基于 P4 的可编程数据平面研究及其应用

林耘森箫[1),2),3)] 毕军[1),2),3)] 周禹[1),2),3)] 张程[1),2),3)] 吴建平[1),2),3)] 刘争争[1),2),3)] 张乙然[1),2),3)]

[1)](清华大学 信息网络科学与网络空间研究院 北京市 100084)

[2)](清华大学 计算机科学与技术系 北京市 100084)

[3)](清华大学 北京信息科学与技术国家研究中心 北京市 100084)

摘 要 可编程协议无关报文处理（Programming Protocol-Independent Packet Processors，简称 P4）使管理员能定制交换机的数据包转发行为，提升了数据平面的可编程能力与报文处理的灵活性，为解决当前网络体系结构中长期存在的挑战以及设计新型数据平面功能提供了一种新的解决方案。本文首先概述了 P4 的研究背景，并介绍了 P4 语言及其架构。然后，本文总结了 P4 存在的问题，并介绍了 P4 最新的研究进展，包括异构平台、编译器、开发工具等方面。之后，本文归纳了 P4 在负载均衡、网络测量、网络安全等领域的最新应用。最后，本文探讨了未来 P4 研究工作的趋势。

关键词 软件定义网络（SDN）；可编程协议无关报文处理（P4）；可编程数据平面；领域特定语言（DSL）；编译器
中图法分类号 TP393

Research and Applications of Programmable Data Plane Based on P4

LIN Yun-Sen-Xiao[1),2),3)] BI Jun[1),2),3)] ZHOU Yu[1),2),3)] ZHANG Cheng[1),2),3)] WU Jian-Ping[1),2),3)]
LIU Zheng-Zheng[1),2),3)] ZHANG Yi-Ran[1),2),3)]

[1)](Institute for Network Sciences and Cyberspace，Tsinghua University，Beijing 100084)

[2)](Department of Computer Science and Technology，Tsinghua University，Beijing 100084)

[3)](Beijing National Research Center for Information Science and Technology (BNRist)，Tsinghua University，Beijing 100084)

Abstract Programming Protocol-Independent Packet Processors (P4) enable operators to customize the packet forwarding behavior of the switch, improving the programmability of data plane and the flexibility of packet processing, which provides a new solution for solving long-term challenges in the current network architecture and designing new data plane functions. We first summarize the background of P4, and introduce the P4 language and its architecture. Then, we present flaws of P4 and introduce the latest research progress of P4, including heterogeneous platforms, compilers, development tools, and so on. Next, we demonstrate applications of P4, including load balancing, network measurement, network security, and so on. Finally, we discuss the future research trends of P4.

Key words Software-Defined Networking (SDN); Programming Protocol-Independent Packet Processors (P4); Programmable Data Plane; Domain Specific Language (DSL); Compiler

---

本课题得到国家“十三五”重点研发计划“网络空间安全”专项基础前沿类项目（相当于原973项目）“地址驱动的网络安全管控体系结构及其机理研究”(2017YFB0801700)、国家自然科学基金项目“面向互联网健康诊断的域间互联关系研究”(61472213)资助. 林耘森箫，男，1994年生，博士研究生，主要研究领域为软件定义网络、可编程数据平面. 毕 军（**通信作者**），男，1972年生，博士，长江学者特聘教授，主要研究领域为网络空间安全、软件定义网络、网络体系结构、源地址验证. E-mail: junbi@tsinghua.edu.cn. 周 禹，男，1993年生，博士研究生，主要研究领域为软件定义网络、可编程数据平面. 张 程，男，1987年生，博士，主要研究领域为软件定义网络、可编程数据平面. 吴建平，1953年生，博士，中国工程院院士，主要研究领域为网络空间安全、源地址验证、IPv4与IPv6过渡、网络体系结构. 刘争争，男，1994年生，硕士研究生，主要研究领域为网络体系结构. 张乙然，女，1995年生，博士研究生，主要研究领域为数据中心网络.

1 引 言

新兴网络技术与协议层出不穷，按需定制网络设备的数据包处理行为成为日益迫切的需求[1]。然而，传统交换机的局限性使其难以满足这样的需求：（1）传统交换机内控制逻辑与底层转发硬件紧密耦合，导致新的业务逻辑难以实现；（2）交换机品牌类别多种多样，不同品牌交换机的管理接口封闭独立，因此网络管理员只能通过命令行依次操作每台设备，手动地将高级网络策略转化为设备配置命令并下发到不同品牌的交换机上，配置难度和运维成本随着网络规模急剧上升；（3）报文处理逻辑不可更改，不具备可编程性，只能依靠更换设备的方式来更新升级支持新协议与新功能，并且交换机长达数年的研发周期与高昂的购置成本增加了其支持新协议和新功能的时间成本与资金成本。

因此，如何让网络变得更加开放与可编程成为了学术界和工业界的关键研究方向之一。上世纪 90 年代，DARPA 提出了主动网络这一新型网络体系架构[2]，首次提出了面向定制化服务的可编程网络基础设施的想法，希望通过通用编程接口来管理单个网络节点上的资源(处理器、存储和分组队列等)，允许数据包携带代码来构建自定义功能的数据包处理逻辑。然而，由于需求不明确、协议兼容性差等问题，主动网络并未得到实际部署。世界多国开展过未来网络体系结构相关的研究项目，例如美国的 GENI[3]、欧盟的 FIRE[4]、日本的 JGN2plus[5]，以及我国的 SOFIA[6]等。IETF 提出的 ForCES[7]将网络元素分为控制件和转发件，用 ForCES 协议来实现各部件的协同和交互，以提高网络的管控能力；4D[8]架构重新设计了互联网控制和管理结构，将控制平面与数据平面分离，实现控制平面逻辑中心化与自动化；RCP[9]是基于 AS 结构的逻辑中央平台，通过原型系统验证了 BGP 的路由决策与转发分离架构的可行性；SANE[10]是面向企业网的管理架构，所有路由和接入控制决策都由一台逻辑中央服务器控制。上述众多工作中逐步形成的控制逻辑与数据转发分离的思想与经验为后续的网络研究提供了启迪。

在 2006 年，斯坦福大学的 Martin Casado 博士和他的导师 Nick McKeown 教授领导了一个关于网络安全与管理的项目 Ethane[11]。该项目通过一个集中式的控制器，使管理员能够方便地定义基于网络流的安全控制策略，并将这些安全策略自动地下发到各种网络设备中，从而实现高效的网络安全控制。受此项目启发，斯坦福大学的研究人员们进一步抽象 Ethane 的设计，将传统转发设备的数据平面和控制平面相互解耦，通过集中式控制器以标准化的接口对网络设备进行配置和管理，增强了网络管控的灵活性与支持新协议的能力，那么这将为网络设备的管理和编程提供更多的可能性，从而进一步推动网络创新技术发展。他们在 2008 年首次提出了 OpenFlow[12]的概念，并详细介绍其工作原理，还列举了 OpenFlow 的应用场景，包括校园网络内的协议测试、网络域内的访问控制、网络环境隔离等。随后，基于 OpenFlow 技术，Nick McKeown 等人提出了软件定义网络（Software-Defined Networking, SDN）的概念，引起了学术界和工业界的广泛关注。

OpenFlow 将控制平面与数据平面分离，为网络管理提供统一编程模型，然而学术界与工业界关于 OpenFlow 的研究工作也主要是围绕在控制平面的可编程能力上，对数据平面的关注较少。并且，在 OpenFlow 协议所定义的数据平面模型中，一个重要的问题就是匹配动作表的匹配域是协议相关的，为支持新协议只能变得越发臃肿。具体而言，OpenFlow 支持的匹配域数目随着新版本支持特性的更新而不断增加，从 1.0 版本的 12 个匹配域变为 1.3 版本的 40 个匹配域，最后到 1.5 版本的 45 个匹配域[13]。由于 OpenFlow 缺乏弹性定制匹配域的能力，因此每增加一个匹配域就需要重新编写控制器程序、交换机的协议栈以及交换机转发芯片的数据包处理逻辑，这无疑增加了交换机设计的难度，也严重影响 OpenFlow 协议的可扩展性。

为解决 OpenFlow 自身设计所带来的可扩展性差的问题，Nick McKeown 等人提出了可编程协议无关报文处理语言 P4[13]以及相应的转发模型[14,15]。借助 P4 带来的数据平面编程能力，管理员不仅可以实现诸如网桥、路由器、防火墙等已有的网络设备功能与网络协议，而且还可以很容易地支持包括 VxLAN[16]、RCP[17]在内的新协议。并且，诸如带状态负载均衡[18,19]、大流检测[20,21,22]、分布式计算[23,24]等工作现在也可以通过 P4 在数据平面上实现，从而大幅提升性能。P4 与可编程数据平面的研究引起了学术界与工业界的广泛关注，近几年各大顶级会议和期刊上 P4 相关的学术论文大量涌现，包括谷歌、AT&T、阿里巴巴、腾讯在内的超过 100 家世界知名大型公司加

入 P4 语言联盟[25]。

本文是首篇以基于 P4 的可编程数据平面为核心的研究综述。文中系统地提出 P4 的研究框架，为相关领域的研究者提供了重要的研究参考。其次，本文讨论基于 P4 的可编程数据平面的研究问题、挑战以及现有的解决手段。然后，本文讨论了 P4 为解决现有网络中长期存在的问题带来的机遇，展示 P4 在多个方面的应用。回顾当前基于 P4 的可编程数据平面相关工作，不仅可以了解其发展历程，还能了解其未被充分挖掘的研究重点与潜在的应用场景，为学术界和工业界的下一步研究工作做好铺垫。

本文在第 2 章重点介绍了 P4 语言及其架构，并讨论了 P4 的局限性；第 3 章总结了 P4 相关研究的最新进展；第 4 章系统地梳理了基于 P4 的应用；第 5 章和第 6 章讨论了未来 P4 相关的研究重点和发展趋势并总结全文。

2 P4 语言及架构

本章主要介绍 P4 语言的设计目标、抽象转发模型、工作流程以及语法要素，并讨论 P4 的局限性，然后介绍了其他数据平面编程语言。

2.1 P4 设计目标

P4 的核心设计目标如下：

（1）可重配置性：交换机的数据包处理方式能够被重新配置。由于网络新协议不断涌现，传统交换机只能通过更换设备的方式支持新协议。因此，这个设计目标是为了在不更换交换机硬件的前提下通过编程的方式灵活定义数据平面的报文处理流程。

（2）协议无关性：交换机支持的数据包处理行为不受协议类型局限，并且管理员可以定制交换机本身所支持的协议。由于目前交换机所支持的大量协议并不能在所有场景中得到使用，存在很多冗余协议；同时支持新的网络功能依赖于新协议的定义。因此，这个设计目标是为了让数据平面设备无需关注协议的语法语义内容，让网络管理人员可以去除不需要的协议、快速定义新的协议。

（3）平台无关性：网络管理员能够独立于特定的底层平台来描述报文处理功能，管理员编写的 P4 程序与平台实现无关。由于不同的数据平面平台可能会有相同的报文处理逻辑，存在跨平台移植代码的需求；并且异构的底层平台会提高网络的编程门槛。因此，平台无关性的好处在于使 P4 代码能够跨平台无缝移植，并且网络管理员可以像编写 C 或者 C++程序那样不用关注底层架构，从而减轻网络管理员负担。

2.2 P4 抽象转发模型

为了实现上述目标，在 OpenFlow 已有的匹配+动作模式基础之上，P4 的抽象转发模型如图 1 所示。

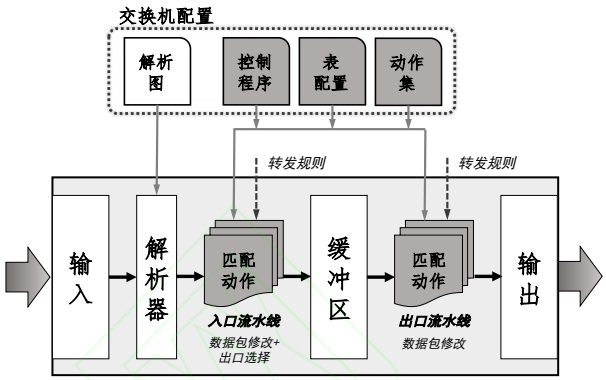

图 1 P4 抽象转发模型[13]

从组成部分上来看，该抽象转发模型主要包含三部分——解析器、多级流水线和缓冲区：

A. 解析器。报文首先经过一个可重配置解析器，解析器的功能是将头部域从报文中提取出来，按照数据包头解析图解析，余下的载荷与头部分开缓存，并且载荷不参与后续匹配。管理员可以定制数据包头结构和解析流程，解析流程会被编译器编译为数据包头解析图并配置到解析器上。

B. 多级流水线。（1）从空间上来看，被提取出来的头部经过多级匹配动作表，这些多级匹配动作表以流水线的形式组织起来，分为入口流水线和出口流水线两部分。位于入口流水线的匹配动作表决定了报文的输出端口与队列，基于入口流水线的处理，报文可能被转发、复制、丢弃或者触发流控；位于出口流水线的匹配动作表主要负责修改报文头部。（2）从逻辑上来看，流水线是由匹配动作表组成的一个有向无环图（Directed Acyclic Graph，DAG），这个 DAG 也被称作数据平面控制流。在编写 P4 程序时，管理员可以按照 P4 语法规范去定义控制流以及每张匹配动作表要匹配什么样的数据包、执行什么样的动作，从而达到自定义数据平面流水线处理逻辑的目的；在运行时，数据平面则会按照定义的控制流处理数据包。此外，报文头部在各级匹配动作表之间传递的过程中，可以携带被称为元数据的额外信息，具体来说，元数据可以包括入端口信息、时间戳等等。

C. 缓冲区。缓冲区用来缓存载荷与交换机队列

中等待被匹配动作表处理的已解析的头部。

在抽象转发模型方面，P4 与 OpenFlow 主要不同点在于：（1）OpenFlow 仅支持固定协议的解析器，而 P4 支持可编程的解析器来自定义新的头部；（2）OpenFlow 假定多级匹配动作表是顺序执行的，而 P4 的匹配动作表可以是并行或者顺序执行的；（3）OpenFlow 假定动作是一系列固定的动作，而 P4 的动作可以是由交换机所支持的协议无关原动作组成的复合动作。

2.3 P4 工作流程

对于 P4 程序而言，其工作流程可以分为两个相互独立的阶段——配置阶段和运行阶段：

A. 配置阶段决定了交换机支持何种协议以及交换机如何处理报文。在该阶段中，管理员按照 P4 语言规范编写 P4 程序、定义数据平面行为，包括数据包头解析器定义、控制流定义、匹配动作表定义、用户元数据定义等。P4 程序编写完成后，管理员可以利用 P4 编程语言的前端编译器将 P4 程序编译成一种高级中间表示（High Level Intermediate Representation，HLIR）[26]。然后，后端编译器会将 HLIR 文件编译到具体硬件或者软件平台上，编译器除了配置数据平面之外，还会生成供控制平面在运行时调用的运行时控制管理接口。

B. 运行阶段决定了网络策略在指定的时间作用在报文上。在该阶段中，数据平面会按照 P4 程序定义的逻辑处理数据包，而运行在控制平面的应用程序会调用配置阶段生成的运行时管理控制接口，实现运行时向数据平面下发匹配规则。

2.4 P4 语法要素

P4 语言是一种能够实现可重配置性、协议无关性、平台无关性的数据平面编程语言，其中语法要素是 P4 提供给管理员来描述基于抽象转发模型的数据包处理行为的手段。目前 P4 语言规范包括 2014 年发布的 $P4_{14}$[27]与 2016 年发布的 $P4_{16}$[28]。以下为 P4 语言的语法要素，表 1 为部分 P4 语法要素示例：

（1）头部（Header）。头部结构定义了数据平面需要解析的数据包头格式，包括数据包头所含域的结构、宽度和值的限定等。头部的内容决定数据包后续的操作。

（2）表（Table）。表的格式为匹配+动作，即匹配域和相应的执行动作。

（3）动作（Action）。动作用来描述数据包头部和元数据如何被处理。P4 定义了一套协议无关的原动作（Primitive Action），包括丢包、设置字段、复制包头等。并且，管理员可以自定义一个由多个原动作组成的复合动作（Compound Action）。

（4）控制流（Control Flow）。控制流是匹配动作表组成的有向无环图，定义了数据包在不同的匹配动作表中的跳转关系。其跳转逻辑包括顺序逻辑、判断逻辑等，但是不允许一张匹配动作表对同一个数据包执行多次匹配。

（5）解析器（Parser）。解析器定义了如何鉴别数据包头部以及数据包头部域的有效顺序，用于指导数据平面上的物理解析器按照何种逻辑去解析数据包。

（6）逆解析器（Deparser）。逆解析器是当数据平面多级流水线处理完数据包后，将处理完的数据包头部进行重组的结构。

（7）用户自定义元数据（User-defined metadata）。用户自定义元数据是用户自定义的与每个报文相关的数据结构，其中可以存放用户定义的值、临时变量等。用户自定义元数据能够有效地支持数据平面带状态处理，数据包在各级匹配动作表之间传递时，可以携带这些用户自定义元数据以完成匹配动作表间的参数传递。

（8）固有元数据（Intrinsic metadata）。固有元数据是由数据平面自身产生或者消耗的信息，例如收到报文的入端口信息、报文将被转发的出端口信息等。可以利用读取或者设置固有元数据来完成一些数据平面预定义的功能，例如多端口洪泛等。

（9）外部对象*（Extern object*）。外部对象是事先定义好的库函数结构，P4 程序可以通过 API 对其进行调用。但是其内部行为是硬件实现的（例如：校验和单元），不能使用 P4 语言序来编程。

（10）结构定义*（Architecture definition*）。结构定义是一系列用来描述可编程网络设备的声明。

注：末尾两个带*号的要素为 $P4_{16}$ 新添加的。

表 1 部分 P4 语法要素示例

语法要素	$P4_{14}$ 程序示例	$P4_{16}$ 程序示例

头部	header ethernet_t{
	fields{
	dstAddr : 48;
	srcAddr : 48;
	etherType : 16;
	}
	}
		header ethernet_t {
		bit<48> dstAddr;
		bit<48> srcAddr;
		bit<16> etherType;
		}

表	table forward_table() {
	reads{
	ethernet.dstAddr:
	exact;
	}
	actions{
	forward1;
	NoAction;
	}
	size : 32;
	}
		table forward_table() {
		key = {
		hdr.ethernet.dstAddr:
		exact;
		}
		actions = {
		forward1;
		NoAction;
		}
		size = 32;
		default_action=NoAction();
		}

动作	action forward(port){
	modify_field(standard_
	metadata.egress_spec,
	port);
	}
		action forward(bit<9> port) {
		standard_metadata.
		egress_spec = port;
		}

控制流	control ingress{
	apply(forward_table);
	}
		control ingress(inout headers
		hdr, inout metadata meta) {
		apply {
		forward_table.apply();
		}
		}

解析器	parser TopParser{
	extract(ethernet);
	return
	select(latest.ethertype){
	0x0800: ipv4;
	default: ingress;
	}
	}
		parser TopParser(packet_in b,
		out Parsed_packet hdr){
		state start {
		transition parse_ethernet;
		}
		state parse_ethernet {
		b.extract(hdr.ethernet);
		transition
		select(hdr.ethernet.etherType){
		0x0800: parse_ipv4;
		default: accept;
		}
		}
		}

2.5 P4 语言的局限性

P4 作为数据平面的领域特定语言（Domain Specific Language, DSL），虽然能够提升数据平面的可编程能力、实现一些以前未曾在数据平面实现的功能，为网络数据包处理带来极大的便利，但是由于其表达能力有限以及数据平面本身的局限性，P4 并非是目前网络领域所有问题的万能解决方案。了解 P4 语言及其编程模型存在的局限性，不仅能够为 P4 找到更加适合的应用场景，而且能够为改进 P4 语言、寻找 SDN 下一代解决方案提供新的思路。

直到 $P4_{16}$ 版本发布，P4 语言仍然存在以下的局限性：

（1）语法上的局限性

• P4 程序不支持循环解析。没有专门的循环函数，解析器只能通过有限状态机来支持循环。

• P4 程序不支持动态内存分配。资源消耗需要在编译时静态估计，会导致资源利用不充分。

• P4 程序不支持指针或者引用。

• $P4_{14}$ 程序通过 counter、register 和 meter 来维持跨越不同数据包之间的状态，但是 $P4_{14}$ 无法遍历所有计数器来计算统计信息；$P4_{16}$ 为了维持跨越不同数据包之间的状态，需要使用外部方法（extern method）。

（2）功能上的局限性

• P4 程序自身不能支持多播与广播，需要通过设置特定的固有元数据表示“广播组”来触发 P4 的外部机制，执行所需的报文复制。

• P4 程序不能支持描述队列、调度或者多路复用。

• P4 程序不适合做深度包检测。一般而言，很多对报文载荷做操作的事情 P4 都不能完成。

• P4 程序不支持报文的分段和重组，因此 P4 程序不能实现 TCP 协议的所有功能。

• P4 程序不支持产生新的报文（例如：ICMP reply），只能处理已有的报文。

（3）标准规范上的局限性

• 目前数据平面和控制平面之间没有标准的通信方式，目前通常使用自定义的外部方法（例如：learning）来完成。

如何将编程语言进一步抽象、完善数据平面编程语言的语义、更好地支持模块化、更好地利用数据平面有限的资源、在保证线速处理转发的基础上支持更丰富的数据包处理逻辑，都是未来对数据平面编程语言和编程模型研究中需要考虑的。

2.6 其他数据平面编程语言

除了 P4 语言，其他数据平面领域特定语言也可以描述数据平面的报文处理逻辑。以下是除 P4 以外其他主流 DSL 的介绍以及与 P4 语言的区别：

（1）POF[29]将数据包头视为{偏移量，长度}元组，从而生成类似汇编语言的低级编程语言。虽然 POF 能够简化编译器，但是把数据包解析的负担转嫁给了管理员，并且没有提供一个简单易用的编程模型来描述数据包头和解析器。

（2）Click[30]使用 C++编写的模块来构建交换机，富有表现力，适合表达数据包在 CPU 中的处理逻辑。但是 Click 不能满足“控制器—交换机”的架构需求，无法将“解析—匹配—动作”的流水线操作映射到指定的硬件上，无法描述交换机上的数据包处理逻辑。

（3）packetC[31]允许访问数据包载荷、允许通过为全局共享内存提供同步结构来实现带状态处理。但是 packetC 着眼于 NPU 以及软件交换机等灵活性更高但是性能较低的设备，很难满足诸多网络场景下高吞吐的线速转发需求。

（4）PX[32]是面向 FPGA 平台（例如：Xilinx Virtex-7）的数据包处理语言，将数据包解析和处理的高级声明规范转换为 Verilog 或 VHDL 中目标基板的寄存器传输级（Register-Transfer Level，RTL）描述，然后通过标准 FPGA 工具链将此 RTL 描述进一步编译为 FPGA 的可执行文件。然而，PX 仅将 RTL 作为输出，仅能运行在 FPGA 平台上，无法满足平台无关性的需求。

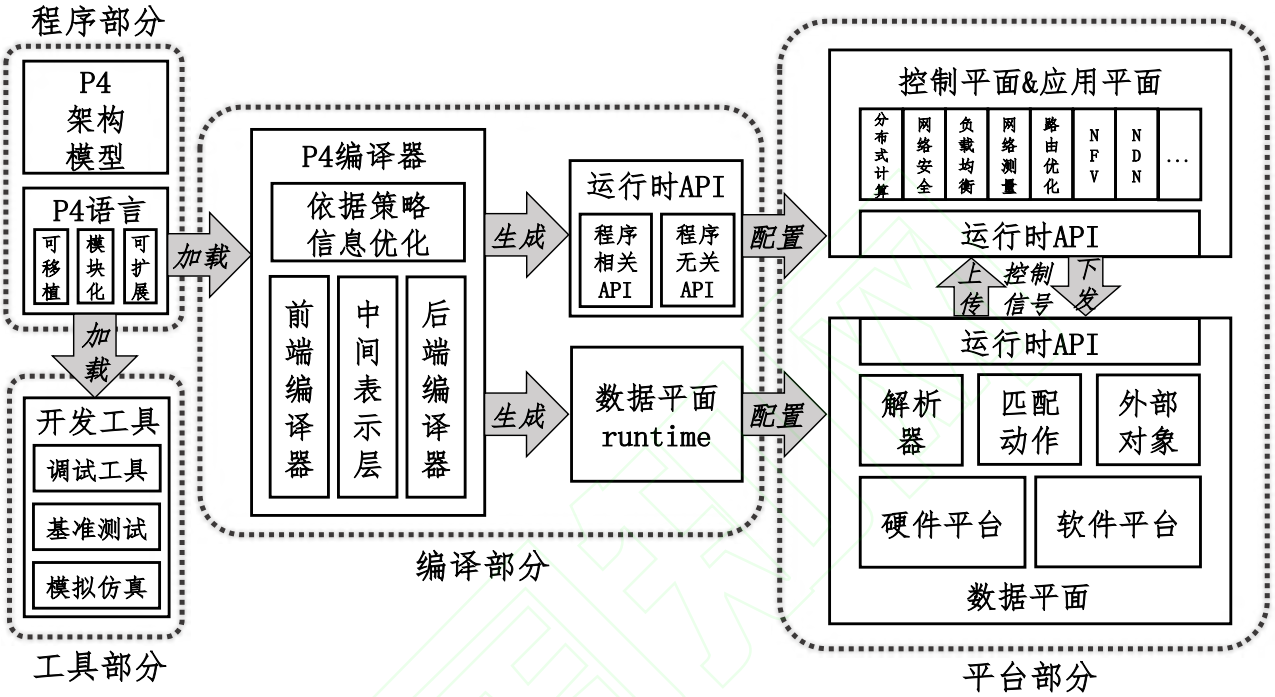

图 2 P4 研究框架

3 P4 相关研究的最新进展

自定义数据平面报文处理逻辑，增强数据平面可编程能力，使网络设备灵活支持各种新协议和新功能是学术界和工业界的美好愿景与努力的方向，P4 语言与可编程数据平面正是朝该目标迈进的一大步。但是，对于如何使硬件平台或者软件平台支持现有的 P4 语言、如何设计并优化编译器、如何对 P4 程序进行调试与性能测试等问题，P4 语言联盟[25]与 P4 语言规范[27,28]并没有给出相应的解决方案。学术界和工业界对 P4 语言与可编程数据平面的落地和优化作出了一系列相关研究，本章将分门别类地逐一介绍。如图 2 所示，以 P4 工作流程为依托，本章系统地提出了 P4 的研究框架，并将本章的内容组织了起来。

3.1 异构平台与编译器设计

可以用来做网络数据包处理的数据平面不仅包括可编程交换机、FPGA、GPU 等硬件平台，还包括软件交换机在内的软件平台。不同的平台有着各自不同的特点：例如，可编程交换机这类硬件平台的性能较高，但是包括软件交换机在内的软件平台的灵活性更强。因此，负责将 P4 程序编译到不同平台的后端编译器设计成为了一个重要的研究点。

3.1.1 硬件平台与后端编译器

（1）在可编程交换机上运行 P4 程序

文献[33]针对 P4 语言首次为 RMT[14]，Intel Flexpipe[34]以及 Cavium XPliant[35]等可编程交换机芯片提出了编译器设计方法，该工作主要解决了可编程交换机芯片中的匹配动作表配置问题。首先，该工作系统地研究了可编程芯片中的结构和资源限制，并对 P4 程序中决定匹配动作表相对位置的依赖关系进行了深入的分析。在上述分析的基础上，该工作提出了匹配动作表的映射问题，并结合已有的贪心算法和整数线性规划算法解决该问题中存在的四类限制：针对单个逻辑表的流表项分配

限制、针对物理表的内存容量限制、针对逻辑表之间的依赖限制以及可编程芯片特有的限制。该工作针对贪心算法和整数线性规划算法进行了综合地实验分析，其实验结果表明贪心算法拥有较高的运行效率，但是针对不同目标（如优化资源、优化延迟等）的优化效果弱于整数线性规划。

（2）在 FPGA 上运行 P4 程序

由于网络数据包需要被线速处理，因此对硬件平台的性能有着较高要求；并且，在数据中心和一些企业网中，需要修改网络策略或者部署新的协议来满足实际的生产需求，因此，数据包处理对硬件平台的灵活性与可编程性提出了新的要求。由于 FPGA 比 CPU 有更高的性能，比 ASIC 有更强的灵活性，因此将 FPGA 用来做数据包处理的硬件平台可以很好地兼顾性能与灵活性。目前，包括微软、亚马逊、百度在内的许多云服务提供商已经在各自数据中心里部署了 FPGA 来提高应用的性能[36]。

但是，FPGA 使用低级语言编程，上手难度较大，调试非常复杂，并且无法做到不同平台间的移植。于是，文献[38，39]设计了报文解析器生成模型以及相应的转换算法，可以将 P4 的解析描述图转化为适合在 FPGA 上部署的合成 VHDL 代码。其中，报文解析器硬件架构由一组可重配置解析器引擎以流水线方式构成，生成的解析器支持对 100Gbps 流量的线速解析，但是与手写的 VHDL 相比，在延迟和资源消耗方面增加了一倍性能开销。

随后，Han Wang 等人[36,37]设计了一个可以将完整的 P4 程序编译成 FPGA 程序的开源编译器，将程序员编写的 P4 代码先编译为 Bluespec 代码，然后生成 Verilog 代码。编译生成的代码可以线速运行在 Xilinx 或者 Altera 等多种商用 FPGA 硬件上，为 FPGA 上新协议和应用的开发提供了一个更加灵活、高效、可移植的编程抽象，降低了 FPGA 的开发门槛，缩短了开发周期。但是，生成的解析器的吞吐会随着所支持协议栈的复杂度的增加而减少。

于是，文献[40]针对以前工作在性能和资源利用上的不足，提出了将 P4 程序高效转化到 FPGA 上的开源框架和相关算法，将延迟降低了 45%、查表的资源占用降低了 40%。Michal Kekely 等人[41]为了能够兼顾较快的处理速度与较高的内存资源使用，基于 DCFL 算法设计了将 P4 匹配动作表映射到 FPGA 的新硬件架构。更进一步，Jakub Cabal 等人[42]为了对数据包做更快的解析以实现更高的吞吐，提出了一个可以由 P4 语言重配置的解析器架构，将复杂的报文解析划分为多个简单的并行报文解析，首次实现了能够在单个 FPGA 上达到 1Tbps 的吞吐，并且能够在小包突发流的情况下保证线速转发，只引入了很小的硬件资源开销。

（3）在 GPU 上运行 P4 程序

为了充分利用 GPU 多核并行计算的优势，提升 P4 程序的执行效率，文献[43,44]研究了将 P4 程序映射到 GPU 与 CPU 组成的混合架构上。该工作的核心思想是将数据包的解析操作在 CPU 上执行，将数据包的匹配动作操作在 GPU 上并行执行，并且做了如下的优化：对 GPU 内核设计做了优化以支持匹配动作的原语、提出了延迟隐藏技术来缩小 CPU/GPU 间通信开销、设计了负载均衡策略来最大化 CPU 与 GPU 的资源利用率。具体的工作流程如下：在配置阶段将 P4 程序转化为中间表示形式，然后使用正则表达式从 IR 中找到所有表的正确执行顺序，最后将匹配动作表加载到 GPU 中；在运行阶段，数据包通过直接内存访问从网卡加载到内存，然后 CPU 控制数据包以批处理的方式从内存按照一定比例分别加载到 GPU 与 CPU，最后 GPU 与 CPU 将各自处理完的报文返回至内存，交付给网卡。实验结果表明该系统可以实现较高的查找速度与较小的报文延迟。

（4）抽象编程模型的研究

P4 使得交换机支持带状态数据包处理，使以前需要在控制平面运行的一些处理逻辑可以直接实现在数据平面。然而，想要高效正确地管理分布式的、带状态的数据平面是很有挑战的。针对此问题，文献[45] 提出了一个带状态编程模型——SNAP。通过 SNAP，编程人员可以在“一个大交换机模型”（One Big Switch）上进行编程，只需要针对一个交换机定义最基本的数据包处理逻辑，该系统会自动将程序部署到多个交换机并完成存储、通信等协调工作。SNAP 的编译器会检查各状态间的读写依赖，并将 SNAP 程序转化为扩展转发决策图（extended Forwarding Decision Diagrams, xFDD），之后使用混合整数线性规划（Mixed-integer Linear Program, MILP）来联合优化状态存储和路由，最后据此生成各个交换机的具体配置。

3.1.2 软件平台与数据平面虚拟化

（1）软件平台

可编程数据平面除了上述硬件平台，还包括软件交换机在内的软件平台，目前有一部分研究工作关注利用 P4 来描述软件平台的数据包处理逻辑，

并提供相应的编译器设计。

在数据中心里，管理程序（Hypervisor）通常会使用软件交换机来完成虚拟机之间的通信。但是，通常情况下软件交换机的实现基于大量的内核代码，如果想要更改软件交换机的某些行为，需要对其实现细节非常的了解。因此，软件交换机的升级和对新协议的支持是一个复杂而艰难的过程。

针对这一问题，文献[46]基于如今最流行的软件交换机 Open vSwitch（OVS）[47]提出了可编程的、协议无关的软件交换机架构 PISCES[46]，使得软件交换机的数据包转发行为可以使用 P4 语言进行描述，将定制协议的行为与底层软件交换机的代码解耦，只需要修改 P4 程序就能支持新协议、添加新功能。PISCES 将写好的 P4 程序经过编译器编译产生能够在 OVS 上运行解析、匹配和动作的代码。PISCES 的性能与 OVS 基本接近，但是实现相同功能的 PISCES 比 OVS 节省约 40 倍的代码量。

对于 DPDK[48]这样的快速包处理套件，有工作[49]研究了针对 DPDK 的 P4 编译器模型设计，该编译器模型将软件交换机模型抽象分为平台相关部分与平台无关部分。该编译器仅需负责将 P4 程序编译到平台无关部分，而无需考虑平台相关部分，但是生成平台无关部分的代码之后，需要重新对软件交换机进行编译。p4c-bmv2[50]是 P4 语言联盟提供针对 P4 行为模型的编译器，它可以生成 P4 行为模型所需的 JSON 配置文件。在这种编译模式下，作为软件交换机的 P4 行为模型无需再重新编译。针对 P4 到软件交换机的编译设计，文献[51]利用了预取和批处理两种方式对基于 P4 程序编译生成的软件交换机性能进行了优化，使得 P4 软件交换机编译器生成的代码的执行效率能够接近经过手工代码优化的软件交换机。

对于 VPP（Vector Packet Processing）这样的基于 x86 CPU 架构开发的高性能数据包处理软件平台，有工作[52,53]设计并实现了将 P4 程序编译到该平台上。在 VPP 中，报文以批处理的形式实现每一级流水线的处理，这使得 VPP 具有较高的并行处理能力和较高的可扩展性。由于该工作将 P4 程序中的表映射到了 VPP 中的节点处理，因此使得基于该工作开发 VPP 程序变得更加简单高效。

更进一步，开放数据平面（OpenDataPlane，ODP）为数据平面编程提供了统一的抽象接口，使得基于 ODP 规范 API 开发的数据平面程序可以移植到不同厂商生产的不同软件平台上（例如：x86, x86+DPDK, ARM-SoC 等）。文献[54,55]为 P4 程序实现了一个从 P4 语言到 ODP 的编译器 MACSAD，从而借助 ODP 项目的多平台支持性，使得 P4 程序可以无缝的迁移到大量支持 ODP 接口的设备上。该工作设计并实现了从 P4 程序到 ODP 规范 API 的编译器,主要包括三部分：辅助前端插件以插件的方式支持不同 DSL 的前端编译器；核心编译器将不同 DSL 前端编译器生成的中间语言统一编译为 ODP 支持的 API；辅助后端插件负责屏蔽不同设备平台的实现，为核心编译器提供统一的底层接口。

（2）数据平面虚拟化

最初，单个 P4 可编程交换机提供的数据平面只能支持单个网络环境，无法在单个网络设备上同时支持多租户。Hyper4[56]在单个物理数据平面上虚拟出多个可编程数据平面，不同的网络功能可以同时在相同的物理数据平面上运行，允许在运行时修改程序以及连接它们的虚拟网络，而不会中断当前活动的程序。具体来说，Hyper4 在交换机上部署一个 P4 程序 hp4.p4，通过把实际需要执行的功能 foo.p4 等效翻译成 hp4.p4 程序的表项，通过数据平面管理单元对不同虚拟数据平面进行隔离。

但是由于缺乏结构化的设计，Hyper4 中的数据平面只能被部分虚拟化，并且 Hyper4 不支持包括布尔表达式在内的重要元素，使得 Hyper4 不适用于大多数 P4 程序。此外，Hyper4 使用了大量的重投递操作，使得 Hyper4 遭受严重的性能损失，以及硬编码的实现方式也导致硬件资源被过度消耗。

受 Hyper4 的启发，HyperV[57]提出了具有结构化设计和抽象模型的虚拟化数据平面的解决方案。HyperV 采用控制流排序和动态阶段映射技术，可以实现数据平面的完全虚拟化，使得任意 P4 程序都可以被部署到虚拟数据平面上。并且，HyperV 通过抽象出 stage slot 的模型，可以容纳任意数量的 stage，不同程序的不同 stage 可以分配到同一个 slot，因此大大提升了资源使用效率。此外，HyperV 提出了快速解析和绕过流水线技术，进一步降低了虚拟化所带来的性能影响。因此，HyperV 可以实现虚拟化和性能之间的较好折衷。与 Hyper4 相比，HyperV 实现了更全面的数据平面虚拟化，在提供 2.5 倍性能提升的同时节省了 4 倍的资源使用。

在实际的生产环境中，不同的网络功能可能会存在冲突，并且网络功能的动态管理需要中断数据平面的操作，增加了网络功能的操作开销（OPEX）。为此，MPVisor[58]使用高性能的数据平面管理程序

与模块化可编程 API 来使得可以在不中断数据平面操作的同时实例化新的网络功能、实施策略隔离。

3.2 编译中间表示与编译器优化

除了针对不同平台进行相应的编译器设计，在编译过程中对 P4 程序进行优化，实现对数据平面资源的高效利用也是重要的研究方向。

3.2.1 编译中间表示设计

由 Python 对象表示的高级中间表示 HLIR 可以由 P4 语言联盟提供的前端编译器 p4-hlir[26]生成。HLIR 可以便捷地转换为其他语言，是包括 p4c-bmv2 在内的众多后端编译器的输入。p4-hlir 已经成为分析 P4 程序的重要工具。

此外，文献[59]为包括 P4 在内的多种领域专用语言提供了中间表达语言 NetASM，并且 NetASM 可以通过平台特定的后端编译器编译运行在不同平台上。在整个可编程数据平面编译架构中，NetASM 充当了后端编译器与前端编译器的桥梁，NetASM 一方面具有足够的表达能力，满足各种面向可编程数据平面的领域特定语言的表达需求；另一方面 NetASM 也具有表达底层指令的能力，能够兼容底层平台的特性。

3.2.2 基于策略信息的编译优化

P5[60]创新性地提出了基于上层策略信息的 P4 程序编译优化。P5 实现编译优化的主要原理是网络流量仅仅需要被特定的交换机功能所处理（例如：有些数据包仅仅需要被 L2 转发功能处理，而有些数据包则需要被 L3 路由以及防火墙功能所处理）。网络流量对于交换机功能需求的差异性一方面使得在网络中一些特定的功能无需被所有交换机支持，另一方面可以通过编译过程中重组交换机功能执行顺序的方式消除某些功能之间的依赖关系，在满足网络流差异化需求的前提下实现对 P4 程序的优化。因此，P5 能够提升 P4 程序的并行度，同时减少所需的匹配动作表的级数，降低 P4 程序的部署要求。但是，P5 仅仅能够根据历史策略信息进行编译时优化，当网络流量的策略的需求发生变化时，P5 无法做出动态调整以满足网络流量策略的需求。

表 2 不同平台编译器对比

代表工作	编译器类型	输入	输出	针对的平台	平台相关优化	编译器实现方式	是否开源	主要技术特点
p4-hlir[26]	前端编译器	P4	HLIR	*	无	Python	开源	P4 语言联盟提供的前端编译器
p4c-bmv2[50]	后端编译器	HLIR	JSON	bmv2	无	C 语言	开源	P4 语言联盟提供的后端编译器
Jose et al.[33]	后端编译器	HLIR	交换机配置	可编程交换机	使用 ILP 和贪心算法优化延迟、流水线占用和功耗	CPLEX	未开源	解决了可编程交换机芯片中的匹配动作表配置问题
P4FPGA[36]	后端编译器	HLIR	Bluespec	FPGA	利用硬件与语义的并行来优化吞吐和延迟	C++	开源	简化 FPGA 开发，融合 P4 的灵活性和 FPGA 的高性能
P4GPU[43]	后端编译器	HLIR	内核配置	CPU/GPU	通过延迟隐藏技术减小 CPU/GPU 通信开销	Python	开源	充分利用 GPU 多核并行计算的优势，提升 P4 程序的执行效率
SNAP[45]	前端编译器	SNAP	NetASM	分布式带状态数据平面	使用 MILP 优化路由和交换机配置	Python	开源	提供了“一个大交换机”抽象模型来管理分布式带状态数据平面
PISCES[46]	前端+后端	P4	C 语言	OVS	通过修改 IR 优化 C 语言代码	扩展 p4c-bmv2	开源	简化了定制与维护软件交换机数据包处理逻辑的难度
Laki et al.[49]	后端编译器	HLIR	C 语言	DPDK	无	Python	开源	将模型分为平台相关和平台无关两部分
PVPP[52]	后端编译器	JSON	C 语言	VPP	使用 VPP 暴露出来的底层接口优化 PVPP	Python	开源	编译器对底层程序指令有更细粒度的控制
MACSAD[54]	后端编译器	HLIR	MacS	ODP	使用核心编译器优化代码	扩展 GCC	开源	兼顾性能和可移植性，实现多平台间无缝移植

3.3 对 P4 语言进行扩展的研究

网络管理员可以使用 P4 语言在网络设备上定义数据平面的行为，实现新的网络功能。然而随着需要实现的功能数量的增加，P4 程序的大小和复杂性也在增加，这给开发和管理 P4 程序带来了挑战。并且，在运行阶段，大量有相同行为的流会重复匹配同一个匹配动作表项，带来转发速率和吞吐量上的开销。于是，可以通过扩展 P4 语言来解决或者改善上述问题。

3.3.1 P4 模块化编程研究

实践表明，为了满足大量不同的网络策略的需求，程序需要复杂的设计，开发难度高；并且，在 P4 程序部署之后，不能根据实际的策略需求变动，不能实时动态地编排网络功能，灵活性差；此外，P4 程序具有可自定义流表结构和控制流的灵活性，使得在大量可编程设备上，管理流表项和数据平面行为非常复杂。

ClickP4[61]提出了模块化的编程模型和支持动态编排网络功能的数据平面结构来解决上述问题。管理员不需要直接开发复杂的 P4 程序，只需要开发功能模块，把所有需要的模块和库注册到配置文件中，ClickP4 就可以将其编译集成到 ClickP4 代码中，降低了开发难度。在数据平面上，ClickP4 按流水线的形式组织网络功能，通过流水线头部和尾部的初始化器和倒带器维护状态机和对应于每个报文的令牌，可以在运行时为不同的流构建不同的功能链，实现动态编排。管理员可以使用 ClickP4 提供的一套编排描述原语便捷地管理、编排网络功能。但是，ClickP4 在引入模块化设计的同时也带来了性能上的损失。

3.3.2 P4 转发匹配表缓存机制研究

P4 通过将一系列匹配动作表（MAT）以流水线形式依次作用于报文。即使第一个报文通过匹配 MAT 就可以确定该流的行为，之后的报文仍然需要在所有 MAT 中进行头部域匹配查找并依次执行动作，既消耗了时间又消耗了数据平面资源。

CacheP4[62]采用在流水线之前设计缓存表的方法优化上述问题。报文如果匹配缓存可以直接按照表中存储的复合动作完成处理，那么就跳过后续流水线的步骤，从而提升了 P4 设备的数据包转发速率和吞吐量。在配置阶段，CacheP4 会先对原 P4 程序进行预处理分析，生成缓存表的结构，并输出带缓存的 P4 程序，然后编译配置到 P4 设备中；在运行阶段，控制平面根据需要缓存的报文头部信息和 P4 设备中已有的流表项，计算并填入新的表项，完成选定流的性能加速。实验表明，CacheP4 可以进一步提升网络性能，并且优化收益会随着跳过的匹配动作表数目的增加而增大。但是，在缓存未命中的情况下，CacheP4 比无缓存时的性能表现更差。

3.4 P4 开发工具

3.4.1 调试开发工具

P4 作为数据平面编程语言，在加速网络创新、降低网络开发准入门槛的同时，也为数据平面引入了更多的软件错误。同传统的软件编程语言一样，想要保证程序的正确性就需要一套针对特定语言的调试开发工具，但是传统成熟的 ASIC 等硬件调试开发工具无法被直接应用到 P4 上。目前主要有两种研究方向，一种是直接从 P4 语言模型自身出发，从头构建符合语言特性的调试开发工具；另一种则是将 P4 语言模型转换为其他的成熟语言模型，直接使用该语言模型的调试开发工具。

（1）针对 P4 语言模型本身的调试工具

p4pktgen[63]通过符号执行来生成包括测试包、表条目和预期路径在内的 P4 程序测试用例，并通过 bmv2[64]验证这些测试用例。p4pktgen 在使用深度优先遍历访问节点来生成路径时，依据上下文处理前缀并及时进行回溯尽早丢弃搜索空间中不可行的部分，可以对大型 P4 程序生成有效的测试用例，展现静态 P4 程序中存在的错误。

然而，在部署 P4 程序之后，运行期间也可能会出现错误，并且运行时的错误具有多样性、复杂性、不可见性等特点，让静态调试工具很难发挥作用。P4DB[65]提出了 P4 程序的运行时调试方案，提供了网络级、设备级和流表级的调试视图。管理员可以通过 P4DB 提供的 watch, break 和 next 等原语来进行调试，但是会带来吞吐下降与延迟。

（2）将 P4 语言模型进行转换的调试工具

文献[66]提出了被称为 ASSERT-P4 的调试方法，通过在 P4 源程序中人工插入断言语句标注来反映所预期的网络行为。之后，添加了标注的 P4 源程序和附加的流表规则信息被翻译成 C 语言程序，最后通过符号执行来检测是否违反断言从而检测源程序的正确性。该方法能够检测出静态 P4 程序中的隐晦错误，但是符号执行会带来指数级的时间复杂度，限制了验证特性的数目。

类似的，文献[67]也是采用线下静态程序分析，将 P4 程序翻译成操作语义集，在合并去冗余优化后转化为 Datalog 要求的 DNF 格式规则供网络管理

员进行监测，自动验证网络数据包的可达性。

3.4.2 基准测试工具

随着 P4 语言的发展，有许多将 P4 程序编译到不同平台的编译器陆续被提出，但对于这些不同的编译器，缺乏一个统一的评价标准。针对这个问题，Whippersnapper[68]提供了一套综合 P4 语言各特性的评价基准，为评估不同 P4 编译器的设计与实现提供了一套通用的指标。

为适应不同平台，Whippersnapper 分为平台相关基准和平台无关基准两部分，使用延时、吞吐量、内存占用等不同指标，对 P4 在编译器中的解析、处理、状态访问、数据包修改等不同特性分别进行评估，便于对 P4 编译器进行分析和评价反映出这些编译器在性能上的差异。

3.4.3 模拟仿真工具

对可编程数据平面相关的架构及应用进行快速模拟验证和分析，能极大提高研究开发的效率。PFPSim[69]提供了这样的模拟器，使用者可以自己定义可编程数据平面设备的基础架构以及运行于其上的数据包处理模块，以此完成对可编程数据平面的底层设备架构、数据包处理逻辑与应用程序的模拟、验证和评估。PFPSim 通过一种抽象的转发架构描述语言来帮助使用者定义适用于目标应用的底层架构，在此基础上，操作者可以用 C++和 P4 来定义不同的模块来完成数据包转发等功能。然后，PFPSim 提供的编译器来将不同代码及操作者输入的配置文件统一编译为可执行的二进制模型，最后可以通过数据包生成器向模型发送模拟数据流来进行模拟。但是，PFPSim 无法模拟包含多个 P4 设备的大规模网络。

同样作为可编程数据平面的模拟仿真工具，NS4[70,71]更注重对包含多个可编程数据平面设备的大规模网络的模拟。NS4 提出了一个基于离散事件的模拟器，在 ns-3 网络模拟器基础上，增加了模拟 P4 交换机的模块，并增加了对应的控制器模块。操作者可以使用 ns-3 的接口定义网络拓扑，其中可以包含多个 NS4 提供的 P4 交换机模块，同时使用 P4 来定义数据包的处理行为，并加载相应的模块。相比其它模拟器，NS4 可以模拟多个可编程交换机协同工作的情形，并且由于 NS4 占用的资源更少，因此可以模拟更大规模的网络。

表 3 基于 P4 的应用

应用领域	解决的问题	基于 P4 的应用
负载均衡与资源分配	二层三层负载均衡	HULA[73]
	传输层负载均衡	SilkRoad[18,19]、Beamer[76]、NDP[78]
	应用层负载均衡	NetCache[80]、AppSwitch[81]、NetChain[82]
	资源分配协议的近似实现	Sharma N K [83]、Sharma N K[84]、Cascone C[85]
网络测量、监控与诊断	通用方法	FlowRadar[87]、UnivMon[88]、INT[89]
	大流检测	HashPipe[20,21]、Harrison R[22]
	网络故障诊断	Dapper[94]、LossRadar[95]、KeySight[96]
	监控查询语言	Marple[97,98]、Sonata[99]
网络安全	防御 DDoS 攻击	Afek Y[106]
	防止监控结果受到操控	Pereira F[107]
	保障安全策略的执行	Freire L[108]、sandboxing[109]
分布式计算与分布式系统	将计算任务卸载到数据平面	p4mr[111]、DAIET[23,24]、Linear Road[112]
	将一致性协议卸载到数据平面	Dang H T[114]、Li J[115]、Zhang Y[116]
	带状态更新与迁移	ez-Segway[117]、Swing State[118]
网络功能虚拟化	模块化转发框架	HyMoS[119]
	虚拟网络资源管理	NERV[120]
路由与流表资源优化	实现新型转发机制	PRPL[121]、Braun W[122]
	智能流表过期机制	He C H[123]
其他	P4 在其他领域的应用	BLESS[124]、Edwards T G[125]、Geyer F[127]、Wu Z[128]、Signorello S[129]

4 基于 P4 的应用

采用 P4 技术与可编程数据平面，网络管理员可以使用编程的方式对交换机的报文处理逻辑进行更改，从而很容易地实现新功能、支持新协议，缩短了开发周期与开发成本。并且，将一些原本由中间件实现的网络功能与端服务器实现的应用卸载到可编程数据平面上，还能获得可观的性能收益，提升网络与应用的整体表现。于是，随着 P4 与可编程数据平面的兴起，端系统、中间件、网络设备之间的工作界限开始变得模糊，学术界和工业界的研究人员也开始重新思考网络应该承担的角色与功能[23,24,111,112,115,126]。

如表 3 所示，本章从负载均衡与资源分配，网络测量、监控与诊断，网络安全，使用 P4 技术提升其他技术性能等方面展现了学术界与工业界基于 P4 与可编程数据平面作出的应用成果。

4.1 面向负载均衡与资源分配的应用

4.1.1 二层三层负载均衡

数据中心网络的节点有很大的度（degree），可以支持多路径路由，为负载均衡提供了拓扑上的条件。但是传统的负载均衡机制 ECMP 只是随机地分摊流量，在某些路径（例如：两条大流挤占同一条链路、非对称拓扑、链路故障）上依然会产生拥塞；而基于拥塞感知的负载均衡技术 CONGA[72]存在可扩展性差的缺点，只能在边缘节点维护小部分拥塞状态信息。针对上述缺点，文献[73]使用可编程数据平面，提出了数据平面逐跳负载均衡机制 HULA，设计了生成与转发探针的机制来感知网络拥塞状况。交换机只维护通向目的节点的最佳下一跳信息，而非像 CONGA 那样维护所有路径信息，因此提高了可扩展性；并且，最佳路径上按照 flowlet 粒度转发，避免了 TCP 包乱序。ns-2 上的仿真结果显示 HULA 比已有的负载均衡方案在流完成时间上快 1.6 至 3.3 倍。

4.1.2 传输层负载均衡

在大型数据中心中，44%的流量是虚拟 IP 地址流量[74], 需要做带状态负载均衡来维持每条连接的一致性，而数据中心通常会使用上百台服务器专门来做负载均衡，占用大约 4%的计算资源[75]。并且，使用服务器做负载均衡会引入较高计算开销，带来较高的延迟与抖动，影响用户体验。文献[18,19]提出了 SilkRoad，使用可编程交换机来做带状态四层负载均衡。通过存储流的五元组哈希值与 DIP 池版本号来分别减少匹配域和动作的大小，从而实现在交换机上同时维护上百万条流的带状态连接；通过硬件实现的 bloom filter 来确保即使 DIP 池发生更新也能维持每条连接的一致性。SilkRoad 可以使用一台可编程交换机替代上百台服务器来完成带状态负载均衡的工作，不仅将带状态负载均衡的成本降低了两个数量级，而且还降低了延迟与抖动。

除了像 SilkRoad 那样在交换机上维护每条流状态的负载均衡解决方案以外，也可以使用 P4 在网络设备上实现无状态的 TCP 负载均衡器 Beamer[76]。Beamer 的核心想法是利用后端服务器维持每条流的状态信息，并且让当前服务器将未知状态的数据包发送给其余的服务器进行处理，从而使得多路复用器无需保存流的状态，实现了网络设备无状态的负载均衡。

此外，由于现有的传输层协议并不能很好地兼顾低延迟与高吞吐传输[77]，借助 P4 技术可以在数据中心定制全新的传输层协议 NDP[78]来解决该问题。在 NDP 的设计中，发送方与接收方不进行连接握手，发送方一开始就以全速发送报文，交换机对报文进行逐包的多路径负载均衡；并且，交换机采用很小的缓冲区，当缓冲区队列快满时采用砍掉载荷、优先转发头部的方式告知接收方。虽然逐包的负载均衡会引入报文乱序，但是转发头部的方式不会丢失元数据，可以告知接收方实时流量的全局视图。在包括 P4 交换机在内的多种部署实验结果表明，NDP 可以同时实现小流近似最优的流完成时间与大流的高吞吐传输。

4.1.3 应用层负载均衡

当今互联网上的许多服务（例如：搜索、社交网络、电子商务等）依赖于高并发、低延迟的键值对查询。然而，由于 hot items 会受到远多于其他项目的请求[79]，因此会导致网络负载的严重不均衡，从而使得服务器过载或者空闲，造成吞吐的下降与响应时间的长尾分布。文献[80]提出了一种新的键值对存储架构 NetCache，利用可编程交换机给用户提供热键值对项目，从而均衡跨存储节点的负载。NetCache 的核心是数据平面设备使用匹配动作表对 key 进行分类、使用寄存器来存储 value、使用大流检测器来辨认 hot items，控制平面只负责 key 的插入和删除。实验表明单个可编程交换机每秒可以处理超过 20 亿次的请求，提升了 3 到 10 倍的吞吐；即使在高度不均衡和快速变化的工作负载下，NetCache 也能实现高聚合吞吐量和低延迟。

类似的，AppSwitch[81]也在可编程交换机上对键值缓存操作进行负载均衡，减少了端到端的平均延迟。更进一步，NetChain[82]设计了新的链复制协议与快速故障转移机制，在实现低延时、高吞吐的键值对查询的同时还提供了强一致性和容错性。

4.1.4 资源分配协议的近似实现

上述负载均衡相关的研究工作说明可编程交换机能够很好地均衡流量，然而由于受到计算能力以及硬件资源的限制，可编程交换机不具备足够的能力来支持网络协议的实现。于是，将一些协议做近似实现也是一种可行的解决方案。文献[83]采用基数估计方法近似计算流聚合值、采用加法近似计算乘除法，以及采用 count-min sketch 技术近似维持每个流状态，在数据平面上实现了近似的资源分配协议。考虑到可部署性，文献[84]在可编程交换机上采用近似公平队列按照轮询的方式进行调度，使用多个先进先出队列与近似排序队列来缓存数据包，达到接近于公平队列机制的性能。考虑到可扩展性，文献[85]设计了近似公平带宽分配新机制 FDPA，基于流的历史发送速率，动态地赋予其优先级，用以解决公平队列调度扩展性不足的问题。

4.2 面向网络测量、监控与诊断的应用

4.2.1 网络测量与监控的通用方法

对网络进行日常的管理和维护依赖于及时有效的网络测量和监控。NetFlow[86]是应用广泛的网络监控工具，但是在监控所有流时会带来较高的处理时间与较大的存储空间消耗，很难在数据中心的商用交换机中进行部署，因此 NetFlow 需要对数据包进行抽样，只能监控一部分流。由于对暂态路由循环、路由黑洞、突发流等进行检测需要在短时间内对所有流不采样地进行监控，于是文献[87]设计了 FlowRadar，其核心思想是在可编程交换机上使用扩展的可逆布鲁过滤器查询表对每条流的计数器进行编码，然后使用远程采集器的计算能力对全网流计数器进行解码和分析。FlowRadar 比 NetFlow 的可扩展性更好，可以监控所有的流。

UnivMon[88]提出了通用的流监控框架，提供了“One Big Switch”的抽象，在数据平面使用 sketch 进行测量，并把这个 sketch 上传到控制平面，控制平面根据自己的监控任务选择一个估计函数得到估计结果，从而可以兼顾通用性和准确性。

除了上述方案，还可以使用带内网络遥测（In-band network telementry，INT）[89]技术周期性地将测量结果上传至终端。INT 将数据包经过交换机队列的时间等元数据写在数据包的 INT 头部中，能够为终端设备提供细粒度的网络状况。采用 INT 技术，可以查询交换机内部的状态（例如：队列大小、链路利用率、排队延迟等），从而可以实现网络瞬态故障诊断、数据平面验证等高级应用。将 INT 技术和知识定义网络（knowledge-defined network，KDN）相结合，可以构建自驱动网络（self-driving network）[90]，减轻网络管理的成本。

4.2.2 大流检测相关研究

上述工作所提供的通用网络测量与监控方法在某些具体场景下并不适用，例如大流检测。因为上述工作需要通过控制器或者终端设备周期性地与交换机交互来获取网络事件，缺乏实时性，同时也增加了链路带宽占用，因此不宜采用控制器参与的方式检测大流。有工作[20，21]直接在数据平面检测大流，提出了 HashPipe 算法，为每个大流维护 counter，同时对小流进行“驱逐”，获得了可观的性能收益，使用少于 80KB 的交换机内存可以检测出 95%的大流。更进一步，文献[22]研究了在数据平面上对全网范围分布的大流做检测，每台交换机为不同的流设置不同的阈值，超过阈值的流被认为是 heavy hitter，将其计数发给协调器，协调器整合不同交换机上的数据来计算全网 heavy hitter，还可以根据每台边缘交换机流量分布自适应地调整每台交换机上的 heavy hitter 检测阈值。

4.2.3 网络故障诊断相关研究

在大型网络中故障经常会发生[91]，影响上层应用的性能，甚至造成重大经济损失[92]。因此需要及时地对 TCP 流大小是否异常、网络是否丢包、何处发生故障等问题作出正确诊断。

数据中心里 99%的流为 TCP 流[93]，若对其日志进行离线处理，不仅实时性差而且低效，因此需要在线诊断；如果修改终端虚拟机的协议栈来做检测又会侵犯租户的权益；如果在核心层做检测又不能获得终端视角（例如：往返时延）。于是，文献[94]提出了 Dapper 系统，使用可编程数据平面在“靠近”终端的位置（例如：hypervisor、网卡、架顶交换机等）对 TCP 连接的性能瓶颈做实时的诊断。Dapper 可以检测出来一条 TCP 连接是受发送方限制（例如：共享带宽竞争力不足），还是受网络限制（例如：网络拥塞），还是受接收方限制（例如：接收缓冲区很小），从而获知 TCP 连接的瓶颈位置。

为了减轻丢包对应用性能和吞吐的影响，文献[95]提出了快速检测丢包的通用系统 LossRadar，使

用可逆布鲁过滤器来计算数据包的摘要，分别在流进入和离开一个网络域时部署上游计量表和下游计量表，比较该对计量表产生的数据包摘要的差异来实时检测和定位丢包。

对于故障检测问题，KeySight[96]提出了一个可扩展的故障处理平台，使用布鲁过滤器以数据包行为（packets behavior）为粒度做 postcard 上传。与逐包和逐流上传 postcard 的方案相比，兼顾了检测的覆盖范围和可扩展性。

4.2.4 网络监控查询语言研究

上述的工作表明可编程交换机可以实现丰富的网络测量和故障诊断工作，但是针对不同的网络监控需求，究竟需要什么样的交换机硬件原语才能支持网络性能问题的表达？文献[97，98]提出了网络性能查询语言 Marple，网络管理员只需要使用 Marple 编写一个性能查询请求，通过 Marple 编译器编译并在可编程交换机上运行，查询结果将自动反馈给收集服务器。

Sonata[99]借助流式分析平台和可编程网络设备，提供了数据包级别的网络遥测系统，让管理员可以通过查询的方式收集并分析测量数据。

4.3 面向网络安全的应用

4.3.1 防御 DDoS 攻击

DDoS 攻击是目前网络环境中规模最大、频率最高的网络攻击手段[100,101,102]，传统方法使用中间件来缓解。虽然 SDN 技术可以被用来防御 DDoS 攻击，但是其本身又引入了新的 DDoS 的攻击点[103,104,105]（例如：SYN-flood 攻击会使控制信道的链路容量和数据平面缓存很快成为瓶颈）。文献[106]提出了可以完全实现在可编程数据平面上的 DDoS 流量清理方法，避免了与控制器的交互。数据平面设备会拦截并代理 SYN 请求，根据报文重新计算出的随机 SYN-ACK challenge 作为应答号，成功响应 challenge 的源地址会被加入白名单并重设连接。在通过 challenge 验证前，服务器和数据平面设备都不需要维护任何带状态的信息，可以应对很大的攻击流量。当攻击流量超过了单一设备资源限制时，使用同一条转发路径上所有可编程数据平面的资源对攻击流量进行协同过滤，每个设备只负责清洗一部分流量，剩下的交由下游设备处理。

4.3.2 防止网络监控结果受到操控

由于设备的资源限制，网络监控中经常会采用抽样或者使用 sketch 来对报文做统计。如[20，88]实现了基于 sketch 的监控算法，但这些算法的设计没有考虑安全问题：由于算法公开，攻击者可以通过观测、操纵经过监控设备的流量，预测甚至控制监控结果，从而使网络管理员根据监控结果采取不当的操作，使安全策略失效。文献[107]设计了基于加密哈希函数的 count-min sketch 算法，攻击者在不知道密钥的情况下无法预测监控算法的行为，从而解决了上述安全问题；并通过存储和查询旧密钥的统计信息，避免了运行时密钥更新带来的前后哈希函数映射计数器位置的不一致。

4.3.3 保障安全策略的执行

虽然 P4 增加了数据平面可编程性，但同时也增加了数据平面产生软件漏洞的可能性，这些漏洞很可能会被利用来绕过一些安全策略（例如：网络隔离、报文头部完整性），造成安全隐患。文献[108]提出了一种基于断言检查和符号执行的静态程序分析机制来验证 P4 程序的行为，保证安全策略的执行。首先，需要人工地使用断言语言，在程序中相应的位置标注应当满足的安全策略或期望的程序行为；然后标注好的程序将和转发规则一起被翻译为等价的 C 语言程序；最后符号执行 C 程序并枚举程序的各个分支，遍历程序所有可能的执行路线，在执行到断言处检查验证程序的行为是否和标注一致，报告不一致的情况。该方案的 P4 程序验证的时间复杂度是以流表内动作数目为底数、流表个数为指数的形式，可扩展性较差。

此外，由于 P4 程序直接和底层平台交互，也可能会造成安全策略的违背。文献[109]提出了软件沙箱（Sand box）的机制，通过监控数据平面程序与底层设备的交互来拦截系统调用（如动作原语等），将数据平面程序行为限制在安全策略范围内。

4.4 使用 P4 技术提升其他技术的性能

4.4.1 分布式计算与分布式系统

（1）将计算任务卸载到数据平面

由于可编程数据平面可以对报文做灵活的操作以及具有高达 6.5Tbps 的吞吐[110]，为网内计算提供了新的机遇。在数据中心里，服务器之间依靠交换机进行通信，如果把一部分原本在服务器端完成的计算任务（Map-Reduce, Machine Learning, Graph Processing, Stream Processing 等）卸载到可编程数据平面上执行，那么当数据包穿过网络到达终端服务器时，计算任务也就相应完成了，不仅可以减少网络流量、缓解拥塞，还可以缓解应用层负担、腾出服务器 CPU 周期。但是，由于可编程网络设备的内存资源有限、动作集有限、对每个报文的操

作次数有限，因此必须审慎地选择放在网络中完成的计算工作，才能做到既符合上述限制，又保证计算的全局正确性，提高数据中心的整体性能。

文献[111]提出了并行编程框架 p4mr，对数据平面计算开销做了初步的理论建模，并对 Map-Reduce 等计算任务卸载到数据平面的收益进行了实验评估。文献[23，24]提出了 DAIET 系统，使用基于 P4 的可编程交换机执行网内数据聚合的工作，实验表明达到了近 90%的数据聚合率。

流处理（Stream Processing）不需要进行循环操作，并且所需记录的状态有限，因此适合在可编程数据平面上完成。文献[112]使用 P4 语言在可编程数据平面上实现了流处理应用 Linear Road[113]，比现有基于软件的 Linear Road 吞吐更高、速度更快。

（2）将一致性协议卸载到数据平面

一致性是分布式计算系统的重要目标，其中，Paxos 协议是许多分布式系统和服务的基础组成部分，用来实现应用层的一致性复制备份，以确保当服务器故障时数据的可用性。Paxos 协议包括四种角色：proposer，coordinator，acceptor 和 learner。然而，Paxos 需要对每个请求进行复杂的协调，引入了较高的延迟，限制了系统的可扩展性。有研究[114]将 coordinator 和 acceptor 的工作直接实现在数据平面上，从而对 Paxos 进行加速。此外，文献[115]给出了一种在数据中心实现较小复制开销的新方法，将网络与应用层协议的角色进行了重新划分：网络设备使用有序不可靠组播（Ordered Unreliable Multicast, OUM）的新原语将请求排序，保证请求是有序的，但是不保证请求的可靠传输，可能会有丢包；应用层使用新的复制协议 Network Ordered Paxos（NOPaxos）来确保已经有序报文的可靠性传输，从而实现了高性能的复制备份，比 Paxos 在延迟和吞吐量上都有明显优化。

除了 Paxos 协议，Raft 一致性算法也能够在可编程数据平面上实现，从而降低一致性延迟、感知网络故障、提升 Raft 一致性算法在分布式系统中的性能。在 Raft 协议中，有 leader，follower 和 candidate 三种角色，其中 leader 通过发起 AppendEntries 的远程程序调用 （Remote Procedure Call, RPC）来发送心跳信息或日志信息。文献[116]将 AppendEntries RPC 的信息处理由可编程交换机实现。前端的可编程交换机负责日志的复制和提交、运行 Raft 感知的数据包转发，并且通过改写到达的数据包实现对 Raft 请求的快速响应；后端的服务器运行完整的 Raft 协议。实验结果表明心跳信息延迟降低了十分之一，写请求延迟降低了接近五分之三。

（3）带状态更新与迁移

对于分布式系统而言，网络更新与带状态迁移的正确性与高效性同样很重要。文献[117]提出了一种分布式网络更新机制 ez-Segway，能够快速地进行网络状态的一致性更新，避免更新时的转发异常和链路拥塞。文献[118]提出了一个通用的状态管理框架 Swing State，核心思想是在数据包中捎带状态更新信息来实现状态迁移，能够支持带状态数据平面的状态一致性迁移。在 P4 程序编译时，Swing State 分析器自动识别需要迁移的流与共享的状态，修改 P4 程序使其支持运行时的实时迁移。当控制器下发流迁移的指令时，Swing State 的状态管理器检查程序分析的结果，在确认状态迁移的安全性后，通知源端和接收端进行状态迁移。

4.4.2 网络功能虚拟化

P4 与可编程数据平面也为加速网络功能虚拟化（Network Function Virtualization, NFV）提供了新的机遇。目前主要有两个研究方向，一个是利用 P4 的灵活性和可编程性实现高性能网络功能，另一个是利用 P4 实现虚拟环境下的网络资源管理。

（1）P4 驱动的模块化转发框架

包括防火墙、负载均衡器在内的许多网络功能已经实现在了 NFV 平台上，但是目前网络管理员不得不通过牺牲可编程性来保证 NFV 系统的性能。为了解决上述问题，HyMoS[119]利用兼容 P4 的网卡提升 NFV 平台的性能，同时能够提供模块化的软件和硬件设计，并保证了较高的可编程性。

（2）P4 驱动的虚拟网络资源管理

NERV[120]是一种能够支持多种网络架构的网络资源管理器。基于微服务架构的设计理念和 P4 可编程协议无关的转发元件，NERV 提出了一种包含多个微小应用的框架，每个应用能够管理网络中的一组资源，同时兼容多种不同的协议栈和转发策略。NERV 利用标准 API 将这些应用重组，允许管理员调用通用接口来管理网络资源。

4.4.3 路由与流表资源优化

流表资源是影响网络可扩展性的重要因素。一方面 P4 能够实现更加灵活的流表匹配方式，另一方面，P4 能够提供智能的流表过期机制实现流表资源的充分利用。因此 P4 为流表资源优化以及实现更可扩展的数据平面带来了机遇。

（1）基于 P4 的实现新型转发机制

在大规模的企业网或者数据中心网络中，实现网络级策略仍然具有相当大的挑战。目前数据平面设备主要依赖二层到四层的标识符来指定转发策略，而并不关注上层应用信息。PRPL[121]将用户ID、进程 ID 等进程级信息通过哈希操作嵌入到数据包中，可编程交换机则通过进程级信息实现更加智能的数据包处理。文献[122]实现了软件定义多播，能够提供可扩展的数据包多播服务，并支持包括 BIER、BIER-TE 在内的多种功能。

（2）基于 P4 的智能流表过期机制

现有的 OpenFlow 流表过期机制依赖于一个固定的超时时间，交换机主动地移除超时的流表项，这个超时时间值的设定是流表空间利用率与交换机——控制器通信开销之间的折衷。文献[123]使用基于 P4 的交换机，能够在 TCP 流结束时及时从流表中删除相应的表项，减少流表空间的占用，并且不会带来额外的与控制器通信的开销。通过检测 TCP 流中 FIN 和 RST 报文来确定流是否结束，从而移除多余的流表项；对于 UDP 等没有结束标志的流，仍通过超时过期机制移除流表项。

4.5 其他应用

4.5.1 物联网

物联网（Internet-of-Things, IoT）是互联网的应用扩展。为了实现自动化管理大规模物联网，文献[124] 提出了 BLESS，可以灵活地实现包括 IoT 服务切片以及组织在内的多种功能。其中可编程交换机的主要功能是实现高性能的 IoT 设备连接模块以及数据转发模块。

4.5.2 多媒体网络

目前实时视频流量传输已经成为十分重要的网络应用，主要目标是为视频供应商提供足够的敏捷性和灵活性，带来的好处主要包括与商用交换机和服务器网络接口兼容、灵活地重组和协调多媒体流等。尽管 SDN 为实现可编程的多播控制提供了控制接口，但是底层的交换设备所支持的多播功能是十分有限的。因此文献[125]提出利用可编程交换机的协议无关特性和可编程性加速视频流转发，实现高性能的视频转发技术。

4.5.3 航空电子领域

由于航空领域对于专用网络协议的安全性有特殊要求，通常不会使用普通的商用交换机。而 P4 的出现允许开发者自己定义数据包处理逻辑，增加了通用交换机应用的可能性。此外，航空领域对网络时延要求十分严格，而 P4 使得数据包转发模型更加独立，为形式化地分析系统时延提供了可能。为说明 P4 在航空领域的适用性，文献[127]使用 P4 基本实现了 AFDX 协议，同时，该文献也指出现在的 P4 在安全性、功能性上仍不完善，在航空电子领域的 P4 商业应用仍需时日。

4.5.4 云服务市场

为了提供多样化的网络服务，亚马逊、微软等云服务商往往会开放云服务市场来提供第三方开发者开发的服务。不过，现有的云服务商店在设计上仍存在缺陷：第三方提供的服务必须和供应商已有服务独立，不能对现有服务功能进行改善，使得许多功能不易或无法由第三方实现。文献[128]使用 P4 技术设计了新的第三方服务体系结构，通过交换机识别数据包所需要的服务，将这些数据包复制或直接转发到对应的服务模块进行处理；同时为了高效率地对数据包所需服务进行区分，该工作在数据包头部增加了一个域来提供必要信息帮助判别；通过插入多个数据包头部，可以实现多种服务的协作。相比传统网络，P4 带来的可编程性使得在交换机上完成流量的区分和不同服务的编排成为可能。

4.5.5 新型网络体系结构

包括命名数据网络（Named-Data Networking, NDN）在内的新型网络体系结构在短期内较难部署，借助 P4 技术可以在真实网络设备上实现其处理逻辑。文献[129]使用可编程交换机实现了 NDN 数据包的处理逻辑，包括对 NDN 两种类型数据包格式的解析、对存储兴趣包处理信息的 PIT 表定义、对存储数据包转发位置的 FIB 表定义，以及相应下发规则所需的控制逻辑。由于 NDN 协议中最长前缀匹配等逻辑并不被 P4 直接支持，所以使用 P4 的基本功能实现 NDN 的功能会造成性能上的损失。

4.6 P4 应用小结

P4 这项技术在提供数据平面可编程能力的同时兼顾了报文处理速度，因此数据平面可以在保证高速处理报文的前提下完成自定义的报文处理逻辑，于是数据平面可以帮助终端设备完成部分工作，从而获得大幅性能提升。此外，可重配置性、协议无关性、平台无关性的特点能够大幅降低支持新应用的开发时间与成本，加快网络创新的步伐。

但是，由于 P4 在设计上并不是图灵完全的，并且数据平面资源也十分珍贵，因此将各种应用卸载到可编程数据平面之前需要仔细思考两个问题：（1）什么样的应用急需被卸载到数据平面执行？（2）什么样的应用适合在数据平面执行？在对已

有 P4 应用总结的基础之上，本文认为：（1）需要局部测量+决策的应用、需要频繁通信的应用、I/O 为瓶颈的应用、延迟敏感的应用等，急需被卸载到数据平面上执行，这样可以充分利用数据平面快速响应局部事件、低时延报文处理、高吞吐报文转发等优势，最大化提升网络与上层应用的整体性能；（2）可以被“匹配+动作”模式描述的应用、可以被分布式处理的应用等，适合在数据平面上执行，因为只有这样才能符合数据平面的抽象转发模型以及数据平面分布式特性，是将应用卸载到数据平面执行的前提。

总之，寻求既急需又适合在数据平面实现的应用，并进行科学合理的系统设计，是迈向高度定制化、功能高度分化的高可编程网络的第一步。

5 未来工作

P4 与可编程数据平面目前已经得到了广泛的关注，不仅学术界对 P4 关键技术及其相关应用场景进行了研究，而且工业界已经有成熟的商业产品问世[34,35,110]。P4 技术的出现给网络研究带来了诸多机遇，但同时也面临着更多的挑战。

（1）对 P4 语言进行扩展

P4 语言从最初的 $P4_{14}$ 发展为如今的 $P4_{16}$，虽然增添了一些新特性，但是如 2.5 节所述，P4 语言存在着语法上的局限性、功能上的局限性、标准规范上的局限性，制约着 P4 语言能够表达的数据包处理逻辑。对 P4 程序的扩展主要有两个方向，首先是平台无关的扩展，即仅对 P4 语言本身进行扩展，并且保证扩展是不依赖于平台的；其次是平台相关的扩展。平台相关的扩展包含两种方式：第一种是扩展新原动作，原动作是 P4 语言的最小处理单元，目前 P4 语言所支持的原动作集合是固定的，因此所能支持的数据包处理功能也相对比较局限，不同平台可以扩展不同的原动作集合，这种扩展方式的好处是程序员比较方便地调用原动作；第二种是增加新的 P4 语言要素，比如设备内增加新的带状态操作 ALU，此时仅增加原动作并不能够完整地定义带状态操作，因此需要定义新的 P4 语言要素。

（2）高效利用有限的数据平面资源

相比终端服务器，可编程数据平面本身的计算资源与存储资源有限，使用 P4 语言实现自定义数据包处理逻辑的时候需要仔细斟酌数据平面能够完成的计算与可用的资源，通常需要使用哈希操作以及布鲁过滤器等数据结构来做折衷。因此，对可编程数据平面计算资源和存储资源进行有效抽象，能够帮助程序员合理地设计和实现数据平面应用。其次，创建可编程数据平面资源模型能够指导 P4 程序编译器设计，使其更好地对数据平面的使用进行优化。

（3）P4 技术新应用场景研究

P4 技术具有可重配置、协议无关、平台无关等特性，利用这些特性可以解决现有网络中的诸多问题，同时还能将一些原本由中间件和终端设备完成的工作卸载到可编程数据平面上，从而获得可观的性能收益。例如，利用可编程数据平面可以实现网络功能的卸载，增加整体的数据包处理能力；其次，利用可编程数据平面实现可扩展、细粒度的网络监控技术，利用 P4 的可编程能力对测量数据实现初步地分析和处理，实现高效的网络故障检测和定位。可以预见，P4 技术在诸多本文未提及的应用场景下仍可有所作为。

（4）P4 安全问题研究

P4 技术在提升数据平面可编程性的同时也增加了数据平面软件漏洞，从而破坏已有的安全策略，引入新的安全问题。可编程数据平面可能会在三方面存在漏洞，分别是 P4 表项、P4 程序以及 P4 交换机。目前研究人员已经开始关注 P4 表项和 P4 程序的验证，但是缺乏对于 P4 交换机的验证。同时，目前 P4 验证的相关工作存在执行效率低、可扩展性差等缺陷，也是未来研究需要解决的问题。因此，对这三方面进行验证分析，完善数据平面的安全机制和框架，确保安全策略的执行，将成为 P4 与可编程数据平面安全的重要保障。

6 结束语

P4 与可编程数据平面目前是受到学术界与工业界广泛关注的热门技术之一。P4 作为新兴技术，之所以能够得到长足发展，在于它具有独特的优势：首先，P4 吸收了 SDN 数据平面与控制平面分离的思想，促进了网络设备的开放性，加快了新协议与新应用的部署；然后，P4 着眼于数据平面的可编程性，具备 OpenFlow 等技术并不具有的可重配置、协议无关、平台无关的特性，进一步抽象了网络设备转发模型，提升了数据平面灵活性，降低了网络运维成本；最后，由于 P4 融合了数据平面高性能报文处理能力与可编程能力，因此许多在中间件与终端服务器上完成的工作可以在数据平面上完成，从而获得性能上的大幅提升。

致 谢 感谢清华大学信息网络科学与网络空间研究院网络体系结构与 IPv6 研究室中孙晨、郑智隆、张岱、张梦豪、白家松、操佳敏、况鹏对本论文的支持，感谢清华大学中梁健哲、夏照越、段光林对本论文的支持。

参 考 文 献

[1] Nunes B A A, Mendonca M, Nguyen X N, et al. A survey of software-defined networking: Past, present, and future of programmable networks. IEEE Communications Surveys & Tutorials, 2014, 16(3): 1617-1634.

[2] Tennenhouse D L, Smith J M, Sincoskie W D, et al. A survey of active network research. IEEE communications Magazine, 1997, 35(1): 80-86.

[3] Elliott C. GENI: opening up new classes of experiments in global networking. IEEE internet computing, 2010, 14(1): 39-42.

[4] Gavras A, Karila A, Fdida S, et al. Future internet research and experimentation: the FIRE initiative. ACM SIGCOMM Computer Communication Review, 2007, 37(3): 89-92.

[5] JGN2plus. 2012. http://www.jgn.nict.go.jp/english/index.html

[6] SOFIA. 2012. http://fi.ict.ac.cn/research/sofia_overview.htm

[7] Doria A, Salim JH, Haas R, et al. Forwarding and control element separation (ForCES) protocol specification. IETF RFC 5810, 2010.

[8] Greenberg A, Hjalmtysson G, Maltz D A, et al. A clean slate 4D approach to network control and management. ACM SIGCOMM Computer Communication Review, 2005, 35(5): 41-54.

[9] Caesar M, Caldwell D, Feamster N, et al. Design and implementation of a routing control platform//Proceedings of the 2Nd Conference on Symposium on Networked Systems Design & Implementation. Boston, USA, 2005: 15-28.

[10] Casado M, Garfinkel T, Akella A, et al. SANE: A Protection Architecture for Enterprise Networks// Proceedings of USENIX Security Symposium. Boston, USA, 2006, 49: 137-151.

[11] Casado M, Freedman M J, Pettit J, et al. Ethane: Taking control of the enterprise. ACM SIGCOMM Computer Communication Review. ACM, 2007, 37(4): 1-12.

[12] McKeown N, Anderson T, Balakrishnan H, et al. OpenFlow: enabling innovation in campus networks. ACM SIGCOMM Computer Communication Review, 2008, 38(2): 69-74.

[13] Bosshart P, Daly D, Gibb G, et al. P4: Programming protocol-independent packet processors. ACM SIGCOMM Computer Communication Review, 2014, 44(3): 87-95.

[14] Bosshart P, Gibb G, Kim H S, et al. Forwarding metamorphosis: Fast programmable match-action processing in hardware for SDN// Proceedings of the Special Interest Group on Data Communication. Hong Kong, China, 2013, 43(4): 99-110.

[15] Chole S, Fingerhut A, Ma S, et al. dRMT: Disaggregated Programmable Switching//Proceedings of the Conference of the ACM Special Interest Group on Data Communication. Saint Louis, USA, 2017: 1-14.

[16] IETF. VXLAN: A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks, May 2013. https://tools.ietf.org/html/draft-mahalingam-dutt-dcops-vxlan-04.

[17] Dukkipati N . Rate Control Protocol (RCP): Congestion control to make flows complete quickly. These Instructions, 2007(8):256-257.

[18] Miao R, Zeng H, Kim C, et al. SilkRoad: Making Stateful Layer-4 Load Balancing Fast and Cheap Using Switching ASICs//Proceedings of the Conference of the ACM Special Interest Group on Data Communication. Saint Louis, USA, 2017: 15-28.

[19] Lee J, Miao R, Kim C, et al. Stateful Layer-4 Load Balancing in Switching ASICs//Proceedings of the SIGCOMM Posters and Demos. Saint Louis, USA, 2017: 133-135.

[20] Sivaraman V, Narayana S, Rottenstreich O, et al. Heavy-hitter detection entirely in the data plane//Proceedings of the Symposium on SDN Research. Santa Clara, USA, 2017: 164-176.

[21] Popescu D A, Antichi G, Moore A W. Enabling Fast Hierarchical Heavy Hitter Detection using Programmable Data Planes//Proceedings of the Symposium on SDN Research. Santa Clara, USA, 2017: 191-192.

[22] Harrison R, Cai Q, Gupta A, et al. Network-Wide Heavy Hitter Detection with Commodity Switches//Proceedings of the Symposium on SDN Research. Los Angeles, USA, 2018: 8-14.

[23] Sapio A, Abdelaziz I, Aldilaijan A, et al. In-Network Computation is a Dumb Idea Whose Time Has Come//Proceedings of the 16th ACM Workshop on Hot Topics in Networks. California, USA, 2017: 150-156.

[24] Sapio A, Abdelaziz I, Canini M, et al. DAIET: a system for data aggregation inside the network//Proceedings of the 2017 Symposium on Cloud Computing. California, USA, 2017: 626-626.

[25] P4 Language Consortium. Website. https://p4.org

[26] P4-hlir. https://github.com/p4lang/p4-hlir.

[27] P4 Language Consortium. The p4_14 language specification. Website. https://p4.org/p4-spec/p4-14/v1.0.4/tex/p4.pdf

[28] P4 Language Consortium. The p4_16 language specification. Website. https://p4.org/p4-spec/docs/P4-16-v1.0.0-spec.pdf

[29] Song H. Protocol-oblivious forwarding: Unleash the power of SDN through a future-proof forwarding plane//Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking. Hong Kong, China, 2013: 127-132.

[30] Kohler E, Morris R, Chen B, et al. The Click modular router. ACM Transactions on Computer Systems, 2000, 18(3): 263-297.

[31] Duncan R, Jungck P. packetC language for high performance packet processing//Proceedings of High Performance Computing and Communications. Seoul, Korea, 2009: 450-457.

[32] Brebner G, Jiang W. High-Speed Packet Processing using Reconfigurable Computing. IEEE Micro, 2014 (1): 8-18.

[33] Jose L, Yan L, Varghese G, et al. Compiling Packet Programs to

Reconfigurable Switches//Proceedings of Networked Systems Design and Implementation. Santa Clara, USA, 2015: 103-115.

[34] Intel FlexPipe. http://www.intel.com/content/dam/www/public/us/en/documents/product-briefs/ ethernet-switch-fm6000-series-brief.pdf

[35] Xpliant packet architecture (xpa) press release. https://cavium.com/xpliant-ethernet-switch-product-family.html

[36] Wang H, Soulé R, Dang H T, et al. P4FPGA: a rapid prototyping framework for p4//Proceedings of the Symposium on SDN Research. Santa Clara, USA, 2017: 122-135.

[37] Wang H, Lee K S, Shrivastav V, et al. P4FPGA: High Level Synthesis for Networking//Proceedings of the SIGCOMM Posters and Demos. Salvador, Brazil, 2016: 620

[38] Benek P, Pu V, Kubtov H. P4-to-VHDL: Automatic generation of 100 gbps packet parsers//Proceedings of Field-Programmable Custom Computing Machines. Washington DC, USA, 2016: 148-155.

[39] Benáček P, Puš V, Kubátová H, et al. P4-To-VHDL: Automatic generation of high-speed input and output network blocks. Microprocessors and Microsystems, 2018, 56: 22-33.

[40] Santiago da Silva J, Boyer F R, Langlois J M. P4-Compatible High-Level Synthesis of Low Latency 100 Gb/s Streaming Packet Parsers in FPGAs//Proceedings of the 2018 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. Monterey, USA, 2018: 147-152.

[41] Kekely M, Korenek J. Mapping of P4 match action tables to FPGA//Proceedings of Field Programmable Logic and Applications. Ghent, Belgium, 2017: 1-2.

[42] Cabal J, Benáček P, Kekely L, et al. Configurable FPGA Packet Parser for Terabit Networks with Guaranteed Wire-Speed Throughput//Proceedings of the 2018 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. Monterey, USA, 2018: 249-258.

[43] Li P, Luo Y. P4GPU: Acceleration of programmable data plane using a cpu-gpu heterogeneous architecture//Proceedings of High Performance Switching and Routing. Yokohama, Japan, 2016: 168-175.

[44] Li P, Luo Y. P4GPU: Accelerate packet processing of a p4 program with a cpu-gpu heterogeneous architecture//Proceedings of the 2016 Symposium on Architectures for Networking and Communications Systems. Ithaca, USA, 2016: 125-126.

[45] Arashloo M T, Koral Y, Greenberg M, et al. SNAP: Stateful network-wide abstractions for packet processing//Proceedings of the Conference of the ACM Special Interest Group on Data Communication. Salvador, Brazil, 2016: 29-43.

[46] Shahbaz M, Choi S, Pfaff B, et al. Pisces: A programmable, protocol-independent software switch//Proceedings of the Conference of the ACM Special Interest Group on Data Communication. Salvador, Brazil, 2016: 525-538.

[47] Pfaff B, Pettit J, Koponen T, et al. The design and implementation of open vswitch//Proceedings of Symposium on Networked Systems Design and Implementation. Santa Clara, USA, 2015: 117-130.

[48] Intel data plane development kit. http://dpdk.org/.

[49] Laki S, Horpácsi D, Vörös P, et al. High speed packet forwarding compiled from protocol independent data plane specifications// Proceedings of the Conference of the ACM Special Interest Group on Data Communication. Salvador, Brazil, 2016: 629-630.

[50] p4c-bmv2. https://github.com/p4lang/p4c

[51] Bhardwaj A, Shree A, Reddy V B, et al. A Preliminary Performance Model for Optimizing Software Packet Processing Pipelines//Proceedings of the 8th Asia-Pacific Workshop on Systems. Hong Kong, China, 2017: 26-32.

[52] Choi S, Long X, Shahbaz M, et al. The Case for a Flexible Low-Level Backend for Software Data Planes//Proceedings of the First Asia-Pacific Workshop on Networking. Hong Kong, China, 2017: 71-77.

[53] Choi S, Long X, Shahbaz M, et al. PVPP: A Programmable Vector Packet Processor//Proceedings of the Symposium on SDN Research. Santa Clara, USA, 2017: 197-198.

[54] Patra P G, Rothenberg C E, Pongracz G. MACSAD: High performance dataplane applications on the move//Proceedings of High Performance Switching and Routing. Campinas, Brazil, 2017: 1-6.

[55] Patra P G, Rothenberg C E, Pongracz G. MACSAD: Multi-Architecture Compiler System for Abstract Dataplanes (Aka Partnering P4 with ODP)//Proceedings of the SIGCOMM Posters and Demos. Salvador, Brazil, 2016: 623-624.

[56] Hancock D, Van Der Merwe J. Hyper4: Using p4 to virtualize the programmable data plane//Proceedings of the 12th International on Conference on emerging Networking EXperiments and Technologies. Irvine, USA, 2016: 35-49.

[57] Zhang C, Bi J, Zhou Y, et al. Hyperv: A high performance hypervisor for virtualization of the programmable data plane//Proceedings of Computer Communication and Networks. Vancouver, Canada, 2017: 1-9.

[58] Zhang C, Bi J, Zhou Y, et al. Mpvisor: A modular programmable data plane hypervisor//Proceedings of the Symposium on SDN Research. Santa Clara, USA, 2017: 179-180.

[59] Shahbaz M, Feamster N. The case for an intermediate representation for programmable data planes//Proceedings of the 1st ACM SIGCOMM Symposium on Software Defined Networking Research. Santa Clara，USA, 2015: 3-8.

[60] Abhashkumar A, Lee J, Tourrilhes J, et al. P5: Policy-driven optimization of P4 pipeline//Proceedings of the Symposium on SDN Research. Santa Clara, USA, 2017: 136-142.

[61] Zhou Y, Bi J. ClickP4: Towards modular programming of p4//Proceedings of the SIGCOMM Posters and Demos. Saint Louis, USA, 2017: 100-102.

[62] Ma Z, Bi J, Zhang C, et al. CacheP4: A Behavior-level Caching Mechanism for P4//Proceedings of the SIGCOMM Posters and Demos. Saint Louis, USA, 2017: 108-110.

[63] Nötzli A, Khan J, Fingerhut A, et al. P4pktgen: Automated test case generation for p4 programs//Proceedings of the Symposium on SDN

Research. Los Angeles, USA, 2018: 5-11.

[64] bmv2. https://github.com/p4lang/behavioral-model

[65] Zhang C, Bi J, Zhou Y, et al. P4DB: On-the-fly debugging of the programmable data plane//Proceedings of International Conference on Network Protocols. Toronto, Canada, 2017: 1-10.

[66] Freire L, Neves M, Leal L, et al. Uncovering Bugs in P4 Programs with Assertion-based Verification//Proceedings of the Symposium on SDN Research. Los Angeles, USA, 2018: 4-10.

[67] Lopes N, Bjørner N, McKeown N, et al. Automatically verifying reachability and well-formedness in P4 Networks. Washington: Microsoft, Technical Report: 01, 2016.

[68] Dang H T, Wang H, Jepsen T, et al. Whippersnapper: A P4 language benchmark suite//Proceedings of the Symposium on SDN Research. Santa Clara, USA, 2017: 95-101.

[69] Abdi S, Aftab U, Bailey G, et al. PFPSim: a programmable forwarding plane simulator//Proceedings of the 2016 Symposium on Architectures for Networking and Communications Systems. Santa Clara, USA, 2016: 55-60.

[70] Bai J, Bi J, Kuang P, et al. NS4: Enabling Programmable Data Plane Simulation//Proceedings of the Symposium on SDN Research. Los Angeles, USA, 2018: 12-17.

[71] Fan C, Bi J, Zhou Y, et al. NS4: A P4-driven Network Simulator//Proceedings of the SIGCOMM Posters and Demos. Saint Louis, USA, 2017: 105-107.

[72] Alizadeh M, Edsall T, Dharmapurikar S, et al. CONGA: distributed congestion-aware load balancing for datacenters//Proceedings of the Conference of the ACM Special Interest Group on Data Communication. Chicago, USA, 2014, 44(4): 503-514.

[73] Katta N, Hira M, Kim C, et al. Hula: Scalable load balancing using programmable data planes//Proceedings of the Symposium on SDN Research. Santa Clara, USA, 2016: 10-21.

[74] Patel P, Bansal D, Yuan L, et al. Ananta: Cloud scale load balancing. ACM SIGCOMM Computer Communication Review, 2013, 43(4): 207-218.

[75] Eisenbud D E, Yi C, Contavalli C, et al. Maglev: A Fast and Reliable Software Network Load Balancer//Proceedings of Networked Systems Design and Implementation. Santa Clara, USA, 2016: 523-535.

[76] Olteanu V, Agache A, Voinescu A, et al. Stateless datacenter load-balancing with beamer//Proceedings of Networked Systems Design and Implementation. Renton, USA, 2018, 18: 125-139.

[77] Guo C, Wu H, Deng Z, et al. RDMA over commodity ethernet at scale// Proceedings of the Conference of the ACM Special Interest Group on Data Communication. Florianópolis, Brazil, 2016: 202-215.

[78] Handley M, Raiciu C, Agache A, et al. Re-architecting datacenter networks and stacks for low latency and high performance//Proceedings of the Conference of the ACM Special Interest Group on Data Communication. Saint Louis, USA, 2017: 29-42.

[79] Atikoglu B, Xu Y, Frachtenberg E, et al. Workload analysis of a large-scale key-value store//Proceedings of the ACM Special Interest Group on Performance Evaluation. London, UK, 2012, 40(1): 53-64.

[80] Jin X, Li X, Zhang H, et al. NetCache: Balancing Key-Value Stores with Fast In-Network Caching//Proceedings of the 26th Symposium on Operating Systems Principles. Shanghai, China, 2017: 121-136.

[81] Cidon E, Choi S, Katti S, et al. AppSwitch: Application-layer Load Balancing within a Software Switch//Proceedings of the First Asia-Pacific Workshop on Networking. Hong Kong, China, 2017: 64-70.

[82] Jin X, Li X, Zhang H, et al. NetChain: Scale-Free Sub-RTT Coordination//Proceedings of Networked Systems Design and Implementation. Renton, USA, 2018: 35-49.

[83] Sharma N K, Kaufmann A, Anderson T E, et al. Evaluating the Power of Flexible Packet Processing for Network Resource Allocation// Proceedings of Networked Systems Design and Implementation. Boston, USA, 2017: 67-82.

[84] Sharma N K, Liu M, Atreya K, et al. Approximating Fair Queueing on Reconfigurable Switches//Proceedings of Networked Systems Design and Implementation. Renton, USA, 2018: 1-16

[85] Cascone C, Bonelli N, Bianchi L, et al. Towards approximate fair bandwidth sharing via dynamic priority queuing//Proceedings of Local and Metropolitan Area Networks. Osaka, Japan, 2017: 1-6.

[86] NetFlow. https://www.ietf.org/rfc/ rfc3954.txt.

[87] Li Y, Miao R, Kim C, et al. FlowRadar: A Better NetFlow for Data Centers//Proceedings of Networked Systems Design and Implementation. Santa Clara, USA, 2016: 311-324.

[88] Liu Z, Manousis A, Vorsanger G, et al. One sketch to rule them all: Rethinking network flow monitoring with univmon//Proceedings of the Conference of the ACM Special Interest Group on Data Communication. Salvador, Brazil, 2016: 101-114.

[89] Kim C, Sivaraman A, Katta N, et al. In-band network telemetry via programmable dataplanes//Proceedings of the Symposium on SDN Research. Santa Clara, USA, 2015: 1-2

[90] Hyun J, Hong J W K. Knowledge-defined networking using in-band network telemetry//Network Operations and Management Symposium. Seoul, Korea, 2017: 54-57.

[91] Z. Li, M. Liang, L. O'Brien, and H. Zhang. The cloud's cloudy moment: A systematic survey of public cloud service outage. International Journal of Cloud Computing and Services Science, 2013, 2(5):321–331.

[92] Bigelow, S. The causes and costs of data center system downtime. Advisory board Q&A. http://searchdatacenter.techtarget.com/feature/ The-causes-and-costs-of-data-center-system-downtime-Advisory-Board-QA. June 2011

[93] Alizadeh M, Greenberg A, Maltz D A, et al. Data center tcp (dctcp). ACM SIGCOMM computer communication review, 2011, 41(4): 63-74.

[94] Ghasemi M, Benson T, Rexford J. Dapper: Data plane performance diagnosis of tcp//Proceedings of the Symposium on SDN Research. Santa Clara, USA, 2017: 61-74.

[95] Li Y, Miao R, Kim C, et al. Lossradar: Fast detection of lost packets in

data center networks//Proceedings of the 12th International on Conference on emerging Networking EXperiments and Technologies. Irvine, USA, 2016: 481-495.

[96] Xia Z, Bi J, Zhou Y, et al. KeySight: A Scalable Troubleshooting Platform Based on Network Telemetry//Proceedings of the Symposium on SDN Research. Los Angeles, USA, 2018: 20-21.

[97] Narayana S, Sivaraman A, Nathan V, et al. Language-directed hardware design for network performance monitoring//Proceedings of the Conference of the ACM Special Interest Group on Data Communication. Saint Louis, USA, 2017: 85-98.

[98] Nathan V, Narayana S, Sivaraman A, et al. Demonstration of the Marple System for Network Performance Monitoring//Proceedings of the SIGCOMM Posters and Demos. Saint Louis, USA, 2017: 57-59.

[99] Gupta A, Harrison R, Canini M, et al. Sonata: query-driven streaming network telemetry//Proceedings of the Conference of the ACM Special Interest Group on Data Communication. Budapest, Hungary, 2018: 357-371.

[100] Akamai's security Q3 2015 report. https://www.stateoftheinternet.com/downloads/pdfs/2015-cloud-security-report-q3.pdf/.

[101] DDoS Threat Landscape Report 2013-2014 Security Alliance. https://www.incapsula.com/blog/wp-content/uploads/2015/08/2013-14 ddos threat landscape.pdf.

[102] Radware 2014-2015 Global Application and Network Security Report. https://www.radware.com/ert-report-2014/.

[103] Stateful Connection Tracking in Open vSwitch. http://openvswitch.org/support/ovscon2014/17/1030-conntrack nat.pdf/.

[104] Bianchi G, Bonola M, Capone A, et al. OpenState: programming platform-independent stateful openflow applications inside the switch. ACM SIGCOMM Computer Communication Review, 2014, 44(2): 44-51.

[105] Shin S, Gu G. Attacking software-defined networks: A first feasibility study//Proceedings of the ACM SIGCOMM workshop on Hot topics in software defined networking. Hong Kong, China, 2013: 165-166.

[106] Afek Y, Bremler-Barr A, Shafir L. Network anti-spoofing with SDN data plane//Procedings of the IEEE International Conference on Computer Communications. Atlanta, USA, 2017: 1-9.

[107] Pereira F, Neves N, Ramos F M V. Secure network monitoring using programmable data planes//Proceedings of Network Function Virtualization and Software Defined Networks. Berlin, Germany, 2017: 286-291.

[108] Freire L, Neves M, Schaeffer-Filho A, et al. POSTER: Finding Vulnerabilities in P4 Programs with Assertion-based Verification//Proceedings of ACM Special Interest Group on Security, Audit and Control. Dallas, USA, 2017: 2495-2497.

[109] Neves M, Levchenko K, Barcellos M. Sandboxing Data Plane Programs for Fun and Profit//Proceedings of the SIGCOMM Posters and Demos. Saint Louis, USA, 2017: 103-104.

[110] Barefoot. Website. https://barefootnetworks.com

[111] Chen L, Chen G, Lingys J, et al. Programmable Switch as a Parallel Computing Device. arXiv preprint arXiv:1803.01491, 2018.

[112] Jepsen T, Moshref M, Carzaniga A, et al. Life in the fast lane: A line-rate linear road//Proceedings of the Symposium on SDN Research. Los Angeles, USA, 2018: 10-16.

[113] Arasu A, Cherniack M, Galvez E, et al. Linear road: a stream data management benchmark//Proceedings of the Thirtieth international conference on Very large data bases. Toronto, Canada, 2004: 480-491.

[114] Dang H T, Canini M, Pedone F, et al. Paxos made switch-y. ACM SIGCOMM Computer Communication Review, 2016, 46(2): 18-24.

[115] Li J, Michael E, Sharma N K, et al. Just Say NO to Paxos Overhead: Replacing Consensus with Network Ordering//Proceedings of USENIX Symposium on Operating Systems Design and Implementation. Savannah, USA, 2016: 467-483.

[116] Zhang Y, Han B, Zhang Z L, et al. Network-Assisted Raft Consensus Algorithm//Proceedings of the SIGCOMM Posters and Demos. Saint Louis, USA, 2017: 94-96.

[117] Nguyen T D, Chiesa M, Canini M. Decentralized Consistent Updates in SDN//Proceedings of the Symposium on SDN Research. Santa Clara, USA, 2017: 21-33.

[118] Luo S, Yu H, Vanbever L. Swing State: Consistent Updates for Stateful and Programmable Data Planes//Proceedings of the Symposium on SDN Research. Santa Clara, USA, 2017: 115-121.

[119] Aghdai A, Xu Y, Chao H J. Design of a hybrid modular switch// Proceedings of Network Function Virtualization and Software Defined Networks. Berlin, Germany, 2017: 1-6.

[120] Moura M, Silva F, Frosi P, et al. NERV: A constraint-free network resources manager for virtualized environments//Proceedings of Computers and Communications. Heraklion, Greece, 2017: 411-417.

[121] Michel O, Keller E. Policy Routing Using Process-Level Identifiers//Proceedings of IEEE International Conference on Cloud Engineering. Berlin, Germany, 2016: 7-12.

[122] Braun W, Hartmann J, Menth M. Scalable and reliable software-defined multicast with BIER and P4//Proceedings of International Symposium on Integrated Network Management. Lisbon, Portugal, 2017: 905-906.

[123] He C H, Chang B Y, Chakraborty S, et al. A Zero Flow Entry Expiration Timeout P4 Switch//Proceedings of the Symposium on SDN Research. Los Angeles, USA, 2018: 19-20.

[124] Uddin M, Mukherjee S, Chang H, et al. SDN-based service automation for IoT//Proceedings of International Conference on Network Protocols. Toronto, Canada, 2017: 1-10.

[125] Edwards T G, Ciarleglio N. Timestamp-Aware RTP Video Switching Using Programmable Data Plan//Proceedings of ACM SIGCOMM Industrial Demo. Saint Louis, USA, 2017: 1-2.

[126] Sivaraman A, Kim C, Krishnamoorthy R, et al. Dc. p4: Programming the forwarding plane of a data-center switch//Proceedings of the 1st ACM SIGCOMM Symposium on Software Defined Networking

Research. Santa Clara, USA, 2015: 2-9.

[127] Geyer F, Winkel M. Towards Embedded Packet Processing Devices for Rapid Prototyping of Avionic Applications//Proceedings of 9th European Congress on Embedded Real Time Software and Systems. Toulouse, France, 2018: 1-9

[128] Wu Z, Madhyastha H V. Rethinking cloud service marketplaces//Proceedings of the 15th ACM Workshop on Hot Topics in Networks. Atlanta, USA, 2016: 134-140.

[129] Signorello S, State R, François J, et al. NDN. p4: Programming information-centric data-planes//Proceedings of 2016 IEEE NetSoft Conference and Workshops (NetSoft). Seoul, Korea, 2016: 384-389.

LIN Yun-Sen-Xiao, born in 1994, Ph.D. candidate. His research interests include software-defined networking and programmable data plane.

BI Jun, born in 1972, Ph.D. , Chang Jiang Scholar Distinguished Professor. His research interests include cyberspace security, software-defined networking, network architecture, and source address verification.

ZHOU Yu, born in 1993, Ph.D. candidate. His research interests include software-defined networking and programmable data plane.

ZHANG Cheng, born in 1987, Ph.D. His research interests include software-defined networking and programmable data plane.

WU Jian-Ping, born in 1953, Ph.D. , academician of Chinese Academy of Engineering. His research interests include cyberspace security, source address verification, IPv4 and IPv6 transition, and network architecture.

LIU Zheng-Zheng, born in 1994, M. S. candidate. His research interests include network architecture.

ZHANG Yi-Ran, born in 1995, Ph.D. candidate. Her research interests include data center network.

Background

Programming Protocol-Independent Packet Processors (P4) enable administrators to customize the packet forwarding behavior of the switch, improving the programmability of data plane and the flexibility of packet processing, which provides a new solution for solving the problems in the current network architecture and designing new network applications. This paper conducts a survey study on the research progress of P4 and its applications, including heterogeneous platforms, compilers, development tools, load balancing, network measurement, network security, etc. At last, this paper discusses the future research trends in this area.

The work is supported by National “Thirteenth Five-Year” Key R&D Plan of China under Grant No. 2017YFB0801700 and the National Nature Science Foundation of China under Grant No. 61472213. This project aims to make advances to the programmability of data plane. This paper summarizes the research progress of P4 and its applications in recent years.