阵列 (FPGA); 编程抽象; 高层次综合 (HLS)

中图法分类号: TP393

中文引用格式: 赵鹏, 程光, 赵德宇. 基于FPGA的高性能可编程数据平面研究综述. 软件学报, 2023, 34(11): 5330–5354. http://www.jos.org.cn/1000-9825/6669.htm

英文引用格式: Zhao P, Cheng G, Zhao DY. Survey on FPGA-based High-performance Programmable Data Plane. Ruan Jian Xue Bao/Journal of Software, 2023, 34(11): 5330–5354 (in Chinese). http://www.jos.org.cn/1000-9825/6669.htm

Survey on FPGA-based High-performance Programmable Data Plane

ZHAO Peng[1,2], CHENG Guang[1,2], ZHAO De-Yu[1,2]

[1](School of Cyber Science and Engineering, Southeast University, Nanjing 211189, China)

[2](Key Laboratory of Computer Network and Information Integration (Southeast University), Ministry of Education, Nanjing 211189, China)

Abstract: The programmable data plane (PDP), allowing offloading and accelerating network applications, creates revolutionary development opportunities for such applications. Also, it promotes the innovation and evolution of the network by supporting the rapid implementation and deployment of new protocols and services. It has thus been a research hotspot in the field of the network in recent years. With its general computing architecture, rich on-chip resources and extended interfaces, field-programmable gate array (FPGA) provides a variety of implementations of PDP for a wider range of application scenarios. It also offers the possibility to explore more general PDP abstraction. Therefore, FPGA-based PDP (F-PDP) has been widely concerned by the academic and industrial communities. In this study, F-PDP abstraction is described by category. Then, the research progress of key technologies for building network applications with F-PDP is outlined, and programmable network devices based on F-PDP are presented. After that, the application research based on F-PDP in recent years is reviewed in detail from three aspects: improving network performance, building a network measurement framework, and deploying network security applications. Finally, the possible future research trends of F-PDP are discussed.

Key words: programmable data plane (PDP); field-programmable gate array (FPGA); programming abstraction; high-level synthesis (HLS)

* 基金项目: 国家重点研发计划 (2018YFB1800602)
收稿时间: 2021-06-03; 修改时间: 2021-11-24; 采用时间: 2022-03-11; jos 在线出版时间: 2023-04-04
CNKI 网络首发时间: 2023-04-06

© 中国科学院软件研究所 http://www.jos.org.cn

随着互联网带宽的飞速增长和新协议、新服务的不断涌现, 网络设备 (比如路由器、交换机、网卡等) 在满足线速处理需求的同时应具备可编程性[1].

传统的商用网络设备虽然能支持线速处理, 但因为依赖于专用硬件和封闭系统, 业务升级和新服务的部署只能求助于设备供应商; 且控制逻辑和转发逻辑紧密耦合, 所以编程困难, 可扩展性差, 导致新功能、新协议实现周期长, 无法在生产网络中实现快速部署与测试, 阻碍了网络创新和演进. 此外, 传统网络设备功能固定, 通常包含一个大的协议超集以适应不同的应用场景, 但过多的协议并存, 一方面增加了处理逻辑的复杂度, 另一方面消耗了更多的硬件资源, 进而增加了开发和制造成本.

为了解决上述问题, 早在 2003 年, 由 IETF 提出的 ForCES 第 1 次阐述了转控分离的思想[2,3]. ForCES 将网元分为控制件和转发件, 并通过 ForCES 协议进行交互. 转发件由多个逻辑功能块 (logical functional block, LFB) 及其互连组成, 每个 LFB 实现单个包处理功能, LFB 之间的互连表示了数据包处理路径. LFB 的可复用性和可定制化使得 ForCES 模型非常灵活和强大, 但可惜的是, 由于缺少开源社区和产业界推动, ForCES 的应用并没有达到预期的水平.

受 ForCES 启发, Casado 等人[4]提出了一种新的企业网架构, 该架构包含一个集中控制器和一组简单的 Ethane 交换机. 集中控制器为所有数据包定义转发策略并自动下发给 Ethane 交换机, 实现高效的网络管控; Ethane 交换机包含一个到集中控制器的安全通道还有一个简单的流表. 在此基础上, McKeown 等人对 Ethane 交换机流表做了进一步抽象, 增加了交换机和控制器之间的标准通信协议 OpenFlow[5], 进而提出了软件定义网络 (software-defined networking, SDN) 概念[6].

SDN 使用开放的南向 API 而不是封闭的专用接口实现转控分离, 促进了控制器和交换机的并行演进. 控制平面提供了一个开放的基于通用操作系统的编程环境, 实现了用户可编程. 同时, 控制软件使用南向 API 对数据平面进行编程, 可实现新服务在生产网络中的快速部署, 促进网络演化和创新.

然而, SDN 的可编程性主要在于控制平面, 数据平面的可编程性是有限的, 这是由于 OpenFlow 是协议相关的, 控制平面与数据平面在协议语义层进行通信. 因此, OpenFlow 只能不断发布新版本规范, 增加新的协议原语以支持新协议. 随着网络的发展, 所需要的特定于协议的指令会爆炸式地增加. 这种被动式演进方式将导致 OpenFlow 越发臃肿, 增加了交换机设计复杂度. 除了这种内在复杂性之外, 每次发布新版本都可能出现向后兼容性问题, 即需要重新编写包处理逻辑, 甚至需要重新设计交换机的硬件[7].

意识到 OpenFlow 协议局限性, Bosshart 等人提出了可编程数据平面抽象可重构匹配-动作表 (reconfigurable match-action table, RMT), 充分解放了网络处理中数据平面的编程能力[8]. 同时, 设计并实现了可编程协议无关包处理语言 P4, 并开源了相关的语言规范、开发工具链以及项目代码, 不断努力推动生态完善, 这一开创性成果受到了工业界和学术界的青睐[9].

然而, RMT 抽象在功能上存在局限性, 比如, 对有状态网络处理支持有限, 无法描述队列调度, 不支持对报文载荷进行操作等. 此外, 可编程专用集成电路 (application specific integrated circuits, ASIC) 的计算和存储资源是有限的, 且很难扩展, 这进一步限制了实践中能够支持的数据平面功能.

纯软件实现的数据平面提供了最高的可编程性和灵活性, 可以实现更加完善的数据平面功能, 但性能不足. 现有研究和实现通常通过采用多核 CPU 和服务器集群来提高网络处理性能, 但前者受限于“功耗墙”, 后者会增加集群拓扑, 主机间通信的设计难度, 增加资产成本和运营成本[1].

现场可编程门阵列 (field-programmable gate array, FPGA) 在软件和 ASIC 之间提供了折中方案. 首先, 相对于软件解决方案, FPGA 能够提供更高的性能. 一方面, FPGA 内置了大量的并行性, 理论上可提供数千个并行运行的“核”[1]. 不同于 CPU 的单指令多数据流 (single instruction multiple data, SIMD) 执行模型, 在 FPGA 中并行处理的数据不需要执行相同的操作, 从而避免了不必要的资源浪费, 消除了额外延迟. 因此, FPGA 的数据并行性可获得更高的加速比. 另一方面, FPGA 功能在配置时就已经确定, 能提供确定时延, 相对于依赖复杂指令集的 CPU, FPGA 可以通过定制数据通路, 在没有指令负载开销的情况下获得更高的流水线并行性. 鉴于此, FPGA 常用于加

© 中国科学院软件研究所 http://www.jos.org.cn

速网络应用. 其次, 相对于可编程 ASIC, FPGA 拥有更高的灵活性和可扩展性. 一方面, FPGA 属于通用计算架构, 可以在其上探索和实现比 RMT 更加完善的数据平面抽象, 以支持更多的数据平面功能卸载. 另一方面, FPGA 拥有丰富的片内计算和存储资源, 并且提供了丰富的接口用于片外扩展, 相对于可编程 ASIC, 资源受限的问题得到了缓解. 最后, FPGA 可以很方便地与 CPU 构建异构系统, CPU 用作数据平面的补充, 处理复杂数据平面功能, 进一步提高数据平面的可编程性. 因此, FPGA 在可编程数据平面的设计, 实现和应用方面潜力巨大.

已有部分学者进行了相关技术的分析和研究, 但从公开发表的论文和资料来看, 国内外对可编程数据平面的研究主要集中于 P4[9−11]. 由上所述, FPGA 在保证网络处理性能的同时给可编程数据平面带来了更多的实现可能, 而本文主要针对基于 FPGA 的高性能可编程数据平面领域进行梳理, 分析和总结.

图 1 给出了基于 FPGA 的可编程数据平面研究框架, 也指导本文的行文架构. 本文第 1 节详细介绍 3 类基于 FPGA 的可编程数据平面抽象. 第 2 节从高级语言与高层次综合工具、软硬件协同开发技术以及多数据平面切换与共存这 3 个方面对基于 FPGA 快速构建数据平面功能所涉及的关键技术进行详细阐述和分析. 第 3 节介绍基于 F-PDP 的新型网络设备. 第 4 节系统地梳理基于 F-PDP 的网络应用. 第 5 节对未来研究方向进行展望. 第 6 节对全文工作进行总结.

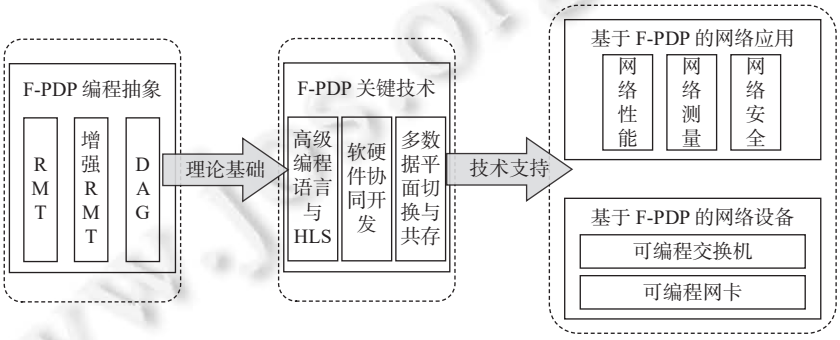


图 1 基于 FPGA 的可编程数据平面研究框架

1 基于 FPGA 的可编程数据平面抽象

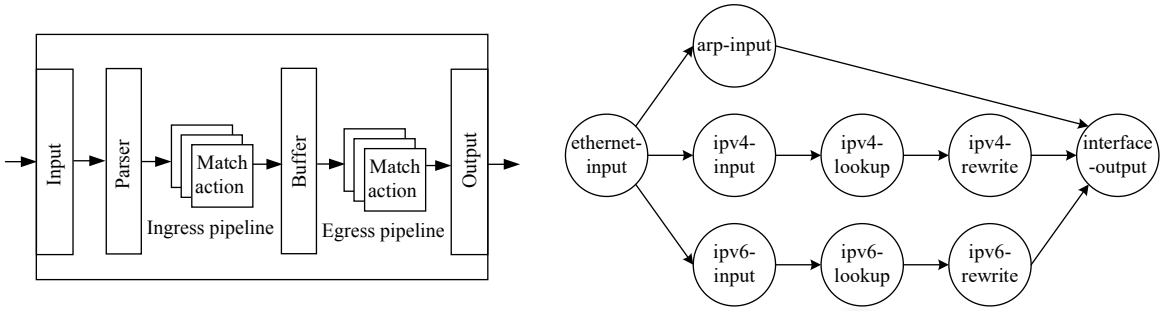
编程抽象允许程序员以一种对应用领域来说很自然的方式来描述功能, 然后通过自动化工具映射到目标平台上[12]. 好的数据平面抽象和编程模型一方面隐藏了底层硬件细节, 另一方面能够高度契合和充分描述数据包在数据平面上的处理流程. 近年来大量的研究致力于构建和完善数据平面抽象, 所提出的方案可分为 3 类, 即 RMT、增强 RMT (E-RMT) 以及有向无环图 (directed acyclic graph, DAG).

1.1 RMT 抽象

为了充分解放网络处理中数据平面的编程能力, Bosshart 等人提出了 RMT 编程模型, 并基于此模型提出了可编程协议无关包处理语言 P4. P4 提供了更高层次的抽象来定义协议头、包解析以及包处理逻辑. 具体而言, 如图 2(a) 所示, P4 通过有限状态机实现包解析, 通过级联多个匹配-动作表来定制包处理功能. P4 具有可重配置性, 可以根据应用场景配置不同的网络功能, 无需实现一个协议超集. P4 具有协议无关性, 结合可重配置性, 可以快速构建和部署新协议, 无需购买新的网络设备. P4 具有目标平台无关性, 这一特性一方面使得软件开发人员不用关注底层架构, 减轻了开发人员负担; 另一方面支持 P4 程序的跨平台无缝移植, 提高了生产效率[9].

然而, RMT 流水线中各个阶段的内存由对应阶段独占, 无法被其他阶段回收并使用, 这导致了 RMT 模型资源利用率低, 尤其在匹配和动作不均衡的情况下. 鉴于此, Chole 等人[13]提出 dRMT 模型, 通过内存和计算资源的非集计化形成资源池, 被流水线中所有阶段所共享, 提供了比 RMT 模型更高的灵活性, 提高了资源利用率, 但目前该模型还没有被设备制造商所采用.

© 中国科学院软件研究所 http://www.jos.org.cn



(a) RMT 抽象 (b) DAG 抽象

图 2 基于 FPGA 的数据平面抽象

1.2 增强 RMT 抽象

Sivaraman 等人[14]给出了一个用 P4 表示的数据中心交换机数据平面转发行为的实例, 构建实例的过程确定了 P4 的优点, 指出了一些不足并建议了未来可能的进化路径, 其中一些已经实现, 例如, P4 在其新版本中实现了语言定义和目标体系结构模型分离, 但仍然存在局限性, 比如, 无法在线编程解析器, 缺乏对有状态网络处理和通用队列调度模型的支持等. 近年来, 大量的研究基于 FPGA 硬件结构实现增强的 RMT 编程抽象.

1.2.1 在线可编程解析器

RMT 抽象通过控制器编程匹配-动作表来定制网络功能, 快速方便, 但若需要支持新协议, 则需要重新编写解析器逻辑并编译生成 bit 流文件, 最后下载到 FPGA 硬件上, 无法支持新协议的运行时加载. 鉴于此, Song[15]提出了一种称为协议无关转发 (protocol oblivious forwarding, POF) 的硬件抽象, POF 通过<*offset, length*> 二元组定义字段, 其中 *offset* 表示字段在数据包中的起始位置, *length* 表示其长度, 这使得 POF 能够在不参考特定协议的情况下定位数据包中的任意字段, POF 还定义了一组协议无关转发指令集 (POF-FIS) 用于构建表流水线来处理数据包. 与 RMT 抽象不同, POF 通过流表构建实现数据包解析, 进而实现了新协议的在线编程和运行时切换. 但 POF 的抽象层次较低, 增加了程序员的编程负担.

1.2.2 有状态网络处理

RMT 数据平面编程模型对有状态网络处理的支持有限, 为了实现复杂的有状态网络功能, 通常需要将数据包转发给控制器处理, 而交换机和控制器之间的传输通道存在性能瓶颈, 这不仅增加了处理延迟, 还直接限制了系统吞吐量. 为了减少控制器的参与, 一个行之有效的方法就是在数据平面编程模型中引入基于扩展有限状态机 (extended finite state machines, XFSM) 的抽象, 将相关工作从控制器卸载到交换机来提高系统的性能和可伸缩性.

OpenState[16]将有状态网络处理抽象为 XFSM, 作为 RMT 抽象的超集. 不同于 RMT 抽象, OpenState 拥有状态表和 XFSM 表这 2 种类型的查找表, 它们一起协作共同实现流状态的保存和更新. OpenState 允许在转发设备中实现多个有状态任务, 而不会增加控制平面的复杂性和开销. 然而, OpenState 只支持简单的 Mealy 状态机, 这严重限制了能够部署的有状态应用程序类型.

基于类似的思想, Moshref 等人[17]于同期并行开发了一个新的 SDN 交换机原语 FAST. 相对于 OpenState, FAST 对查找表做了一些优化, 包括: (1) 使用状态机筛选器将一个庞大的状态表分解为多个小的状态表; (2) 使用 Hash 表作为状态表; (3) 将 XFSM 表解耦为状态转移表和动作表, 提高了表查找和更新的效率, 提供了更灵活的可编程性. 此外, FAST 引入了运行在控制器的交换机代理, 一方面可以实现状态机的被动安装, 另一方面可以协调完成全局任务.

Bianchi 等人[18]基于 OpenState 提出了 OPP, 通过引入寄存器实现了对完全 XFSM 的支持, 显著地扩展了可以在设备上编程的应用程序的种类, 但这需要对硬件设计进行更多的扩展, 该工作设计了一个基于 FPGA 的 OPP 架构原型. 然而, OPP 的抽象层次非常低, 实践中指定了一种直接配置硬件的机器语言, 对于开发人员并不友好.

Pontarelli 等人[19]提出了一个用于在硬件中构建有状态包处理功能的开放抽象 FlowBlaze, 该抽象同样基于

© 中国科学院软件研究所 http://www.jos.org.cn

XFSM, 并引入了流状态的显式定义以支持流级并行性. 相对于之前的工作, FlowBlaze 给出了状态一致性访问模型, 探讨了 XFSM 与 RMT 集成的相关问题, 并详细阐述了 XFSM 流水线的实现细节. 作者在 NetFPGA SmartNIC 和 mSwitch 上实现了不同版本的 FlowBlaze, 相关代码已开源.

除了上述基于 XFSM 的解决方案, Sivaraman 等人[20]提出了 Banzai 模型, 用自定义的原子操作对匹配-动作表中的“动作”进行建模. 原子操作增加了对局部状态的支持, 同时能确保在单周期内完成以支持流水线并行性. 同时开发了一种类 C 的命令式编程语言 Domino, 用于表示数据平面算法. Domino 引入了包事务的概念, 即一个顺序执行的包处理代码块. 程序员只需要使用 Domino 编写包事务, Domino 编译器会自动将其转化为 Banzai 流水线.

Hang 等人[21]在 Domino 的基础上提出了改进的领域特定语言 E-Domino, E-Domino 扩展了 Domino 的语法, 扩充了包事务集, 支持更多的外部函数等. 基于 Domino 的解决方案需要原子操作在单周期内完成, 这限制了能够支持的有状态处理函数的种类. Cascone 等人[22]经过研究发现, 在特定工作负载下, 包的有状态处理时间预算可以达到最新解决方案的 30 倍, 而不会影响整体转发性能, 但该文没有详细说明哪些有状态处理可以通过额外的时间预算来实现.

1.2.3 通用队列调度模型

网络交换设备中通常会配备大容量队列, 用于吸收突发流量, 并通过精心设计的排队和调度算法实现主动队列管理 (active queuing management, AQM), 以达到缓解拥塞, 降低端到端延迟, 提高链路利用率等目的.

目前, RMT 抽象并不支持排队和调度算法的可编程性, 这些通常以功能固定的外部组件的方式实现, 主要原因是没有一个通用的调度器或者编程抽象适用于所有的排队和调度算法. 文献 [23] 明确了现有的排队和调度机制存在着双向循环偏好, 表明了寻求一个最优或者通用方案是徒劳的. 作者认为可行的途径是仔细扩展 SDN 来控制快速路径的调度和排队行为, 并建议为交换机增加一个小型 FPGA 用于实现相关算法, 实验结果证明了该建议的可行性和经济性, 但该方法只是将功能固定的外部组件替换为可重构的 FPGA, 并没有从编程模型上解决问题.

Mittal 等人[24]首先从理论上证明了经典的最短空闲时间优先 (least slack time first, LSTF) 模型对于包出队时间预先已知的调度算法来说是通用的, 其次从经验上证明了 LSTF 可以很好地再现各种调度算法, 最后评估发现 LSTF 在每个性能指标上的表现都与最新的技术相当. 然而, LSTF 认为交换机功能是固定的, 不能编程来修改数据包字段, 并且只维护一个优先级队列, 因此, 其无法模拟如下排队和调度算法, 包括 (1) 数据包出队时间很难预先知道的调度算法; (2) 需要维护和更新交换机状态的调度算法以及 (3) 分层包调度算法.

Sivaraman 等人[25]分析了 LSTF 模型的限制, 基于许多调度算法在数据包入队的时候就能明确包调度的顺序和时间这一发现, 提出 PIFO (push-in-first-out) 队列抽象, 第 1 次在硬件上实现了可编程包调度. 该调度器虽然能实现更广泛的调度算法, 包括分层调度算法, 但仍然有其局限性, 比如, 无法随意更改同一个流中所有包的调度顺序, 无法实现输出流量整形等.

PIFO 的局限性主要来源于其设计基础, 即在入队时确定数据包的优先级, 该优先级固定不变, 这不足以支持一类广泛的包调度算法, 这类算法要求流包的优先级会在流的生命周期中发生变化. 鉴于此, Shrivastav 等人[26]提出了一种 PIFO 原语的推广, 即 PIEO (push-in-extract-out), 它与 PIFO 一样, 维护一个有序的元素列表; 但与 PIFO 不同的是, PIFO 只允许从列表头部出列, 而 PIEO 在队列出口端执行基于谓词的过滤操作, 允许从队列的任意位置出列. 基于 PIEO, 论文中提出了一个快速、可扩展的调度器硬件设计方案, 并在 FPGA 上实现了原型. 总的来说, PIEO 调度器具备比 PIFO 更强的表述能力, 其可伸缩性也高出 30 倍, 但是谓词函数复杂度的选择往往需要在表述能力和可伸缩性之间做权衡, 同时受到包调度器时间开销和内存占用实际约束的限制.

作为硬件原语, PIFO 和 PIEO 都是通过提供线速运行的优先级队列的抽象来支持可编程包调度, 但它们目前仅处于硬件设计阶段, 且只支持大约 1 000 条流. Alcoz 等人[27]提出一种能够在现有可编程数据平面中实现的自适应包调度器 SP-PIFO, 通过动态调整包优先级和严格优先级队列之间的映射, 最大程度模拟理想 PIFO 的行为. Sharm 等人[28]重新审视“日历队列”抽象, 该抽象能够动态更改数据包优先级, 因此, 可以有效实现大多数调度算法. 基于此, 他们提出一种灵活的包调度器, 即可编程日历队列 (programmable calendar queue, PCQ), 能够在当今可编程的线速交换机上实现. SP-PIFO 和 PCQ 均通过引入粗粒度的队列优先级提高了调度器的可伸缩性.

© 中国科学院软件研究所 http://www.jos.org.cn

1.3 DAG 抽象

Kohler 等人[29]于 1999 年提出经典的基于 DAG 模型的模块化构建方法 Click, 如图 2(b) 所示, 有向图的顶点称为元素, 代表数据包处理模块, 比如包解析、分类、调度、排队等; 有向边称为连接, 代表数据包处理的可能路径. Click 一方面提供了一个与网络处理高度契合的编程抽象, 另一方面通过模块化实现代码复用, 提高了生产效率. 近年来, 大量研究试图利用 Click 简化 FPGA 开发. Cliff[30]是第 1 个尝试将 Click 软件路由器移植到 FPGA 平台上的工作. Cliff 方法的核心思想是提供了一个适用于 FPGA 平台上 Click 元素之间通信的标准协议, 并使用一个简单的, 用户可拓展的有限状态机实现. CUSP[31]对 Cliff 进行了改进, 主要包括: (1) 简化了硬件模块的互连, 提高通信效率; (2) 支持块内元素并行执行, 提高系统吞吐量. 实验结果表明, CUSP 的性能可以达到 Cliff 的 2 倍.

与上述方案只关注 Click 的硬件映射不同, Chimpp[32]提供了基于 Click 软件元素和 Chimpp 硬件元素的软硬件协同开发框架. Click 软件元素用于实现控制面功能, 或者作为数据面的扩展用于实现一些复杂的数据处理功能. 相比于 Cliff 和 CUSP 元素采用固定端口的方式, Chimpp 采用 XML 语法定义元素及其端口, 更加灵活. 此外, Chimpp 还提供了软硬件协同仿真环境, 提高了功能验证效率.

Cliff、CUSP 以及 Chimpp 均需要用硬件描述语言 Verilog/VHDL 重新开发元素, 编程困难, 工作量大. 为了充分利用已有的 Click 元素, Nikander 等人[33]第 1 次尝试将 Click 路由器的 C++代码直接编译成适用于 FPGA 的 RTL 代码. 首先, 采用 LLVM 工具箱将原始 C++代码编译成 LLVM IR (intermediate representation, 中间表示), 并进行一系列针对代码复杂度、冗余、并行性等方面的优化. 其次, 使用 LLVM 工具箱将优化后的 LLVM IR 编译成可综合的 C 代码. 最后, 通过商用高层次综合工具 (HLS) 综合生成 RTL 代码. 然而, 商用 HLS 只支持 C/C++语言的部分特性, 导致 Nikander 等人提供的工具链只能成功编译很少一部分简单的 Click 元素, 效果并不理想. 在后续的工作中, Rinta-Aho 等人[34]选用 AHIR 代替商用 HLS, AHIR 支持更多的 C/C++语言特性且开源可修改. 同时, AHIR 能够接收 LLVM IR 作为输入, 简化了编译流程. 然而, 文献 [33,34] 的方案存在性能瓶颈, 且不支持 CPU/FPGA 联合处理.

针对上述不足, Li 等人[35]提出了 FPGA 网络应用编程框架 ClickNP. 一方面, ClickNP 改进了 Click 编程抽象, 使其更适合 FPGA 实现, 并采用一种扩展的 C 语言进行元素开发, 减少了工作量; 另一方面, ClickNP 通过减少内存依赖, 平衡各级流水等获得更高的流水线并行性, 提高系统吞吐量. 同时, ClickNP 通过设计高吞吐量、低延迟的 PCIe I/O 通道来支 CPU/FPGA 联合处理. 文献的最后评估了基于 ClickNP 实现的几种常见的网络功能, 与最先进的软件实现相比, 吞吐量提高了 10 倍, 延迟降低了 10 倍.

Tian 等人[36]分析 RMT 抽象的不足, 基于 Click 编程的思想提出一个可扩展的数据平面抽象协议 OpenFunction, 用于实现功能复杂的中间件. OpenFunction 扩展了元素类以支持数据包操作、流状态管理以及事件生成, 并引入软件定义的概念实现复杂功能的组装. 该文分别在软件和 FPGA 硬件平台上构建原型系统并进行实验评估, 结果表明 OpenFunction 抽象实现的中间件功能具有较高的性能和平台无关性.

后文表 1 对 3 类基于 FPGA 的可编程数据平面抽象进行了比较分析, 其中 RMT 抽象使用领域特定语言 P4 编程, 编程最简单; 且开发工具链完善, 社区活跃, 受学术界和产业界青睐并被广泛使用. 基于 FPGA 的 RMT 实现主要用作算法验证平台, 或者用于实现可编程 ASIC 因资源受限而无法实现的功能等. 增强 RMT 抽象从在线可编程解析器, 有状态网络处理以及通用队列调度这 3 个方面对 RMT 抽象的表述能力进行补充和完善, FPGA 常用于新的增强 RMT 抽象的探索和原型实现. 基于 DAG 的数据平面抽象限制少, 表述能力最强, 可支持加解密, 载荷操作等. 同时, 通过 FPGA 的动态部分重构 (dynamic partial reconfiguration, DPR) 技术[37]实现功能的动态加载/卸载, 提供了最高的灵活性, 但编程工作量也是最大的.

2 基于 F-PDP 的关键技术研究进展

基于 FPGA 的数据平面抽象给基于 F-PDP 构建网络功能提供了理论基础, 但要快速实现高性能的网络应用还需要解决一些实际问题. 接下来, 本文将从高级编程语言与高层次综合工具、软硬件协同开发技术以及多数据平面切换与共存这 3 方面做详细介绍.

© 中国科学院软件研究所 http://www.jos.org.cn

表 1 PDP 抽象比较

抽象类型		相关技术	优势	不足
RMT		RMT[8], P4[9]	编程简单, 生态成熟	表述能力不足
增强RMT	在线可编程解析器	POF[15]	通过流表构建实现数据包解析, 使得包解析器具备在线编程能力	抽象层次较低, 增加了程序员的编程负担
	有状态网络处理	XFSM[16−19]	有状态网络处理抽象为XFSM, 增加状态表, 容易与RMT模型集成	匹配表资源开销大, 表查找和表更新存在性能瓶颈
		Domino[20,21]	通过自定义原子操作实现局部状态的存储和更新, 编程简单	原子操作需在单周期内完成, 限制了能够支持的有状态处理函数的种类
	通用队列调度	LSTF[24]	以单一调度算法尽可能实现通用性, 性能与各算法最先进方案相当	只支持一类包出队时间预先已知的调度算法
		PIFO[25], PIEO[26]	设计全新的队列抽象, 支持更广泛的调度算法的线速运行	仅处于硬件设计阶段,且只支持大约1000条流
		SP-PIFO[27], PCQ[28]	可在当今的可编程交换机上实现	理论性能不及PIFO, PIEO
DAG		Cliff[30], CUSP[31], Chimpp[32]	利用Click简化FPGA开发, 提高代码复用率	需要用Verilog/VHDL重新开发元素, 编程困难, 工作量大
		C++代码编译技术[33,34]	通过编译技术充分利用已有的Click的C++代码, 减少工作量	编译效果不理想, 存在性能瓶颈
		ClickNP[35]	采用扩展C语言提高开发效率, 同时支持CPU/FPGA联合处理	开发工具链并未开源
		OpenFunction[36]	适用于中间件的数据平面抽象, 并通过软件定义构建复杂功能	OpenFunction语言生成的中间件与FPGA优化后的相比, 性能差距较大

2.1 高级编程语言与高层次综合工具

FPGA 使用硬件描述语言 (比如 Verilog、VHDL 等) 进行编程, 这些语言抽象层次低, 学习难度高; 好的实现还需要程序员具有数字逻辑设计的基础知识和硬件设计的思维方式, 程序员需要从受约束的硬件流水线的角度而不是高级算法的角度考虑问题. 因此, FPGA 编程较为困难.

为了解决 FPGA 编程困难的问题, 如前所述, RMT 抽象提供了高级编程语言 P4, 而增强 RMT 抽象和 DAG 抽象一般支持通用高级语言, 比如 C/C++、C#等, 前者编程简单, 后者为软件开发人员提供了熟悉的开发环境, 软件开发人员可以利用已有的开发经验, 复用之前的代码, 两者均隐藏了底层硬件细节, 提高了开发效率. 无论是 P4 还是通用高级语言都需要高层次综合工具 (HLS) 将代码映射到目标平台.

2.1.1 应用于 FPGA 的 P4 编译器

P4 的平台无关性主要是通过特定编译器实现, 具体而言, 前端编译器将 P4 程序编译成一种高级中间表示, 后端编译器将该中间表示编译到目标平台上. 近年来, 大量的研究工作给出了多种应用于 FPGA 的 P4 编译器的设计和实现.

Wang 等人[38,39]复用了 P4 社区提供的前端编译器和库[40], 同时设计并实现了 P4 程序到 FPGA 平台的后端编译器和运行时系统 P4FPGA. 通过将 P4 提供的高级编程抽象与灵活而强大的 FPGA 硬件相结合, P4FPGA 允许开发人员快速实现和部署新协议和新应用, 相关代码已开源.

FPGA 厂商 Xilinx 也发布了自己的商用网络开发工具 SDNet[41], 该工具支持将 P4 程序转换为可综合的 Verilog 代码[42]. 基于 Xilinx P4-SDNet 和 NetFPGA SUME 开源代码, Ibanez 等人[43]发布了 P4→NetFPGA 工作流程, 允许开发人员使用 P4 语言描述数据包处理过程, 且 P4 程序编译后能够在 NetFPGA SUME 板上线速运行.

文献 [44−47] 设计并实现了一种网络处理模块生成器, 通过高效的转换算法将 P4 程序转换为可综合的 VHDL 代码, 但目前只支持生成包解析器和逆解析器. Kekely 等人[48]设计了一种新的架构来实现 P4 匹配-动作表到 FPGA 的映射, 该架构采用 DCFL 算法[49], 能够有效平衡处理速度和可用内存资源. Cao 等人[50,51]更进一步, 提出了一种基于模板的网络处理器生成框架, 将 P4 程序映射为预先构建的高效 VHDL 模板, 最终生成适当的 FPGA 代码. 他们还提出了一个预构建的评估库, 有助于编译器在映射阶段对设计进行优化, 从而提高最终实现的性能.

© 中国科学院软件研究所 http://www.jos.org.cn

实验结果表明, 生成的网络处理器能够支持接近 100 Gb/s 的线速处理. 然而, 模板的构建需要 FPGA 专家帮忙.

2.1.2 针对通用高级语言的 HLS

现有的 HLS 工具, 比如 Xilinx Vivado HLS[52], 只支持一些基本的数据结构, 如固定大小的数组和队列等. 为了在 HLS 中使用更广泛的数据结构, 如优先级队列、堆和树等, Zhao 等人[53]提出了一种全新的 HLS 体系结构模板, 使用延迟不敏感接口将复杂的数据结构从算法中解耦出来, 以便于流水线和并行化处理.

为了创建高性能电路, HLS 工具要求程序员遵循特定的设计模式, 以实现流水线优化, 但这限制了代码模块化和复用[54]. 针对该问题, Silva 等人[55]提出了 MpO, 一种高性能、低面积和模块化的 HLS 设计方法, 适用于 FPGA 专业知识有限的硬件设计人员和软件开发人员. 该方法利用支持 HLS 的现代 C++的强大功能构建硬件模块, 可以实现高质量的软件描述和高效的硬件生成. Da Sultana 等人[56]构建了一种新的 FPGA 硬件编译器标准库 Emu, 它使开发人员能够快速创建和部署网络功能, 但 Emu 使用 Kiwi[57] HLS 工具对 C#程序进行高级综合. Eran 等人[58]使用现代 C++编写模块化和泛型代码, 实现了一个网络应用程序公共库 NTL, 提高了用 Xilinx Vivado HLS 编写的包处理应用程序的代码可重用性, 该方法目前仍然需要使用 Verilog 和外部 IP 来处理诸如内存管理单元 (MMU) 等事务.

除此之外, FPGA 厂商还提供了基于 OpenCL 的开发工具链 SDAccel[59], 提供了类似 GPU 的编程模型. Ruan 等人[60]分析了 SDAccel 编程模型及其实现在处理流负载时的不足, 指出在 FPGA 中通过 FIFO 进行流式处理的效率比共享内存更高, 可以达到更低的延迟和更高的吞吐量. 基于此, 提出了一种新的用于 FPGA 上流式应用的高级编程平台 ST-Accel. 实验证明, 与 SDAccel 相比, ST-Accel 可以实现 1.6–166 倍吞吐量和 1/3 延迟.

综上所述, 利用高级编程语言配合强大的编译器, 可以简化 FPGA 开发, 模块化思想和公共库可以进一步提高开发效率, 但最终实现的性能依赖于编译器. 然而, 编译过程中得到的中间结果对用户来说是透明的, 这使得手动干预和优化非常困难, 且编译过程中转换步骤越多, 性能损失越大. 表 2 对 FPGA 的编程简化方法进行了总结, 从中可以看出, 对于部分功能块, 可以通过设计编译器将高级语言编写的代码映射到优化后的 VHDL 模板, 实现超 200 Gb/s 的吞吐量.

表 2 FPGA 的编程简化方法总结

关键技术类型	针对的抽象	相关技术	相关工作	主要技术特点
应用于FPGA的P4编译器	RMT	P4编译器	P4FPGA[38,39]	后端编译器, 配合P4社区提供的前端编译器和库将P4程序编译为FPGA程序
			SDNet[41,42]	前端编译器, 将面向SimpleSumeSwitch体系结构的P4程序编译成HDL模块
		模块生成器	P4-To-VHDL[44,45], Cabal等人[46], Luinaud等人[47]	将P4程序映射为架构优化后的适用于部署在FPGA的解析器/逆解析器, 吞吐量超200 Gb/s
			Kekely等人[48]	将P4匹配-动作表映射到FPGA, 采用DCFL算法有效平衡处理速度和可用内存资源
			Cao等人[50,51]	将P4程序编译成优化后的VHDL模板, 可实现接近100 Gb/s线速分组转发
针对通用高级语言的HLS	增强RMT, DAG	开发工具	Vivado HLS[52]	使用最广泛的商用工具, 能高效综合仅适用基本数据结构的算法主导型程序
			SDAccel[59]	基于一个类似GPU的编程模型, 非常适合计算密集型任务
		开发方法	Zhao等人[53]	使用延迟不敏感接口将复杂的数据结构从算法中解耦出来, 以便于流水线和并行化处理
			ST-Accel[60]	使用FIFO进行流式处理, 与SDAccel相比, 可以实现1.6–166倍吞吐量和1/3延迟
			MpO[55], Emu[56], NTL[58]	使用支持HLS的现代C++编写模块化和泛型代码, 构建网络应用公共库, 提高代码复用率和生产效率

© 中国科学院软件研究所 http://www.jos.org.cn

2.2 软硬件协同开发技术

FPGA 在一些网络应用场景下无法独立实现整个数据平面功能, 比如: (1) FPGA 资源不足; (2) 针对一些复杂的网络应用场景, 包括有状态防火墙、DDoS 检测与防御系统等, 纯硬件实现困难; (3) FPGA 以“bump-in-the-wire”的拓扑结构嵌入到网卡中, 给网卡提供可编程性[61]. 这就需要引入主机 CPU 以提供更高的软件灵活性和计算能力, 而软硬件协同开发技术是在 CPU+FPGA 异构架构上快速构建网络应用的关键.

2.2.1 构建基础设施

为 FPGA 创建一个设计通常需要程序员从裸机开始, 不仅需要实现核心应用, 还需要构建重要的基础设施, 包括内存控制器、I/O 系统和调试设施等, 所有这些活动都非常复杂和耗时. 学术界和产业界通过 DPR 技术[37]将应用开发人员从复杂的基础设施构建工作中解放出来. 具体来说, FPGA 硬件被划分为静态区域和一个或多个动态区域, 静态区域用于提供不同应用程序所需的公共硬件堆栈 (DMA 引擎、DRAM、PCIe、Ethernet 逻辑等) 和管理功能, 配合软件堆栈 (FPGA 设备驱动、软件 API 等) 以实现 FPGA 与主机、片外 DRAM、I/O 外设等的通信, 这部分逻辑在 FPGA 上电后即刻加载并一直运行. 动态区域用于实现用户应用程序, 可以在运行时动态加载和卸载而不会影响 FPGA 中的其他逻辑. 静态区域和动态区域通过预定义的接口进行交互以支持各个区域开发的解耦, 应用开发人员可以专注于功能和算法的实现, 而将复杂的基础设施构建工作交予专门团队负责.

软硬件堆栈的实现复杂而耗时, 近年来涌现了一大批优秀的实践. 微软的 Catapult[61]部署在数据中心网络中, 用于加速必应 (Bing) 搜索引擎. 亚马逊的 AWS EC2[62]和 IBM 的 SuperVessel[63]作为加速器资源用于向广大用户提供云服务. 学术界的 NetFPGA[64,65]和 FAST[66]常用于网络设备原型构建与验证, NetFPGA 提供了适用于路由器、交换机和网卡的参考设计, 相对于 NetFPGA, FAST 提供了更全面的软件 API 和新的索引机制来支持更广泛的协同设计模型.

部分研究工作扩展了传统软硬件堆栈的设计, 引入了操作系统的概念. Feniks[67]增加了一系列高级功能, 如应用程序和操作系统之间的性能隔离、高效的云资源访问和灵活的 FPGA 资源调度等. AmorphOS[68]通过一组操作系统级抽象和接口实现内存和 I/O 访问的跨域保护. ReconOS[69]通过委托线程实现硬件线程对操作系统函数的调用, 通过将硬件语义集成到标准操作系统环境中为软硬件线程提供了统一的多线程编程模型和操作系统接口, 允许功能在运行时在软件和硬件之间移动, 从而实现软件/硬件设计空间的快速探索.

为了提高平台兼容性, Corundum[70]开发了十几套代码, 以期支持市面上的主流平台, 虽然大部分代码可以复用, 但也大大增加了开发和维护的成本. RIFFA[71]、AmorphOS 以及 ReconOS 通过使用已定义的操作系统接口而不是低级的特定于平台的接口进行通信和同步, 提高了应用程序在不同可重构平台之间的可移植性. LEAP[72]引入了延迟不敏感通道的通信抽象, 延迟不敏感通道的操作行为和接口类似于硬件和软件编程库中常见的并发 FIFO 模块. 基于此抽象, LEAP 构建了统一的抽象的通信 API 和丰富的服务库, 减轻了编程 FPGA 的负担, 同时提供了代码的可移植性和平台的兼容性. LEAP 中体现的设计原则, 同样适用于其他系统.

2.2.2 高速数据通道

在 CPU+FPGA 异构架构中, FPGA 与主机 CPU 之间的数据通道往往会成为整个系统的性能瓶颈. 近年来, 产业界和学术界对构建高速数据通道进行了大量的研究.

FPGA 厂商一般都提供了 SoC 集成方案, 比如 Altera Cyclone V SoC[73]和 Xilinx Zynq-7000 系列 FPGA[74]都提供了硬 ARM 处理器核, 用于提供端到端的解决方案. 文献 [73,74] 对处理器系统与可编程逻辑之间的数据交换进行了实验研究, 评估了 SoC 提供的软硬件通信接口的性能, 给出了基于 SoC 设计的一些建议. SoC 集成方案能够提供高吞吐量的软硬件数据传输, 但 SoC 集成方案存在一些天然不足, 比如受限于功耗, 处理器核心的性能不高; 处理核心损坏时只能换用新的 FPGA 板卡, 灵活性差; 此外, 处理器核心会占用部分芯片面积.

为了使用更高性能的 CPU, 目前最流行的方式是通过高速串行接口, 比如 PCIe, 将 FPGA 板卡连接到独立主机[75,76]. 如何提高 PCIe 链路的数据吞吐量, 充分发挥 PCIe 的传输能力是有效提高系统吞吐量的关键. 文献 [77−79] 针对 1 代到 3 代 PCIe 分别提供了高速 DMA 引擎实现, 包括 DMA 控制器逻辑和 Linux 驱动程序. 文献 [78] 通过将 DMA 地址列表存储在 FPGA 内部来减少 FPGA 资源占用. 文献 [79] 使用 SR-IOV 和 PCI 直通技术实现在单

© 中国科学院软件研究所 http://www.jos.org.cn

个 FPGA 板卡上加速多个虚拟软件设备. DHL[80]通过引入 UIO (userspace I/O) 驱动, 采用轮询模式、批处理、大页分配以及 NUMA (non-uniform memory access) 感知等技术进一步提高 DMA 引擎的性能. 类似的, Intel[81]发布了数据平面开发套件, 支持 Intel FPGA 与 DPDK[82]集成. hXDP[83]则采用不同的设计思路, 在 FPGA 中实现了 Linux XDP (express data path), 尽量让数据包在靠近网络的位置处理, 以避免 PCIe 总线和操作系统引起的延迟.

相比于 SoC 集成方案, 通过本地主机接口的部署方案可以使用性能更强的主机, 架构上更加灵活, 同时提供了更好的电源和物理空间平衡, 但主机 PCIe 接口的数量限制了 FPGA 计算的可扩展性. 鉴于此, 可以通过网络将 FPGA 设备连接到远程 CPU 主机, 这将 FPGA 与本地主机解耦, 从而极大地提高了 FPGA 计算的可扩展性, 但只适用于对数据传输延迟要求不高的应用场景[84−86].

综上, 目前已有一大批高速数据通道的优秀实践, 可以根据实际应用场景进行选择, 比如对 CPU 性能要求不高, 需要快速构建原型系统时可采用 SoC 集成方式; 对 CPU 性能以及对软硬件之间数据传输吞吐量和延迟要求较高时可选用 PCIe 等高速串口连接方式; 当对通信延迟要求不高, 对系统灵活性、稳定性要求较高时优选基于网络的连接方式. 表 3 对软硬件协同开发技术进行了分析和总结.

表 3 软硬件协同开发技术总结

关键技术类型	相关技术	相关工作	主要技术特点	技术优势
构建基础设施	软硬件堆栈	NetFPGA[64,65]	提供了不同网络设备的参考设计	生态完善, 社区活跃
		FAST[66]	提供了更全面的软件API和新的索引机制来支持更广泛的协同设计模型	支持软硬件协同开发
	引入操作系统概念	Feniks[67]	增加了一系列操作系统级功能	应用程序的代码可移植性和平台兼容性较好
		AmorphOS[68]	通过一组操作系统级抽象和接口实现内存和I/O访问的跨域保护	
		ReconOS[69]	在操作系统中集成硬件语义, 为软硬件线程提供了统一的多线程编程模型和操作系统接口	
		LEAP[72]	引入了延迟不敏感通道的通信抽象, 构建统一的通信API和丰富的服务库	
高速数据通道	SoC集成方式	Molanes等人[73], Siva等人[74]	利用FPGA内部提供的硬ARM处理器核, 提供端到端的解决方案	能快速构建原型系统
	PCIe等高速串口连接方式	Kavianipour等人[77], Rota等人[78], Zazo等人[79]	针对1代到3代PCIe分别提供了高速DMA引擎实现, 包括DMA控制器逻辑和Linux驱动程序	软硬件之间数据传输吞吐量高, 延迟低; 可匹配高性能CPU, 处理复杂业务
		DHL[80]	通过引入UIO驱动提高DMA引擎的性能	
		hXDP[83]	在FPGA中实现Linux XDP, 尽量让数据包在靠近网络的位置处理, 以减少处理延迟	
	基于网络的连接方式	文献[84−86]	通过网络连接, 将FPGA与本地主机解耦, 极大地提高了FPGA计算的可扩展性	系统灵活性、稳定性高

2.3 多数据平面切换和共存

网络设备往往需要支持多个应用场景, 不同场景下需要支持的协议和功能可能不同, 可编程数据平面可以在不改变底层硬件的前提下为上述需求提供支持.

基于 RMT 的编程抽象通过编程匹配-动作表来定制网络功能, 快速方便, 但需要支持新协议时需要重新编写代码并编译. POF 利用匹配-动作表代替 RMT 中的可编程解析器, 在一定程度上缓解了上述问题, 但当新协议采用更宽的匹配键时, 需要重新设计和实现匹配-动作表. 与上述方法不同, 基于 DAG 编程抽象的方案通过 DPR 技术实现功能运行时加载/卸载, 将事先编译好的部分 bit 流文件下载到对应的可重构区域即可, 但将部分 bit 流文件下载到 FPGA 需要花费时间, 且 bit 流文件越大, 所花费的时间也越长. 相对于编程匹配-动作表的方案, DPR 的方案更加灵活, 但编程工作量较大.

© 中国科学院软件研究所 http://www.jos.org.cn

此外, 有些场景下需要同时支持多个数据平面, 比如在不影响正常业务流量的情况下对新协议和新功能进行生产环境验证. 基于 RMT 的编程抽象只能通过添加多条流水线来支持多数据平面, 在不需要的时候移除, 添加和移除都需要重新编译并烧写 FPGA, 会造成业务中断. 基于 DAG 的编程抽象由于采用 DPR 技术而支持运行时加载/卸载, 虽然通过 DPR 实现了可编程逻辑的物理隔离, 但不可避免地会存在一些共享资源, 比如网络带宽、内存等, 因此, 有必要提供功能、性能和故障隔离, 避免不同数据平面之间互相干扰.

SwitchBlade[87]为每个虚拟数据平面 (VDP) 提供单独的转发表和虚拟化接口, 并使用一个简单的网络流量限制器为每个 VDP 提供网络隔离. 类似地, Feniks[67]也提供 IO 虚拟化, 以便多个 VDP 可以使用相同的虚拟 IO 接口并获得相似的 IO 性能. 此外 Feniks 还为可重构区域提供了一组模板, 开发人员只能基于该模板实现加速逻辑, 这在一定程度上限制了恶意代码的植入. Chen 等人[88]通过安全低开销的地址转换实现内存和网络隔离, 以避免内存数据污染、系统崩溃. Rozhko[89]提出了一种更高级的内存和网络隔离形式, 在这种情况下, 不同 VDP 使用的网络带宽和内存会受到监控, 并在必要时进行节流, 以在 VDP 之间提供一定程度的性能隔离. Knodel 等人[90]在其 FPGA 平台中引入了更多的安全特性, 除了提供网络和内存访问隔离功能外, 还对部分 bit 流文件执行一组设计规则检查, 目的是检测可能损坏设备的配置. 表 4 分别对多数据平面切换和共存技术的技术特点和不足做了分析和总结. 从中可以看出, DAG 抽象在这方面存在天然优势.

表 4　多数据平面切换和共存技术总结

关键技术类型	针对的抽象	相关工作	主要技术特点	不足
多数据平面切换	RMT	P4[9]	通过编程匹配-动作表来定制网络功能	无法在线编程解析器
	增强RMT	POF[15]	支持解析器, 有状态网络处理和通用队列调度的在线编程	匹配键位宽固定, 可能无法支持新协议
	DAG	DPR[37]	通过DPR技术实现功能的运行时加载/卸载, 提供了最高的灵活性	编程工作量较大
多数据平面共存	RMT 增强RMT	文献[8,15]	只能通过添加多条流水线实现	添加和移除流水线时需要重新配置FPGA, 会造成业务中断
	DAG	SwitchBlade[87], Rozhko[89]	通过网络流量限制器实现VDP之间的网络隔离	内存, 网络和性能隔离功能会占用FPGA资源, 进一步限制共存的可编程数据平面数目
		Feniks[67], Chen等人[88]	通过地址转换, I/O虚拟化等技术实现内存隔离	
		Knodel等人[90]	支持网络和内存隔离, 支持对bit流文件执行一组设计规则检查	

3　基于 F-PDP 的新型可编程网络设备

近年来, 学术界和产业界对基于 F-PDP 的新型可编程网络设备的设计进行了大量的研究, 提出了许多优秀的架构和具体实现以用于原型系统设计、网络功能验证甚至应用于生产网络中. 接下来, 根据不同的应用场景分别介绍基于 F-PDP 的可编程交换机和基于 F-PDP 的可编程网卡.

3.1　基于 F-PDP 的可编程交换机

可编程数据平面的概念首先在网络转发设备上被提出, 至今, 已涌现了大量有关可编程转发设备的研究成果. Yazdinejad 等人[91]提供了 SDN 交换机的一种 FPGA 实现, 该交换机支持 OpenFlow v1.3 协议. Kushwaha 等人[92]通过多个 FPGA 芯片实现了支持 1.44 Tb/s 线速处理的 SDN 节点. Yazdinejad 等人[93]基于 P4 和 FPGA 提出了一种数据平面设备的高层次体系结构, 该体系结构支持有关数据包解析、分类和处理的主要操作.

更多的研究基于通用多核 CPU+FPGA 的异构体系结构. Yang 等人[66]研究发现, 考虑到性能提升和资源消耗之间的权衡, 有些包处理操作并不适合卸载到 FPGA 上. 因此, Yang 等人[66]基于 CPU+FPGA 提出了一种用于快速原型设计的软硬件协同设计框架 FAST. FAST 能方便地集成现有的一些开源框架, 以实现功能扩展. 论文最后

© 中国科学院软件研究所 http://www.jos.org.cn

通过一些原型案例展示了 FAST 的易用性以及性能, 但 FAST 无法构建流级网络盒. 赵玉宇等人[75]基于通用多核 CPU+FPGA 提出了高性能可演进的下一代网络处理器体系架构 HPENP, 在该架构中, FPGA 作为高速数据平面, CPU 作为控制平面和数据平面的补充, 两者经 PCIe 实现高速数据交换. 论文提出了一些网络处理优化技术, 并对 HPENP 进行原型构建与测试, 证明了该架构的高性能和高灵活性.

3.2 基于 F-PDP 的可编程网卡

数据中心网络中的链路速度已达到 200 Gb/s, 这给 CPU 带来了巨大的网络开销. CPU 需要消耗大量的处理能力用于网络流量的分类、跟踪及引导, 导致无法处理数据分析等更有价值的工作[94]. 因此, 需要将原先由 CPU 处理的部分功能卸载到网卡上处理, 网卡和交换机之间的界限正变得模糊, 网卡需要将数据包路由到适当的处理核, 但也需要对它们进行转换和过滤, 以减少软件处理和内存子系统的开销[95].

微软云一直致力于用 FPGA 加速其网络服务. 文献 [61] 使用 FPGA 加速必应 (Bing) 搜索, 能显著提高排序吞吐量, 降低尾延迟, 但 FPGA 之间通过专用网络连接, 该架构可扩展性较差. 文献 [96] 优化了 FPGA 之间的互连方式, 将 FPGA 放置于网卡与交换机之间, 并通过一种轻量级流控协议通信. 基于此, 微软云设计并实现了 Azure 可编程网卡, 并将其应用于 AccelNet 中. AccelNet 在可编程网卡上实现了主机 SDN 堆栈, 并将包处理从主机 CPU 卸载到可编程网卡上, 在虚拟环境中提供了接近本地的网络性能. 阿里云[97]和腾讯云[98]都设计和实现了自家的基于 FPGA 的可编程网卡, 用于提升裸金属服务器和普通虚拟机的性能. 腾讯用了大约 10 个人的团队花了不到一年的时间就完成了从规划、实现到最终部署的整个过程, 证明了 FPGA 也可以像软件一样实现敏捷开发.

近年来, 各大设备厂商也相继推出了可编程网卡商用产品, 比如 Mellanox Innova 系列[99]、Xilinx ALVEO 系列[100]以及 Inventec SmartNIC[101]等. 学术界也公布了一系列成果, Zilberman 等人[65]提出了一种基于 FPGA 的可编程网卡的参考设计和实现方法, 这种可编程网卡目前被广泛用作可访问的开发环境, 既可以复用现有的代码库, 又可以支持新的设计. Caulfield 等人[102]认为将基于 FPGA 的可编程网卡和可编程交换机相结合, 可以改变设计和部署网络功能的方式. Yan 等人[103]提出了一个基于 FPGA 的支持 P4 的可编程网卡解决方案, 该方案满足 5G/beyond 5G 网络需求. Stephen 等人[104]提出一种新的网卡架构 PANIC, 通过在卸载引擎中引入更多的可编程组件, 如 FPGA、嵌入式处理器或者用于自定义卸载的 ASIC 等, 提高网卡的灵活性, 以支持更多的功能卸载.

综上所述, FPGA 在可编程交换机中常用于架构设计和原型验证, 更受学术界青睐, 产业界很少推出基于 FPGA 的可编程交换机商用产品, 这是因为交换机功能相对固定, 目前在生产 IP 网络中尚未有大规模部署的复杂网络应用卸载场景, 因此, 设备供应商为了保持产品竞争优势更多追求转发性能与功耗. 与之相反, 可编程网卡由于更靠近主机, 更适合网络堆栈和包处理功能卸载, 已在数据中心网络中得到大规模部署, 实际的应用场景促进了设备供应商不断推出新产品. 基于 F-PDP 的可编程网络设备的详细总结如表 5 所示, 由性能指标一栏可以看出, 对于可编程交换机, 单片 FPGA 可以实现 30 Gb/s 的吞吐量, 多片 FPGA 甚至可实现高达 Tb/s 级别的吞吐量, 分组转发延迟低至微秒级; 对于可编程网卡, 单片 FPGA 即可实现超过 100 Gb/s 的吞吐量, 完全能够应用于生产网络环境.

表 5 基于 F-PDP 的可编程网络设备总结

设备类型	应用场景	相关工作	主要技术特点	性能指标
可编程交换机	SDN交换机	Yazdinejad等人[91]	FPGA用于快速构建原型系统, 支持OpenFlow v1.3协议	主频110.38 MHz, 平均功耗291 mW
		Kushwaha等人[92]	利用多个FPGA芯片提高吞吐量	最高支持1.44 Tb/s线速处理, 转发延迟: 单板10 GbE时3 μs, 多板1 TbE时34 μs
		Yazdinejad等人[93]	高层次体系结构, 支持有关数据包解析、分类和处理的主要操作	当分组大小为1.5 KB时支持50.8 Gb/s转发, 转发延迟1.5 μs (只记录了从开始解析到完成匹配动作这一段时间)
	网络处理器	HPENP[75]	基于CPU+FPGA构建新型网络处理器架构	吞吐量30 Gb/s, 分组转发率30 Mp/s, 转发延迟28 μs
	开发平台	FAST[66]	基于CPU+FPGA提供了软硬件协同开发平台	当分组大小为1 KB时, 经过CPU的吞吐量为480 Mb/s, 延迟为60 μs

© 中国科学院软件研究所 http://www.jos.org.cn

表 5 基于 F-PDP 的可编程网络设备总结 (续)

设备类型	应用场景	相关工作	主要技术特点	性能指标
可编程网卡	数据中心网络	Azure[61, 96]	将包处理卸载到网卡, 在虚拟环境中提供接近本地的网络性能	吞吐量为32 Gb/s, TCP延迟小于 15 μs
		阿里云[97] 腾讯云[98]	在网卡中实现HyperVisor, 提升裸金属服务器和普通虚拟机的性能	—
		Caulfield等人[102]	FPGA部署在网卡和ToR交换机之间, 并且通过PCIe直连CPU	
	5G/B5G网络	Yan等人[103]	支持网络切片, 通过DPR技术在几秒内实现数据平面切换	吞吐量最高为84.8 Gb/s
	开发平台	NetFPGA SUME[65]	开放平台, 代码开源, 社区活跃	吞吐量超过100 Gb/s
	通用新架构	PANIC[104]	加入更多可编程组件, 提高网卡灵活性, 以支持更多应用卸载	双流水线在500 MHz主频下支持双端口100 Gb/s NIC
	商用产品	Mellanox Innova[99]	将FPGA与自家ConnectX-5以太网适配器结合, 提供高可编程性	支持100 GbE, 支持PCIe Gen 4.0
		Xilinx ALVEO[100]	FPGA与NXP处理器结合	支持2×QSFP28, 支持PCIe Gen3×8
		Inventec[101]	FPGA与Intel Xeon D结合	双25 Gb以太网端口, 支持PCIe Gen3×16

4 基于 F-PDP 的网络应用

可编程数据平面提供了包级处理和细粒度可见性, 给网络性能、网络测量以及网络安全带来了革命性的发展机遇, 而 FPGA 因其更多更通用的可编程数据平面抽象, 更丰富的计算和存储资源以及更易扩展的架构给数据平面卸载更多更复杂的网络应用带来了可能.

4.1 使用 F-PDP 提升网络性能

可编程数据平面计算和存储网络状态信息并定制转发行为, 实施 AQM、负载均衡以及拥塞控制等, 为网络服务提供服务质量 (quality of service, QoS) 保障, 而有效进行包分类是基础. 随着匹配字段、匹配规则增多, 提供线速包分类变得更具挑战.

(1) 数据包分类

包分类往往需要在匹配速度、内存开销以及规则更新效率之间做权衡. 基于 RMT 的可编程数据平面通常采用 TCAM 进行数据包匹配和分类, 虽然 TCAM 匹配速度最快, 但成本高、功耗高, 容量也有限. Qi 等人[105]设计了一种内存高效的树搜索算法 HyperSplit, 实现超过 100 Gb/s 的多维数据包分类. HyperSplit 提出一种节点合并算法, 在不显著增加内存需求的前提下, 减少了流水线阶段的数目, 提高了搜索效率. HyperSplit 还设计和实现了一种推叶算法, 控制每个阶段的内存使用并支持规则的动态更新. Fong 等人[106]提出一种规则集分割算法来降低规则集的复杂度, 从而减少内存占用; 同时采用超分裂算法为每个规则子集建立决策树, 并行搜索所有决策树, 将每个决策树的结果进行聚合以获得最终结果, 以实现高吞吐量. 然而, 这种算法在规则变化时需要进行的决策树重构非常耗时.

另一类基于位向量的算法可以实现快速规则更新, 但内存消耗大, 尤其在进行范围匹配时, 常需要将其转化为前缀匹配, 从而导致规则数目增加, 在最坏的情况下, 扩展后的规则数将随着匹配字段数的增加而成倍增加. AFBV[107]借鉴了 StrideBV[108]的思想, 以牺牲一定的匹配速率为代价减少了范围匹配的内存占用, 但 AFBV 破坏了流水线并行性. WeeBV[109]分析了各个规则集中通配符的占比, 通过移除规则中的通配符进一步减少内存占用, 然而, WeeBV 不支持规则的快速更新.

(2) 主动队列管理

AQM 的目标是在保证吞吐量的情况下减小延迟, 并且提供流间公平性. 可编程数据平面使得 AQM 在硬件交换机上实现成为可能, 但仍然需要对一些复杂的操作进行近似处理[110]. 比如, P4-CoDel[111]使用最长前缀匹配表查

© 中国科学院软件研究所 http://www.jos.org.cn

表动作近似复杂的平方根运算, AFQ[112]使用 Count-Min 草图的一个变体, 在有限的交换机内存下维护大量流的状态, 同时使用可编程设备每个端口上可用的多个 FIFO 队列, 以近似排序的方式调度数据包, 实现近似公平的带宽分配. 相对于可编程 ASIC, FPGA 一方面具有丰富的片内计算和存储资源, 可以实现更广泛的 AQM 算法, 另一方面可以利用 DPR 技术, 根据应用场景实现合适的 AQM 算法并动态加载. 此外, 如前所述, 大量的研究基于 FPGA 设计并实现了通用的队列调度和管理模型, 比如 PIFO、PIEO 等, 可以集成到基于 RMT 的编程模型中, 通过在线编程相关参数, 实现广泛的队列调度算法.

(3) 负载均衡

Miao 等人[113]提出一种有状态的 4 层负载均衡器 SilkRoad, 使用交换机上 SRAM 存储和更新连接状态, 并使用 Bloom 滤波器维持每条连接的一致性 (PCC), 但 SRAM 大小有限, 严重影响了 SilkRoad 的可扩展性. 与之不同, Beamer[114]实现了一种无状态的 4 层负载均衡器, 它通过调用托管应用程序实例的服务器的帮助来维护 PCC, 从而绕过了交换机数据平面对每条连接状态维护的要求. Beamer 通过一致性 Hash 和菊花链以确保具有相同五元组的数据包始终能够转发到相同的 DIP (direct IP). 当应用程序实例发生变化时, Beamer 可以通过嵌入到报文头部的地址将数据包重定向到之前使用的应用程序. 类似地, SHELL[115]通过一致性 Hash 和基于 IPv6 段路由的 power-of-choices 方案[116]选择合适的应用程序实例处理新的连接请求以实现负载均衡, 并修改 TCP Timestamps 字段用于记录选择历史, 结合负载均衡器上的选择历史表, 在较小的内存占用下确保了更多连接的 PCC, 同时提供了比 Beamer 更高的系统弹性. 然而, Beamer 和 SHELL 的负载均衡策略仅限于一致性 Hash, 即随机负载均衡, 这会导致负载不均衡, 尤其当连接发生倾斜时.

(4) 拥塞控制

拥塞控制是高速网络实现超低延迟、高带宽和网络稳定性的关键, 可编程数据平面拥有更多的计算资源以及定制计算的能力, 能够以线速度为通过它们的流生成细粒度反馈, 收发端可以据此信息调整收发策略, 在确保公平性的同时提升网络转发能力, 避免网络拥塞[117].

Handley 等人[118]提出了一种应用于数据中心网络的新的传输层协议 NDP, 交换机对接收到的数据包进行逐包多路径负载均衡, 当输出队列快满时裁剪掉当前数据包的载荷, 并优先发送剩余包头, 这使得接收方能够全面了解来自所有发送方的即时要求, 并据此调整各个发送方的发送速率, 但逐包多路径负载均衡会导致包重排, 且只适用于对称拓扑. 相对于可编程交换机 ASIC, FPGA 可以很方便地定制快速获取队列状态信息的接口. HPCC[119]通过 ACK 携带的精确的链路负载信息 (包含精确的链路利用率和队列统计信息) 调整发送速率, 通过解决拥塞时带内遥测信息的延迟和对带内遥测信息的过度反应等问题实现快速收敛, 相关算法已在基于 FPGA 的可编程网卡上实现. 实验结果表明, HPCC 能够在避免拥塞的同时利用空闲带宽, 并在网络队列中保持接近零的延迟. P4air[120]首先将拥塞控制算法分成具有良好内部公平性的组, 然后在可编程数据平面上实现了一个指纹算法, 用于识别不同流所属的算法组, 每个算法组使用一组专用队列强制其竞争流之间的公平性, FPGA 中丰富的存储资源确保了该算法的可扩展性.

综上所述, F-PDP 相对于可编程 ASIC 的优势在于其通用性和丰富的硬件资源. 通用性支持定制更多的数据平面操作, 比如 NDP 协议中实时获取队列状态信息的操作, 实现通用队列调度模型等; 丰富的计算资源可以卸载网络堆栈, 实现复杂算法, 比如 HPCC 在 FPGA NIC 上实现负载均衡算法; 丰富的存储资源可以支持广泛的包分类以及有状态处理, 尤其 FPGA 可以很容易地进行片外扩展, 这进一步增强了 F-PDP 的潜力. 后文表 6 是文中提到的基于 F-PDP 的网络性能提升方案优势与不足的对比.

4.2 基于 F-PDP 的通用网络测量框架

可编程数据平面可定制数据包处理行为, 动态部署网络测量任务, 在不影响正常业务的情况下, 快速获取包级和流级状态信息, 提供网络可见性, 给网络测量带来新机遇. F-PDP 又以丰富的片内和片外资源给网络测量的方案设计和实际部署带来更多可能, 基于 F-PDP 的通用网络测量框架主要包括草图和带内遥测 (in-band network telemetry, INT) 两种.

© 中国科学院软件研究所 http://www.jos.org.cn

表 6 基于 F-PDP 的网络性能提升方案总结

应用类型	相关技术	相关工作	优势	不足
数据包分类	基于树结构的算法	HyperSplit[105]	采用节点合并减少了流水线阶段的数目, 提高了搜索效率	规则更新效率低
		Fong等人[106]	分割规则集, 减少内存占用, 提供搜索并行性	
	基于位向量的算法	AFBV[107]	规则更新快, 借鉴StrideBV的思想展开范围匹配规则, 减少内存占用	破坏了流水线并行性, 牺牲了一定的匹配速率
		WeeBV[109]	移除规则中的通配符, 进一步减少内存占用	规则更新效率低
AQM	专用算法	P4-CoDel[111], AFQ[112]	可以根据应用场景选择合适的算法, 算法实现只受限于计算和存储资源	需要对复杂操作做近似处理, 每种算法都要单独实现
	通用架构	PIFO等[25−29]	可以通过调整通用模型参数实现相当一部分调度算法, 编程简单	算法实现受限于模型
负载均衡	有状态处理	SilkRoad[113]	使用交换机上SRAM维护连接状态, 性能高	可维护的连接状态数受限于SRAM大小, 可扩展性差
	无状态处理	Beamer[114], SHELL[115]	通过在报文头部嵌入连接信息实现无状态处理, 可扩展性好	一致性哈希并不能实现真正的负载均衡
拥塞控制	新的传输层协议	NDP[118]	支持逐包多路径负载均衡	需修改终端协议栈, 存在包乱序问题, 只适用于对称拓扑
	ACK携带链路负载信息	HPCC[119]	ACK携带精确的链路负载信息, 算法收敛速度快	需修改终端设备协议栈
	拥塞控制算法分组	P4air[120]	基于组内良好公平性进行带宽分配, 无需更改收发端网络协议栈	指纹算法消耗FPGA资源, 且引入额外的处理延迟

草图为实现细粒度测量提供了一种替代方法. 与数据包采样不同, 草图是一种紧凑的数据结构, 它可以以固定的内存开销完全记录所有观察到的数据包的所选统计信息, 同时只产生有界误差[121]. 可编程数据平面给基于草图的测量方案在生产 IP 网络中的广泛应用带来了新的机遇. 近年来, 为了进一步减少网络测量开销, 提高网络测量通用性和精度, 相关研究对草图进行了重新设计和优化.

早在 2013 年, OpenSketch[122]通过在 NetFPGA 上定制哈希-分类-计数三级流水线, 实现数据平面和测量平面的正交, 使草图更容易实现. OpenSketch 通过提供多种草图数据结构实现通用性, 控制平面根据具体的测量需求调度数据平面上的草图, 未参与计算的草图数据结构极大地浪费了硬件资源. 与之相反, 戴冕等人[123]使用通用草图数据结构采集数据平面流量数据, 并通过将草图部署在 FPGA 片上 SRAM 中, 以极其有限的硬件资源实现了对较大规模网络流量的实时测量.

Tang 等人[124]研究发现草图中计数器的内存利用率较低, 其主要原因是草图的大多数计数器中的高阶位未使用, 鉴于此, 提出了一种基于数据流历史信息的在线学习算法, 用于确定和调整草图计数器的位宽, 以最大限度地提高基于草图的方法的内存利用率.

然而, 上述草图的计算成本仍然较高, 很难在高速网络中实现线速处理. SketchVisor[125]增加了一条单独的数据路径, 该路径提供快速但精度稍低的测量, 以此提高测量的性能和健壮性. 此外, SketchVisor 通过压缩感知技术进一步恢复精确的网络范围内的测量结果. Yang 等人[126]提出一种新的草图结构, 即 Elastic Sketch. Elastic Sketch 使用不同的数据结构跟踪大象流和老鼠流. 与 SketchVisor 不同, Elastic Sketch 只丢弃部分老鼠流信息, 以合理的精度下降为代价, 获得更快的处理速度. 此外, Elastic Sketch 提出了一种压缩和合并 Sketch 的算法, 以适应当前可用带宽.

与上述丢弃部分信息的方法不同, Tong 等人[127]提出了一种通用架构来加速 FPGA 上的草图, 该架构充分利用 FPGA 的硬件并行处理能力, 实现草图算法的完全流水线处理, 包括流水线哈希计算、流水线哈希表维护以及分布式哈希后操作, 使得系统吞吐量高达 150 Gb/s.

除了上述基于草图的测量框架外, 文献 [128] 使用 FPGA 板卡实现了基于 P4 的 INT, 可以以 100 Gb/s 的线速率实时收集每个数据包的信息. INT 将数据包转发和网络测量结合起来, 交换节点在数据包中加入元数据来获取网络状态. 使用 INT, 网络管理员可以直接从数据平面捕获由性能瓶颈、网络故障或错误配置引起的瞬态问题[129].

© 中国科学院软件研究所 http://www.jos.org.cn

IntSight[130]是在 INT 的基础上设计并实现的一个服务级目标 (service level object, SLO) 违规检测和诊断系统, 它在数据平面逐跳计算路径级网络测度并与 SLO 定义的阈值进行比较, 在出口设备汇总遥测信息并向控制平面上报感兴趣的事件, 控制平面进行分析和 SLO 违规诊断, IntSight 以较低的带宽开销实现了细粒度的检测和诊断, 但检测周期和阈值的设置决定了系统开销和性能.

综上所述, 基于 F-PDP 的通用测量框架主要分为两类, 一类是 INT, 其将网络状态信息以元数据的形式嵌入到业务流中以实现网络测量, 可以实现快速而精确的性能分析和故障诊断, 然而, 构造、封装、填充和提取遥测指令和元数据增加了交换节点的处理压力. 此外, 受最大传输单元 (maximum transmission unit, MTU) 限制, 元数据的数目和大小也是有限制的, 否则将严重影响数据传输效率[129]. 因此, 可编程 ASIC 的优势在于处理性能, 将对正常业务流的影响降到最低, 而 F-PDP 的优势在于定制更多的元数据, 常用作原型设计和功能验证. 另一类是基于草图的通用测量框架, 可以实现数据平面和测量平面的正交, 草图结构的多样化以及持续探索, 较高的内存占用使得 F-PDP 成为优选方案. 表 7 是基于 F-PDP 的通用测量框架的优势与不足的对比.

表 7 基于 F-PDP 的通用测量框架总结

测量框架类型	相关技术	相关工作	优势	不足
基于草图的测量框架	通用设计	OpenSketch[122]	定制哈希-分类-计数三级流水线, 实现数据平面和测试平面正交	实现多种草图, 内存开销大, 资源浪费严重
		戴冕等人[123]	在FPGA片上SRAM中部署通用草图数据结构减少硬件资源占用	SRAM访存会增加处理延迟
	内存优化	Tang等人[124]	动态调整调整草图计数器的位宽, 提高内存利用率	在线学习计算成本高, 占用额外资源
	线速处理优化	SketchVisor[125]	增加快速路径, 提供精度稍低的测量, 提高测量的性能和健壮性	在流量偏差大的情况下, 精度可能会很差
		Elastic Sketch[126]	只丢弃部分老鼠流信息, 在确保测量精度的前提下提高测量健壮性	复杂的大象流识别算法增加了处理延迟
		Tong等人[127]	完全流水线草图算法, 提高系统吞吐量	只给出了两种草图算法的加速实现
INT	原型设计	100 G INT[128]	支持100 Gb/s线速处理	只给出了INT出端口实现
	故障诊断	IntSight[130]	以较低带宽实现细粒度检测与诊断	参数需手动设置, 设置不合理会增加系统开销, 降低性能

4.3 使用 F-PDP 部署网络安全应用

在可编程数据平面上部署网络安全应用, 具备包级跟踪和即时响应等优势, 可以尽早发现并阻断威胁, 这在入侵检测与 DDoS 攻击检测中尤为重要. 此外, 网络中的加密流量越来越多, 加密协议也层出不穷, 数据平面支持加密算法是未来的趋势.

(1) 加密算法

加密可以实现数据的机密性以及完整性保护, 同时可以对参与通信的实体进行身份验证, 是安全通信的基础. 随着越来越多的功能被转移到可编程数据平面, 支持具有强大加密属性的哈希函数将成为各种网络用例的关键使能器. 鉴于目前 P4 缺乏对加密哈希函数的支持, Scholz 等人[131]通过将加密哈希算法作为外部函数集成到支持 P4 的 NetFPGA 上, 为了简单, 作者使用了 FPGA 厂家提供的加密哈希 IP 核, 但该核不支持原生 P4 特性, 这需要修改 P4 交换机模型. 文献 [132,133] 分别在基于 FPGA 的数据平面上实现加密哈希算法和 AES 加密算法, 采用深流水线和全扩展技术提高算法效率, 但会消耗更多的硬件资源. Hauser 等人在数据平面上先后实现了更复杂的加密协议套件 IPsec[134]和 MACsec[135], 由于 P4 不支持加密和解密, 因此作者依赖用户定义的外部函数来执行复杂的计算, 代价是增加延迟和降低吞吐量.

(2) 入侵检测与接入控制

PDP 为在数据平面上实现接入控制成为可能, 从而减轻服务器负载. Ricart-Sanchez 等人[136]在 FPGA 上设计并实现了一种 5G 防火墙, 用于分析边缘网络和核心网络之间传输的 GTP 数据, 首先解析 GTP 报文并获取相关报

© 中国科学院软件研究所 http://www.jos.org.cn

头字段, 然后经流水线进行规则匹配, 实施相应策略. 在后续工作中, 作者通过支持多租户来扩展 5G 防火墙, 但基于 TCAM 的数据包过滤成本高、功耗大且支持的规则数有限[137]. Zhao 等人[138]在可编程网卡中的 FPGA 上实现了 100 Gb/s 入侵检测和防护系统 Pigasus, 该系统直接在 FPGA 上执行 TCP 重组, 以便它可以立即对有效载荷数据进行模式匹配. Pigasus 采用层次化模式匹配, 将交互次数少但内存占用高的完全匹配留给 CPU 处理, 同时在 FPGA 上实现快慢双处理路径, 快速路径处理占比高的顺序包, 确保流水线并行性带来的性能增益, 慢速路径采用链表进行 TCP 重组, 进一步减少内存开销. Qin 等人[139]在可编程数据平面上实现了二值神经网络 (BNN) 用于分类数据包, 采用联邦学习方法以较小的通信开销在线和协作学习新的攻击模式, 同时保持较高的分类精度, 但需要将数据发送到本地 CPU 进行本地训练, 仍然会引入较大的延迟.

(3) DDoS 攻击检测与防御

随着互联网上的数据量呈指数级增长, DDoS 攻击变得越来越常见, 大规模 DDoS 攻击的升级以及缺乏健壮高效的防御机制, 激发了在网络中构建防御的想法. Scholz 等人[140]在不同的可编程数据平面上实现了多种 SYN 泛洪预防策略, 可以在 10 Gb/s 线速率下缓解 SYN 洪泛攻击, 但需要依赖外部加密哈希函数. Gondaliya 等人[141]在 NetFPGA SUME 平台上实现了 6 种 IP 地址防欺骗机制, 并评估和比较了不同机制的性能和资源开销. Kuka 等人[142]在数据平面上缓解反射放大攻击, 通过 FPGA 解析和过滤数据包, 并将感兴趣的摘要信息上传到 CPU, CPU 通过查看源 IP 地址对攻击的贡献量来识别激进 IP 地址, 在检测到数据包速率超过阈值时阻断来自激进 IP 的流量. Hoque 等人[143]提出了一种健壮的相关性度量 NaHiD, 只需要选择一小部分流量特征就能快速识别 DDoS 攻击. Pham-Quoc 等人[144]在 FPGA 上集成了多种 DDoS 防御机制, 利用动态重构技术实现 DDoS 过滤计算核心在运行时更新, 允许系统快速适应不同类型的 DDoS 攻击.

综上所述, 在网络安全应用领域, F-PDP 的优势在于对加密算法, 载荷操作等的支持, 这是可编程 ASIC 所不具备的. 丰富的硬件资源能够支持复杂的算法, 比如相关性测度计算、机器学习算法加速等, 这些是在入侵检测与 DDoS 攻击检测中经常使用到的方法, FPGA 在实现这些方法上具有优势, 但需要注意的是这些方法通常需要 CPU 的协助. 此外, F-PDP 的高度灵活性使得运行时根据网络状态动态加载/卸载防御策略成为可能. 基于 F-PDP 的网络安全应用的对比见表 8.

表 8　基于 F-PDP 的网络安全应用总结

应用类型	相关技术	相关工作	优势	不足
加密	加密算法作为外部函数集成	Scholz等人[131]	使用厂家IP核, 实现简单	增加延迟, 降低吞吐量
		IPsec[134], MACsec[135]	使用自定义外部函数, 无需更改RMT模型	
	FPGA加速加密算法	Sundal等人[132], Chen等人[133]	采用深流水线和全扩展技术提高算法效率	硬件资源开销大
入侵检测与接入控制	模式识别	Ricart-Sanchez等人[136,137]	匹配速度快	TCAM成本高, 功耗大
		Pigasus[138]	可对载荷部分操作, 层次化模式匹配提高了系统可扩展性	层次化结构增加了处理延迟
	机器学习	Qin等人[139]	可以识别新的攻击模式, 联邦学习通信开销低	需要CPU参与本地训练
DDoS攻击检测与防御	针对特定DDoS攻击	Scholz等人[140]	实现多种SYN泛洪预防策略, 支持10 Gb/s处理	依赖外部加密哈希函数
		Gondaliya等人[141]	实现6种IP地址防欺骗机制, 快速构建原型并进行性能和开销比较	测试环境发包速率不足以反应算法的真实性能
		Kuka等人[142]	缓解反射放大攻击, FPGA过滤出感兴趣的报文摘要, 减轻CPU负担	CPU参与降低实时性
	通用架构	Hoque等人[143]	提出一种健壮的相关性度量, 只需要小部分流量特征, 实现快速识别	只给出了仿真结果
		Pham-Quoc等人[144]	利用DPR技术实现防御策略动态加载/卸载	未提供DDoS攻击识别算法

© 中国科学院软件研究所 http://www.jos.org.cn

5 未来研究方向展望

基于 FPGA 的高性能可编程数据平面已经得到了广泛的关注和研究, 许多成果已经产业化并大规模部署, 给当今互联网注入了新的活力. 然而, 相关技术的研究仍任重而道远, 一些关键技术的研究仅处于起步阶段. 接下来, 我们对未来可能的研究方向进行了展望.

5.1 设计适用于 FPGA 的网络应用编程抽象

如前所述, 这部分研究成果很多, 但遗憾的是, RMT 抽象使用领域特定语言虽然可以简化编程, 但表述能力有限, 且无法充分发挥 FPGA 提供的并行处理能力. 增强 RMT 和 DAG 抽象使用通用高级语言编程虽然能描述复杂应用, 但现有的 HLS 工具很难生成高性能硬件代码. 因此, 要编写高性能 FPGA 程序, 目前还是只能使用硬件描述语言. 如何提供高层次抽象, 既能屏蔽底层硬件架构细节, 简化编程; 又能使程序员充分发挥硬件特性, 提高性能, 仍然是未来很长一段时间的研究方向.

5.2 设计适用于 FPGA 的网络测量/安全原语

高级通用语言在 FPGA 硬件细粒度抽象上实现网络测量和网络安全应用, 现有研究虽然给出了一些动作原语, 但主要用于数据包转发, 因此, 有必要分析网络测量和网络安全应用中的通用原子操作, 比如设置时间戳, 统计和获取满足特定条件的数据包数目、队列深度、转发延迟以及跨流状态等. 针对每个原子操作, 给出 FPGA 的单周期实现版本, 以方便应用开发. 目前, 相关的研究还很少, 这将是未来值得研究的方向.

5.3 设计支持负载感知的网络功能动态卸载编程抽象

可编程数据平面的资源是有限的, 不恰当的网络功能卸载反而可能会导致可编程数据平面过载, 从而降低系统性能. 之前的研究工作主要专注于静态功能卸载, iPIPE[145]虽然可以根据运行时工作负载移动计算, 但目前只支持基于 SoC 的可编程网卡. 这方面的研究仅处于起步阶段, 具有一定的局限性. 因此, 如何设计基于负载感知的动态卸载编程抽象, 将成为后续的一大研究方向.

5.4 设计基于国产软硬件的新型可编程网络设备架构

转控分离的思想促进了控制平面和数据平面的并行演进, 但这样也会引入同质化竞争, 降低市场准入门槛. 为了保持产品优势, 在无外在竞争压力的情况下, 厂商一般不太愿意推动架构革新. 然而, 在中美贸易战不断升级的大背景下, 一些器件的供应链安全存在较大的变数. 因此, 基于国产软硬件实现高性能可演进的新型网络设备架构有着长远意义.

6 结束语

可编程数据平面是近年来研究的热点, 而 FPGA 因其通用计算架构和丰富的片内资源, 给构建可编程数据平面带来了更多的可能性. 本文从基于 FPGA 的可编程数据平面抽象, 基于 F-PDP 的关键技术, 新型可编程设备以及网络应用这 4 个方面出发, 对近年来的研究成果进行了阐述和分析, 也得出了一些初步的结论.

(1) 基于 RMT 的可编程数据平面抽象编程简单, 但表述能力有限; 基于 DAG 的可编程数据平面抽象表述能力强, 功能加载/卸载更加灵活, 但编程较困难. 目前基于 RMT 的可编程数据平面在业界是主流, 但随着越来越多的网络功能卸载到可编程数据平面上, 对于表述能力的要求也越来越高, 基于 DAG 的可编程数据平面具有更大的潜力.

(2) 受限于工作主频, FPGA 的性能不及可编程 ASIC, 但更通用的计算架构, 更丰富的片内资源和片外扩展接口使得 FPGA 可以实现可编程 ASIC 所不能实现的一些功能, 比如通用队列管理、加密算法加速、载荷数据分析等. FPGA 不仅可作为可编程数据平面应用于生产网络环境中, 还能用于探索未来的可编程数据平面抽象, 使得快速设计和实现更通用更易编程的数据平面抽象成为可能.

目前, 基于 FPGA 的高性能可编程数据平面仍处于快速发展阶段, 虽然有大量的研究成果已经产业化并部署到生产网络中, 但仍然存在许多技术挑战, 例如适用于 FPGA 的网络应用编程抽象研究、基于国产软硬件的新型可编程网络设备架构研究等. 此外, 一些重要的研究方向才初步涉及, 研究还没有深入开展, 例如适用于 FPGA 的网络测量/安全原语研究、支持负载感知的网络功能动态卸载编程抽象的相关研究等, 这些也在文中未来研究方

© 中国科学院软件研究所 http://www.jos.org.cn

向展望中有所提及, 希望能够为后续研究的开展提供参考和建议.

References:

[1] Li BJ. High performance data center systems with programmable network interface cards [Ph.D. Thesis]. Hefei: University of Science and Technology of China, 2019 (in Chinese with English abstract).

[2] Khosravi H, Anderson T. Requirements for separation of IP control and forwarding. RFC 3654. 2003. 1–18. [doi: 10.17487/RFC3654]

[3] Yang L, Dantu R, Anderson T, Gopal R. Forwarding and control element separation (ForCES) framework. RFC 3746, 2004. 1–40. [doi: 10.17487/RFC3746]

[4] Casado M, Freedman MJ, Pettit J, Luo JY, McKeown N, Shenker S. Ethane: Taking control of the enterprise. ACM SIGCOMM Computer Communication Review, 2007, 37(4): 1–12. [doi: 10.1145/1282427.1282382]

[5] McKeown N, Anderson T, Balakrishnan H, Parulkar G, Peterson L, Rexford J, Shenker S, Turner J. OpenFlow: Enabling innovation in campus networks. ACM SIGCOMM Computer Communication Review, 2008, 38(2): 69–74. [doi: 10.1145/1355734.1355746]

[6] Kreutz D, Ramos FMV, Veríssimo PE, Rothenberg CE, Azodolmolky S, Uhlig S. Software-defined networking: A comprehensive survey. Proc. of the IEEE, 2015, 103(1): 14–76. [doi: 10.1109/JPROC.2014.2371999]

[7] Lin YSX, Bi J, Zhou Y, Zhang C, Wu JP, Liu ZZ, Zhang YR. Research and applications of programmable data plane based on P4. Chinese Journal of Computers, 2019, 42(11): 2539–2560 (in Chinese with English abstract). [doi: 10.11897/SP.J.1016.2019.02539]

[8] Bosshart P, Gibb G, Kim HS, Varghese G, McKeown N, Izzard M, Mujica F, Horowitz M. Forwarding metamorphosis: Fast programmable match-action processing in hardware for SDN. ACM SIGCOMM Computer Communication Review, 2013, 43(4): 99–110. [doi: 10.1145/2534169.2486011]

[9] Bosshart P, Daly D, Gibb G, Izzard M, McKeown N, Rexford J, Schlesinger C, Talayco D, Vahdat AM, Varghese G, Walker DP. P4: Programming protocol-independent packet processors. ACM SIGCOMM Computer Communication Review, 2014, 44(3): 87–95. [doi: 10.1145/2656877.2656890]

[10] Hauser F, Häberle M, Merling D, Lindner S, Gurevich V, Zeiger F, Frank R, Menth M. A survey on data plane programming with P4: Fundamentals, advances, and applied research. arXiv:2101.10632, 2021.

[11] Kfoury EF, Crichigno J, Bou-Harb E. An exhaustive survey on P4 programmable data plane switches: Taxonomy, applications, challenges, and future trends. IEEE Access, 2021, 9: 87094–87155. [doi: 10.1109/ACCESS.2021.3086704]

[12] Kapre N, Bayliss S. Survey of domain-specific languages for FPGA computing. In: Proc. of the 26th Int'l Conf. on Field Programmable Logic and Applications (FPL). Lausanne: IEEE, 2016. 1–12. [doi: 10.1109/FPL.2016.7577380]

[13] Chole S, Fingerhut A, Ma S, Sivaraman A, Vargaftik S, Berger A, Mendelson G, Alizadeh M, Chuang ST, Keslassy I, Orda A, Edsall T. dRMT: Disaggregated programmable switching. In: Proc. of the 2017 Conf. of the ACM Special Interest Group on Data Communication. Los Angeles: ACM, 2017. 1–14. [doi: 10.1145/3098822.3098823]

[14] Sivaraman A, Kim C, Krishnamoorthy R, Dixit A, Budiu M. Dc.P4: Programming the forwarding plane of a data-center switch. In: Proc. of the 1st ACM SIGCOMM Symp. on Software Defined Networking Research. Santa: ACM, 2015. 1–8. [doi: 10.1145/2774993.2775007]

[15] Song HY. Protocol-oblivious forwarding: Unleash the power of SDN through a future-proof forwarding plane. In: Proc. of the 2nd ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking. Hong Kong: ACM, 2013. 127–132. [doi: 10.1145/2491185.2491190]

[16] Bianchi G, Bonola M, Capone A, Cascone C. Openstate: Programming platform-independent stateful OpenFlow applications inside the switch. ACM SIGCOMM Computer Communication Review, 2014, 44(2): 44–51. [doi: 10.1145/2602204.2602211]

[17] Moshref M, Bhargava A, Gupta A, Yu M, Govindan R. Flow-level state transition as a new switch primitive for SDN. In: Proc. of the 3rd Workshop on Hot Topics in Software Defined Networking. Chicago: ACM, 2014. 61–66. [doi: 10.1145/2620728.2620729]

[18] Bianchi G, Bonola M, Pontarelli S, Sanvito D, Capone A, Cascone C. Open packet processor: A programmable architecture for wire speed platform-independent stateful in-network processing. arXiv:1605.01977, 2016.

[19] Pontarelli S, Bifulco R, Bonola M, Cascone C, Spaziani M, Bruschi V, Sanvito D, Siracusano G, Capone A, Honda M, Huici F, Bianchi G. FlowBlaze: Stateful packet processing in hardware. In: Proc. of the 16th USENIX Conf. on Networked Systems Design and Implementation. Boston: ACM, 2019. 531–548.

[20] Sivaraman A, Cheung A, Budiu M, Kim C, Alizadeh M, Balakrishnan H, Varghese G, McKeown N, Licking S. Packet transactions: High-level programming for line-rate switches. In: Proc. of the 2016 ACM SIGCOMM Conf. Florianopolis: ACM, 2016. 15–28. [doi: 10.1145/2934872.2934900]

[21] Hang ZJ, Wen M, Shi Y, Zhang CY. Programming protocol-independent packet processors high-level programming (P4HLP): Towards unified high-level programming for a commodity programmable switch. Electronics, 2019, 8(9): 958. [doi: 10.3390/electronics8090958]

© 中国科学院软件研究所 http://www.jos.org.cn

[22] Cascone C, Bifulco R, Pontarelli S, Capone A. Relaxing state-access constraints in stateful programmable data planes. ACM SIGCOMM Computer Communication Review, 2018, 48(1): 3–9. [doi: 10.1145/3211852.3211854]

[23] Sivaraman A, Winstein K, Subramanian S, Balakrishnan H. No silver bullet: Extending SDN to the data plane. In: Proc. of the 12th ACM Workshop on Hot Topics in Networks. ACM, 2013. 1–7. [doi: 10.1145/2535771.2535796]

[24] Mittal R, Agarwal R, Ratnasamy S, Shenker S. Universal packet scheduling. In: Proc. of the 13th USENIX Symp. on Networked Systems Design and Implementation. Santa Clara: ACM, 2016. 501–521.

[25] Sivaraman A, Subramanian S, Alizadeh M, Chole S, Chuang ST, Agrawal A, Balakrishnan H, Edsall T, Katti S, McKeown N. Programmable packet scheduling at line rate. In: Proc. of the 2016 ACM SIGCOMM Conf. Florianopolis: ACM, 2016. 44–57. [doi: 10.1145/2934872.2934899]

[26] Shrivastav V. Fast, scalable, and programmable packet scheduler in hardware. In: Proc. of the 2019 ACM Special Interest Group on Data Communication. Beijing: ACM, 2019. 367–379. [doi: 10.1145/3341302.3342090]

[27] Alcoz AG, Dietmüller A, Vanbever L. SP-PIFO: Approximating push-in first-out behaviors using strict-priority queues. In: Proc. of the 17th USENIX Symp. on Networked Systems Design and Implementation. Santa Clara: ACM, 2020. 59–76.

[28] Sharma NK, Zhao CXY, Liu M, Kannan PG, Kim C, Krishnamurthy A, Sivaraman A. Programmable calendar queues for high-speed packet scheduling. In: Proc. of the 17th USENIX Symp. on Networked Systems Design and Implementation. Santa Clara: ACM, 2020. 685–699.

[29] Kohler E, Morris R, Chen BJ, Jannotti J, Kaashoek MF. The Click modular router. ACM Trans. on Computer Systems, 2000, 18(3): 263–297. [doi: 10.1145/354871.354874]

[30] Kulkarni C, Brebner G, Schelle G. Mapping a domain specific language to a platform FPGA. In: Proc. of the 41st Annual Design Automation Conf. San Diego: ACM, 2004. 924–927. [doi: 10.1145/996566.996811]

[31] Schelle G, Grunwald D. CUSP: A modular framework for high speed network applications on FPGAs. In: Proc. of the 13th ACM/SIGDA Int'l Symp. on Field-programmable Gate Arrays. Monterey: ACM, 2005. 246–257. [doi: 10.1145/1046192.1046224]

[32] Rubow E, McGeer R, Mogul J, Vahdat A. Chimpp: A click-based programming and simulation environment for reconfigurable networking hardware. In: Proc. of the 2010 ACM/IEEE Symp. on Architectures for Networking and Communications Systems (ANCS). La Jolla: IEEE, 2010. 1–10. [doi: 10.1145/1872007.1872052]

[33] Nikander P, Nyman B, Rinta-Aho T, Sahasrabuddhe SD, Kempf J. Towards software-defined silicon: Experiences in compiling Click to NetFPGA. In: Proc. of the 1st European NetFPGA Developers Workshop. 2010.

[34] Rinta-Aho T, Karlstedt M, Desai MP. The click2NetFPGA toolchain. In: Proc. of the 2012 USENIX Annual Technical Conf. Boston: ACM, 2012. 77–88.

[35] Li BJ, Tan K, Luo L, Peng YQ, Luo RQ, Xu NY, Xiong YQ, Cheng P, Chen EH. ClickNP: Highly flexible and high performance network processing with reconfigurable hardware. In: Proc. of the 2016 ACM SIGCOMM Conf. Florianopolis: ACM, 2016. 1–14. [doi: 10.1145/2934872.2934897]

[36] Tian C, Munir A, Liu AX, Yang J, Zhao YM. OpenFunction: An extensible data plane abstraction protocol for platform-independent software-defined middleboxes. IEEE/ACM Trans. on Networking, 2018, 26(3): 1488–1501. [doi: 10.1109/TNET.2018.2829882]

[37] Blodget B, Bobda C, Hübner M, Niyonkuru A. Partial and dynamically reconfiguration of Xilinx Virtex-II FPGAs. In: Proc. of the 2004 Int'l Conf. on Field Programmable Logic and Applications. Leuven: Springer, 2004. 801–810. [doi: 10.1007/978-3-540-30117-2_81]

[38] Wang H, Soulé R, Dang HT, Lee KS, Shrivastav V, Foster N, Weatherspoon H. P4FPGA: A rapid prototyping framework for P4. In: Proc. of the 2017 Symp. on SDN Research. Santa Clara: ACM, 2017. 122–135. [doi: 10.1145/3050220.3050234]

[39] Wang H, Lee KS, Shrivastav V, Weatherspoon H. P4FPGA: High level synthesis for networking. In: Proc. of the 2016 ACM SIGCOMM Workshop Netw Program Lang (NetPL). Florianópolis: ACM, 2016.

[40] P4 Behavioral Model. 2021. https://github.com/p4lang/p4c-bm

[41] Wirbel L. Xilinx SDNet: A new way to specify network hardware. White Paper, Mountain View: The Linley Group, 2014.

[42] Yazdinejad A, Bohlooli A, Jamshidi K. P4 to SDNet: Automatic generation of an efficient protocol-independent packet parser on reconfigurable hardware. In: Proc. of the 8th Int'l Conf. on Computer and Knowledge Engineering (ICCKE). Mashhad: IEEE, 2018. 159–164. [doi: 10.1109/ICCKE.2018.8566590]

[43] Ibanez S, Brebner G, McKeown N, Zilberman N. The P4→netFPGA workflow for line-rate packet processing. In: Proc. of the 2019 ACM/SIGDA Int'l Symp. on Field-Programmable Gate Arrays. Seaside: ACM, 2019. 1–9. [doi: 10.1145/3289602.3293924]

[44] Benácek P, Pu V, Kubátová H. P4-to-VHDL: Automatic generation of 100 Gbps packet parsers. In: Proc. of the 24th IEEE Annual Int'l Symp. on Field-programmable Custom Computing Machines (FCCM). Washington: IEEE, 2016. 148–155. [doi: 10.1109/FCCM.2016.46]

© 中国科学院软件研究所 http://www.jos.org.cn

[45] Benáček P, Puš V, Kubátová H, Čejka T. P4-To-VHDL: Automatic generation of high-speed input and output network blocks. Microprocessors and Microsystems, 2018, 56: 22–33. [doi: 10.1016/j.micpro.2017.10.012]

[46] Cabal J, Benácek P, Foltova J, Holub J. Scalable P4 deparser for speeds over 100 Gbps. In: Proc. of the 27th IEEE Annual Int'l Symp. on Field-programmable Custom Computing Machines (FCCM). San Diego: IEEE, 2019. 323–323. [doi: 10.1109/FCCM.2019.00064]

[47] Luinaud T, Da Silva JS, Langlois JMP, Savaria Y. Design principles for packet deparsers on FPGAs. In: Proc. of the 2021 ACM/SIGDA Int'l Symp. on Field-programmable Gate Arrays. ACM, 2021. 280–286. [doi: 10.1145/3431920.3439303]

[48] Kekely M, Korenek J. Mapping of P4 match action tables to FPGA. In: Proc. of the 27th Int'l Conf. on Field Programmable Logic and Applications (FPL). Ghent: IEEE, 2017. 1–2. [doi: 10.23919/FPL.2017.8056768]

[49] Taylor DE, Turner JS. Scalable packet classification using distributed crossproducing of field labels. In: Proc. of the 24th IEEE Annual Joint Conf. of the IEEE Computer and Communications Societies. Miami: IEEE, 2005. 269–280. [doi: 10.1109/INFCOM.2005.1497898]

[50] Cao Z, Su HY, Yang QM, Wen M, Zhang CY. Poster abstract: A template-based framework for generating network processor in FPGA. In: Proc. of the 2019 IEEE Conf. on Computer Communications Workshops (IEEE INFOCOM 2019). Paris: IEEE, 2019. 1057–1058. [doi: 10.1109/INFCOMW.2019.8845037]

[51] Cao Z, Su HY, Yang QM, Shen JZ, Wen M, Zhang CY. P4 to FPGA—A fast approach for generating efficient network processors. IEEE Access, 2020, 8: 23440–23456. [doi: 10.1109/ACCESS.2020.2970683]

[52] AMD. Xilinx. 2021. https://www.xilinx.com/support/download.html

[53] Zhao R, Liu G, Srinath S, Batten C, Zhang ZR. Improving high-level synthesis with decoupled data structure optimization. In: Proc. of the 53rd ACM/EDAC/IEEE Design Automation Conf. (DAC). Austin: IEEE, 2016. 1–6. [doi: 10.1145/2897937.2898030]

[54] Karras K, Hrica J. Designing protocol processing systems with vivado high-level synthesis. Xilinx application note XAPP 1209 (v1.0.1). 2014.

[55] Da Silva JS, Boyer FR, Langlois JMP. Module-per-object: A human-driven methodology for C++-based high-level synthesis design. In: Proc. of the 27th IEEE Annual Int'l Symp. on Field-programmable Custom Computing Machines (FCCM). San Diego: IEEE, 2019. 218–226. [doi: 10.1109/FCCM.2019.00037]

[56] Sultana N, Galea S, Greaves D, Wójcik M, Shipton J, Clegg R, Mai L, Bressana P, Soulé R, Mortier R, Costa P, Pietzuch PR, Crowcroft J, Moore AW, Zilberman N. EMU: Rapid prototyping of networking services. In: Proc. of the 2017 USENIX Annual Technical Conf. Santa Clara: USENIX, 2017. 459–471. [doi: 10.17863/CAM.13009]

[57] Singh S, Greaves DJ. Kiwi: Synthesis of FPGA circuits from parallel programs. In: Proc. of the 16th Int'l Symp. on Field-programmable Custom Computing Machines. Stanford: IEEE, 2008. 3–12. [doi: 10.1109/FCCM.2008.46]

[58] Eran H, Zeno L, István Z, Silberstein M. Design patterns for code reuse in HLS packet processing pipelines. In: Proc. of the 27th IEEE Annual Int'l Symp. on Field-programmable Custom Computing Machines (FCCM). San Diego: IEEE, 2019. 208–217. [doi: 10.1109/FCCM.2019.00036]

[59] Xilinx. SDAccel: Enabling hardware-accelerated software. 2021. https://www.xilinx.com/products/design-tools/legacy-tools/sdaccel.html

[60] Ruan ZY, He T, Li BJ, Zhou PP, Cong J. ST-Accel: A high-level programming platform for streaming applications on FPGA. In: Proc. of the 26th IEEE Annual Int'l Symp. on Field-programmable Custom Computing Machines (FCCM). Boulder: IEEE, 2018. 9–16. [doi: 10.1109/FCCM.2018.00011]

[61] Putnam A, Caulfield AM, Chung ES, Chiou D, Constantinides K, Demme J, Esmaeilzadeh H, Fowers J, Gopal GP, Gray J, Haselman M, Hauck S, Heil S, Hormati A, Kim JY, Lanka S, Larus J, Peterson E, Pope S, Smith A, Thong J, Xiao PY, Burger D. A reconfigurable fabric for accelerating large-scale datacenter services. In: Proc. of the 41st ACM/IEEE Int'l Symp. on Computer Architecture (ISCA). Minneapolis: IEEE, 2014. 13–24. [doi: 10.1109/ISCA.2014.6853195]

[62] Amazon EC2 F1 Instances. 2021. https://aws.amazon.com/ec2/instance-types/f1/

[63] Lin YH, Shao L. Supervessel: The open cloud service for openpower. White paper, IBM Corporation, 2015.

[64] Lockwood JW, McKeown N, Watson G, Gibb G, Hartke P, Naous J, Raghuraman R, Luo JY. NetFPGA—An open platform for gigabit-rate network switching and routing. In: Proc. of the 2007 IEEE Int'l Conf. on Microelectronic Systems Education (MSE'07). San Diego: IEEE, 2007. 160–161. [doi: 10.1109/MSE.2007.69]

[65] Zilberman N, Audzevich Y, Covington GA, Moore AW. NetFPGA SUME: Toward 100 Gbps as research commodity. IEEE Micro, 2014, 34(5): 32–41. [doi: 10.1109/MM.2014.61]

[66] Yang XR, Sun ZG, Li JN, Yan JL, Li T, Quan W, Xu DL, Antichi G. FAST: Enabling fast software/hardware prototype for network experimentation. In: Proc. of the 27th IEEE/ACM Int'l Symp. on Quality of Service (IWQoS). Phoenix: IEEE, 2019. 1–10. [doi: 10.

© 中国科学院软件研究所 http://www.jos.org.cn

1145/3326285.3329067]

[67] Zhang JS, Xiong YQ, Xu NY, Shu R, Li BJ, Cheng P, Chen G, Moscibroda T. The Feniks FPGA operating system for cloud computing. In: Proc. of the 8th Asia-Pacific Workshop on Systems. Mumbai: ACM, 2017. 1–7. [doi: 10.1145/3124680.3124743]

[68] Khawaja A, Landgraf J, Prakash R, Wei M, Schkufza E, Rossbach CJ. Sharing, protection, and compatibility for reconfigurable fabric with Amorphos. In: Proc. of the 13th USENIX Symp. on Operating Systems Design and Implementation. Carlsbad: ACM, 2018. 107–127.

[69] Agne A, Happe M, Keller A, Lübbers E, Plattner B, Platzner M, Plessl C. ReconOS: An operating system approach for reconfigurable computing. IEEE Micro, 2014, 34(1): 60–71. [doi: 10.1109/MM.2013.110]

[70] Forencich A, Snoeren AC, Porter G, Papen G. Corundum: An open-source 100-Gbps NIC. In: Proc. of the 28th IEEE Annual Int'l Symp. on Field-programmable Custom Computing Machines (FCCM). Fayetteville: IEEE, 2020. 38–46. [doi: 10.1109/FCCM48280.2020.00015]

[71] Jacobsen M, Richmond D, Hogains M, Kastner R. RIFFA 2.1: A reusable integration framework for FPGA accelerators. ACM Trans. on Reconfigurable Technology and Systems, 2015, 8(4): 22. [doi: 10.1145/2815631]

[72] Fleming K, Adler M. The LEAP FPGA operating system. In: Koch D, Hannig F, Ziener D, eds. FPGAs for Software Programmers. Cham: Springer, 2016. 245–258. [doi: 10.1007/978-3-319-26408-0_14]

[73] Molanes RF, Rodríguez-Andina JJ, Farina J. Performance characterization and design guidelines for efficient processor—FPGA communication in Cyclone V FPSoCs. IEEE Trans. on Industrial Electronics, 2018, 65(5): 4368–4377. [doi: 10.1109/TIE.2017.2766581]

[74] Silva J, Sklyarov V, Skliarova I. Comparison of on-chip communications in Zynq-7000 all programmable systems-on-chip. IEEE Embedded Systems Letters, 2015, 7(1): 31–34. [doi: 10.1109/LES.2015.2399656]

[75] Zhao YY, Cheng G, Liu XH, Yuan S, Tang L. Research and applications of next generation network processor. Ruan Jian Xue Bao/Journal of Software, 2021, 32(2): 445–474 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6124.htm [doi: 10.13328/j.cnki.jos.006124]

[76] Niemiec GS, Batista LMS, Schaeffer-Filho AE, Nazar GL. A survey on FPGA support for the feasible execution of virtualized network functions. IEEE Communications Surveys & Tutorials, 2020, 22(1): 504–525. [doi: 10.1109/COMST.2019.2943690]

[77] Kavianipour H, Muschter S, Bohm C. High performance FPGA-based DMA interface for PCIe. IEEE Trans. on Nuclear Science, 2014, 61(2): 745–749. [doi: 10.1109/TNS.2014.2304691]

[78] Rota L, Caselle M, Chilingaryan S, Kopmann A, Weber M. A PCIe DMA architecture for multi-gigabyte per second data transmission. IEEE Trans. on Nuclear Science, 2015, 62(3): 972–976. [doi: 10.1109/TNS.2015.2426877]

[79] Zazo JF, Lopez-Buedo S, Audzevich Y, Moore AW. A PCIe DMA engine to support the virtualization of 40 Gbps FPGA-accelerated network appliances. In: Proc. of the 2015 Int'l Conf. on Reconfigurable Computing and FPGAs (ReConFig). Riviera Maya: IEEE, 2015. 1–6. [doi: 10.1109/ReConFig.2015.7393334]

[80] Li XY, Wang XX, Liu FM, Xu H. DHL: Enabling flexible software network functions with FPGA acceleration. In: Proc. of the 38th IEEE Int'l Conf. on Distributed Computing Systems (ICDCS). Vienna: IEEE, 2018. 1–11. [doi: 10.1109/ICDCS.2018.00011]

[81] Data plane development kit reference manual: Intel FPGA programmable acceleration card N3000. 2019. https://www.intel.cn/content/www/cn/zh/programmable/documentation/bfs1571412151638.html

[82] Intel. Data plane development kit. 2021. http://www.dpdk.org

[83] Brunella MS, Belocchi G, Bonola M, Pontarelli S, Siracusano G, Bianchi G, Cammarano A, Palumbo A, Petrucci L, Bifulco R. hXDP: Efficient software packet processing on FPGA NICs. In: Proc. of the 14th USENIX Symp. on Operating Systems Design and Implementation. ACM, 2020. 973–990.

[84] Weerasinghe J, Polig R, Abel F, Hagleitner C. Network-attached FPGAs for data center applications. In: Proc. of the 2016 Int'l Conf. on Field-programmable Technology (FPT). Xi'an: IEEE, 2016. 36–43. [doi: 10.1109/FPT.2016.7929186]

[85] Knodel O, Genssler PR, Erxleben F, Spallek RG. FPGAs and the cloud—An endless tale of virtualization, elasticity and efficiency. Int'l Journal on Advances in Systems and Measurements, 2018, 11(3–4): 230–249.

[86] Quraishi MH, Tavakoli EB, Ren FB. A survey of system architectures and techniques for FPGA virtualization. IEEE Trans. on Parallel and Distributed Systems, 2021, 32(9): 2216–2230. [doi: 10.1109/TPDS.2021.3063670]

[87] Anwer MB, Motiwala M, Tariq MB, Feamster N. SwitchBlade: A platform for rapid deployment of network protocols on programmable hardware. In: Proc. of the 2010 ACM SIGCOMM Conf. New Delhi: ACM, 2010. 183–194.

[88] Chen F, Shan Y, Zhang Y, Wang Y, Franke H, Cheng XT, Wang K. Enabling FPGAs in the cloud. In: Proc. of the 11th ACM Conf. on Computing Frontiers. Cagliari: ACM, 2014. 1–10. [doi: 10.1145/2597917.2597929]

© 中国科学院软件研究所 http://www.jos.org.cn

[89] Rozhko D. Memory and network interface virtualization for multi-tenant reconfigurable compute devices [MS. Thesis]. Toronto: University of Toronto, 2018.

[90] Knodel O, Lehmann P, Spallek RG. RC3E: Reconfigurable accelerators in data centres and their provision by adapted service models. In: Proc. of the 9th IEEE Int'l Conf. on Cloud Computing (CLOUD). San Francisco: IEEE, 2016. 19–26. [doi: 10.1109/CLOUD.2016.0013]

[91] Yazdinejad A, Bohlooli A, Jamshidi K. Efficient design and hardware implementation of the OpenFlow v1.3 Switch on the Virtex-6 FPGA ML605. The Journal of Supercomputing, 2018, 74(3): 1299–1320. [doi: 10.1007/s11227-017-2175-7]

[92] Kushwaha A, Sharma S, Bazard N, Gumaste A, Mukherjee B. Design, analysis, and a terabit implementation of a source-routing-based SDN data plane. IEEE Systems Journal, 2021, 15(1): 56–67. [doi: 10.1109/JSYST.2020.2995605]

[93] Yazdinejad A, Parizi RM, Bohlooli A, Dehghantanha A, Choo KKR. A high-performance framework for a network programmable packet processor using P4 and FPGA. Journal of Network and Computer Applications, 2020, 156: 102564. [doi: 10.1016/j.jnca.2020.102564]

[94] Achieving a Cloud-scale architecture with DPUs. 2021. https://blog.mellanox.com/2018/09/why-you-need-smart-nic-use-cases/

[95] Kaufmann A, Peter S, Anderson T, Krishnamurthy A. FlexNIC: Rethinking network DMA. In: Proc. of the 15th Workshop on Hot Topics in Operating Systems. USENIX, 2015.

[96] Firestone D, Putnam A, *et al*. Azure accelerated networking: SmartNICs in the public cloud. In: Proc. of the the 15th USENIX Symp. on Networked Systems Design and Implementation. Renton: ACM, 2018. 51–64.

[97] Alibaba Cloud. 2018. https://yq.aliyun.com/articles/594276/

[98] Luo LL, Tencent TEG. Towards converged SmartNIC architecture for bare metal & public clouds. APNet. 2018.

[99] Mellanox Technologies. Mellanox Innova™-2 Flex Open Programmable SmartNIC. 2021. https://www.mellanox.com/files/doc-2020/pb-innova-2-flex.pdf

[100] Xilinx. Adaptable accelerator cards for data center workloads. 2021. https://www.xilinx.com/products/boards-and-kits/alveo.html

[101] Inventec. Inventec FPGA IPU C5020X. 2023. https://ebg.inventec.com/en/product/Accessories/Smart%20NIC%20Card/Inventec%20FPGA%20IPU%20C5020X

[102] Caulfield A, Costa P, Ghobadi M. Beyond SmartNICs: Towards a fully programmable cloud. In: Proc. of the 19th IEEE Int'l Conf. on High Performance Switching and Routing (HPSR). Bucharest: IEEE, 2018. 1–6. [doi: 10.1109/HPSR.2018.8850757]

[103] Yan Y, Beldachi AF, Nejabati R, Simeonidou D. P4-enabled Smart NIC: Enabling sliceable and service-driven optical data centres. Journal of Lightwave Technology, 2020, 38(9): 2688–2694. [doi: 10.1109/JLT.2020.2966517]

[104] Stephens B, Akella A, Swift MM. Your programmable NIC should be a programmable switch. In: Proc. of the 17th ACM Workshop on Hot Topics in Networks. Redmond: ACM, 2018. 36–42. [doi: 10.1145/3286062.3286068]

[105] Qi YX, Fong J, Jiang WR, Xu B, Li J, Prasanna V. Multi-dimensional packet classification on FPGA: 100 Gbps and beyond. In: Proc. of the 2010 Int'l Conf. on Field-programmable Technology. Beijing: IEEE, 2010. 241–248. [doi: 10.1109/FPT.2010.5681492]

[106] Fong J, Wang X, Qi YX, Li J, Jiang WR. ParaSplit: A scalable architecture on FPGA for terabit packet classification. In: Proc. of the 20th IEEE Annual Symp. on High-Performance Interconnects. Santa Clara: IEEE, 2012. 1–8. [doi: 10.1109/HOTI.2012.17]

[107] Zheng L, Qiu ZL, Wang WN, Pan WT, Sun SY, Gao Y. AFBV: A high-performance network flow classification method for multi-dimensional fields and FPGA implementation. Journal of Circuits, Systems and Computers, 2019, 28(14): 1950237. [doi: 10.1142/S0218126619502372]

[108] Ganegedara T, Prasanna VK. StrideBV: Single chip 400G+ packet classification. In: Proc. of the 13th IEEE Int'l Conf. on High Performance Switching and Routing. Belgrade: IEEE, 2012. 1–6. [doi: 10.1109/HPSR.2012.6260820]

[109] Li CL, Li T, Li JN, Li DG, Yang H, Wang BS. Memory optimization for bit-vector-based packet classification on FPGA. Electronics, 2019, 8(10): 1159. [doi: 10.3390/electronics8101159]

[110] Kunze I, Gunz M, Saam D, Wehrle K, Rüth J. Tofino+ P4: A strong compound for AQM on high-speed networks? In: Proc. of the 2021 IFIP/IEEE Int'l Symp. on Integrated Network Management. Bordeaux: IEEE, 2021. 72–80.

[111] Kundel R, Blendin J, Viernickel T, Koldehofe B, Steinmetz R. P4-CoDel: Active queue management in programmable data planes. In: Proc. of the 2018 IEEE Conf. on Network Function Virtualization and Software Defined Networks (NFV-SDN). Verona: IEEE, 2018. 1–4. [doi: 10.1109/NFV-SDN.2018.8725736]

[112] Sharma NK, Liu M, Atreya K, Krishnamurthy A. Approximating fair queueing on reconfigurable switches. In: Proc. of the 15th USENIX Symp. on Networked Systems Design and Implementation. Renton: ACM, 2018. 1–16.

[113] Miao R, Zeng HY, Kim C, Lee J, Yu ML. SilkRoad: Making stateful layer-4 load balancing fast and cheap using switching ASICs. In: Proc. of the 2017 Conf. of the ACM Special Interest Group on Data Communication. Los Angeles: ACM, 2017. 15–28. [doi: 10.1145/

© 中国科学院软件研究所 http://www.jos.org.cn

3098822.3098824]

[114] Olteanu V, Agache A, Voinescu A, Raiciu C. Stateless datacenter load-balancing with beamer. In: Proc. of the 15th USENIX Symp. on Networked Systems Design and Implementation. Berkeley: ACM, 2018. 125–139.

[115] Pit-Claudel B, Desmouceaux Y, Pfister P, Townsley M, Clausen T. Stateless load-aware load balancing in P4. In: Proc. of the 26th IEEE Int'l Conf. on Network Protocols (ICNP). Cambridge: IEEE, 2018. 418–423. [doi: 10.1109/ICNP.2018.00058]

[116] Desmouceaux Y, Pfister P, Tollet J, Townsley M, Clausen T. 6LB: Scalable and application-aware load balancing with segment routing. IEEE/ACM Trans. on Networking, 2018, 26(2): 819–834. [doi: 10.1109/TNET.2018.2799242]

[117] Tahiliani MP, Misra V, Ramakrishnan KK. A principled look at the utility of feedback in congestion control. In: Proc. of the 2019 Workshop on Buffer Sizing. Palo Alto: ACM, 2019. 1–5. [doi: 10.1145/3375235.3375243]

[118] Handley M, Raiciu C, Agache A, Voinescu A, Moore AW, Antichi G, Wójcik M. Re-architecting datacenter networks and stacks for low latency and high performance. In: Proc. of the 2017 Conf. of the ACM Special Interest Group on Data Communication. Los Angeles: ACM, 2017. 29–42. [doi: 10.1145/3098822.3098825]

[119] Li YL, Miao R, Liu HH, Zhuang Y, Feng F, Tang LB, Cao Z, Zhang M, Kelly F, Alizadeh M, Yu ML. HPCC: High precision congestion control. In: Proc. of the 2019 ACM Special Interest Group on Data Communication. Beijing: ACM, 2019. 44–58. [doi: 10.1145/3341302.3342085]

[120] Turkovic B, Kuipers F. P4air: Increasing fairness among competing congestion control algorithms. In: Proc. of the 28th IEEE Int'l Conf. on Network Protocols (ICNP). Madrid: IEEE, 2020. 1–12. [doi: 10.1109/ICNP49622.2020.9259405]

[121] Li SS, Luo LL, Guo DK, Zhang QZ, Fu PT. A survey of sketches in traffic measurement: Design, optimization, application and implementation. arXiv:2012.07214, 2020.

[122] Yu ML, Jose L, Miao R. Software defined traffic measurement with OpenSketch. In: Proc. of the 10th USENIX Symp. on Networked Systems Design and Implementation. Berkeley: ACM, 2013. 29–42.

[123] Dai M, Cheng G. Sketch-based data plane hardware model for software-defined measurement. Journal on Communications, 2017, 38(10): 113–121 (in Chinese with English abstract). [doi: 10.11959/j.issn.1000-436x.2017203]

[124] Tang MJ, Wen M, Shen JZ, Zhao XL, Zhang CY. Towards memory-efficient streaming processing with counter-cascading sketching on FPGA. In: Proc. of the 57th ACM/IEEE Design Automation Conf. (DAC). San Francisco: IEEE, 2020. 1–6. [doi: 10.1109/DAC18072.2020.9218503]

[125] Huang Q, Jin X, Lee PPC, Li RH, Tang L, Chen YC, Zhang G. SketchVisor: Robust network measurement for software packet processing. In: Proc. of the 2017 Conf. of the ACM Special Interest Group on Data Communication. Los Angeles: ACM, 2017. 113–126. [doi: 10.1145/3098822.3098831]

[126] Yang T, Jiang J, Liu P, Huang Q, Gong JZ, Zhou Y, Miao R, Li XM, Uhlig S. Elastic sketch: Adaptive and fast network-wide measurements. In: Proc. of the 2018 Conf. of the ACM Special Interest Group on Data Communication. Budapest: ACM, 2018. 561–575. [doi: 10.1145/3230543.3230544]

[127] Tong D, Prasanna VK. Sketch acceleration on FPGA and its applications in network anomaly detection. IEEE Trans. on Parallel and Distributed Systems, 2018, 29(4): 929–942. [doi: 10.1109/tpds.2017.2766633]

[128] 100G in-band network telemetry with Netcope P4. 2021. https://www.netcope.com/Netcope/media/content/100G-In-band-Network-Telemetry-With-Netcope-P4.pdf

[129] Tan LZ, Su W, Zhang W, Lv JH, Zhang ZY, Miao JY, Liu XX, Li N. In-band network telemetry: A survey. Computer Networks, 2021, 186: 107763. [doi: 10.1016/j.comnet.2020.107763]

[130] Marques J, Levchenko K, Gaspary L. IntSight: Diagnosing SLO violations with in-band network telemetry. In: Proc. of the 16th Int'l Conf. on Emerging Networking Experiments and Technologies. Barcelona: ACM, 2020. 421–434. [doi: 10.1145/3386367.3431306]

[131] Scholz D, Oeldemann A, Geyer F, Gallenmüller S, Stubbe H, Wild T, Herkersdorf A, Carle G. Cryptographic hashing in P4 data planes. In: Proc. of the 2019 ACM/IEEE Symp. on Architectures for Networking and Communications Systems (ANCS). Cambridge: IEEE, 2019. 1–6. [doi: 10.1109/ANCS.2019.8901886]

[132] Sundal M, Chaves R. Efficient FPGA implementation of the SHA-3 hash function. In: Proc. of the 2017 IEEE Computer Society Annual Symp. on VLSI (ISVLSI). Bochum: IEEE, 2017. 86–91. [doi: 10.1109/ISVLSI.2017.24]

[133] Chen S, Hu W, Li ZH. High performance data encryption with AES implementation on FPGA. In: Proc. of the 5th IEEE Int'l Conf. on Big Data Security on Cloud (Big Data Security), IEEE Int'l Conf. on High Performance and Smart Computing, (HPSC) and IEEE Int'l Conf. on Intelligent Data and Security (IDS). Washington: IEEE, 2019. 149–153. [doi: 10.1109/BigDataSecurity-HPSC-IDS.2019.00036]

[134] Hauser F, Häberle M, Schmidt M, Menth M. P4-IPsec: Site-to-site and host-to-site VPN with IPsec in P4-based SDN. arXiv:

© 中国科学院软件研究所 http://www.jos.org.cn

1907.03593, 2019.
[135] Hauser F, Schmidt M, Häberle M, Menth M. P4-MACsec: Dynamic topology monitoring and data layer protection with MACsec in P4-based SDN. IEEE Access, 2020, 8: 58845–58858. [doi: 10.1109/ACCESS.2020.2982859]
[136] Ricart-Sanchez R, Malagon P, Alcaraz-Calero JM, Wang Q. Hardware-accelerated firewall for 5G mobile networks. In: Proc. of the 26th IEEE Int'l Conf. on Network Protocols (ICNP). Cambridge: IEEE, 2018. 446–447. [doi: 10.1109/ICNP.2018.00066]
[137] Ricart-Sanchez R, Malagon P, Alcaraz-Calero JM, Wang Q. NetFPGA-based firewall solution for 5G multi-tenant architectures. In: Proc. of the 2019 IEEE Int'l Conf. on Edge Computing (EDGE). Milan: IEEE, 2019. 132–136. [doi: 10.1109/EDGE.2019.00037]
[138] Zhao ZP, Sadok H, Atre N, Hoe JC, Sekar V, Sherry J. Achieving 100 Gbps intrusion prevention on a single server. In: Proc. of the 14th USENIX Symp. on Operating Systems Design and Implementation. ACM, 2020. 1083–1100.
[139] Qin QF, Poularakis K, Leung KK, Tassiulas L. Line-speed and scalable intrusion detection at the network edge via federated learning. In: Proc. of the 2020 IFIP Networking Conf. Paris: IEEE, 2020. 352–360.
[140] Scholz D, Gallenmüller S, Stubbe H, Carle G. SYN flood defense in programmable data planes. In: Proc. of the 3rd P4 Workshop in Europe. Barcelona: ACM, 2020. 13–20. [doi: 10.1145/3426744.3431323]
[141] Gondaliya H, Sankaran GC, Sivalingam KM. Comparative evaluation of IP Address anti-spoofing mechanisms using a P4/NetFPGA-based switch. In: Proc. of the 3rd P4 Workshop in Europe. Barcelona: ACM, 2020. 1–6. [doi: 10.1145/3426744.3431320]
[142] Kuka M, Vojanec K, Kučera J, Benáček P. Accelerated DDoS attacks mitigation using programmable data plane. In: Proc. of the 2019 ACM/IEEE Symp. on Architectures for Networking and Communications Systems (ANCS). Cambridge: IEEE, 2019. 1–3. [doi: 10.1109/ANCS.2019.8901882]
[143] Hoque N, Kashyap H, Bhattacharyya DK. Real-time DDoS attack detection using FPGA. Computer Communications, 2017, 110: 48–58. [doi: 10.1016/j.comcom.2017.05.015]
[144] Pham-Quoc C, Nguyen B, Thinh TN. FPGA-based multicore architecture for integrating multiple DDoS defense mechanisms. ACM SIGARCH Computer Architecture News, 2016, 44(4): 14–19. [doi: 10.1145/3039902.3039906]
[145] Liu M, Cui TY, Schuh H, Krishnamurthy A, Peter S, Gupta K. Offloading distributed applications onto smartNICs using iPIPE. In: Proc. of the 2019 ACM Special Interest Group on Data Communication. Beijing: ACM, 2019. 318–333. [doi: 10.1145/3341302.3342079]

附中文参考文献:

[1] 李博杰. 基于可编程网卡的高性能数据中心系统 [博士学位论文]. 合肥: 中国科学技术大学, 2019.
[7] 林耘森箫, 毕军, 周禹, 张程, 吴建平, 刘争争, 张乙然. 基于P4的可编程数据平面研究及其应用. 计算机学报, 2019, 42(11): 2539–2560. [doi: 10.11897/SP.J.1016.2019.02539]
[75] 赵玉宇, 程光, 刘旭辉, 袁帅, 唐路. 下一代网络处理器及应用综述. 软件学报, 2021, 32(2): 445–474. http://www.jos.org.cn/1000-9825/6124.htm [doi: 10.13328/j.cnki.jos.006124]
[123] 戴冕, 程光. 基于sketch的软件定义测量数据平面硬件模型. 通信学报, 2017, 38(10): 113–121. [doi: 10.11959/j.issn.1000-436x.2017203]

赵鹏(1988－), 男, 博士生, 主要研究领域为网络处理器, 网络测量.

赵德宇(1998－), 男, 博士生, 主要研究领域为网络测量, FPGA 网络加速, 网络态势感知.

程光(1973－), 男, 博士, 教授, 博士生导师, CCF 杰出会员, 主要研究领域为网络空间安全监测和防护, 网络大数据分析.

© 中国科学院软件研究所 http://www.jos.org.cn