综述

可编程数据平面技术综述

唐鑫新[1,2]，曾学文[1,2]，凌致远[1,2]，宋磊[1,2]
（1. 中国科学院声学研究所国家网络新媒体工程技术研究中心，北京 100190；
2. 中国科学院大学，北京 100049）

摘　要：在软件定义网络中，可编程数据平面提供的编程能力是网络功能虚拟化的基石。可编程数据平面技术的核心是可编程能力与数据包处理性能。首先从数据平面的可编程性出发，探讨现有数据平面的数据包处理抽象。然后，分别对数据平面实施的目标平台与对应平台上的主要流表算法进行介绍，详细论述现有数据平面技术。最后，探讨了高性能数据平面技术存在的关键挑战。

关键词：可编程数据平面；软件定义网络；网络功能虚拟化；高性能数据包处理
中图分类号：TP393
文献标志码：A
doi: 10.11959/j.issn.1000–0801.2023093

Overview of programmable data plane technology

TANG Xinxin[1,2], ZENG Xuewen[1,2], LING Zhiyuan[1,2], SONG Lei[1,2]
1. National Network New Media Engineering Research Center, Institute of Acoustics, Chinese Academy of Sciences, Beijing 100190, China
2. University of Chinese Academy of Sciences, Beijing 100049, China

Abstract: In software defined network, the programmability provided by the programmable data plane is the cornerstone of network function virtualization. The core of programmable data plane technology is programmability and packet processing performance. Firstly, the packet processing abstraction of existing data planes from the programmability of data planes was explored. Then, the existing data plane related technologies were discussed in detail in terms of the target platforms and related algorithms for data plane implementation, respectively. Finally, the key challenges of high-performance data plane technologies were discussed.

Key words: programmable data plane, software defined network, network function virtualization, high-performance packet processing

0　引言

传统网络的分层设计与固定功能的传统网络设备越来越难以满足不断涌现的网络需求。传统网络设备功能固定，新增功能需要由设备厂商进行设计、测试与部署实现[1]，通常开发部署时间周期长[2]，且对于一些小众化的功能需求，如私有协议的支持，设备厂商很可能考虑成本问题而

收稿日期：2022–12–19；**修回日期：**2023–04–15
基金项目：中国科学院 C 类战略性科技先导专项课题（No.XDC02070100）
Foundation Item: Strategic Priority Research Program of Chinese Academy of Sciences (No.XDC02070100)

无法顾及。当下的新型网络服务，如 5G、物联网、大规模云计算以及海量机器学习和大数据应用等，要求网络设备快速且可靠地对其应用需求进行支持[3]。这种强烈的需求催生了一种新的增强网络可编程性的思路：将网络的控制平面与数据平面进行分离，并将可编程性从使用传统设备提供的少量应用程序接口（application program interface，API），逐渐下沉到数据平面。拥有灵活可编程能力的数据平面相对于传统的网络设备，具备迅速的需求满足能力、支持各种新型协议的协议无关能力、网络功能的自定义与快速迭代能力。

可编程数据平面的概念是由软件定义网络[4]（software defined network，SDN）中的数据平面发展而来的。SDN 的数据平面概念可以追溯到 SDN 早期的转发与控制单元分离[5]（forwarding and control element separation，ForCES）提出的管理单元（control element，CE）与转发单元（forwarding element，FE）的分离。在 ForCES 架构中，CE 与 FE 在通信与物理实施两个层面进行分离，但又共同组成一个网元[6]。与 ForCES 架构中的数据平面相比，OpenFlow 体系中提倡的分离则更为彻底：数据平面由更简单的交换机构成，且整个网络的控制平面由一个集中控制器组成[7]。尽管 SDN 从提出便关注网络的可编程性，但其早期的大部分愿景集中在控制平面的可编程性上[8]：OpenFlow 不仅对新协议的支持较为保守，并且仅预定义了一小部分的数据包处理原语，这使得 OpenFlow 交换机在快速支持新型协议与灵活构建数据包处理逻辑两方面严重受限。OpenFlow 数据平面的灵活性匮乏促使了可编程的协议无关包处理器（programming protocol-independent packet processor，P4）[9]与协议无关转发（protocol-oblivious forwarding，POF）[10]等协议无关技术的出现，正式打开了数据平面在可编程与易重构方向的发展局面。在可编程数据平面技术方面，目前的研究主要集中于简单且丰富的抽象表达能力、高速的转发处理性能、面向状态的监测管理能力、稳健的安全性以及高效的可重构性等方面。

1 数据平面的可编程性

传统网络设备管理与转发的强耦合导致的管理维护困难等问题催生了数控分离。数控分离是网络可编程性的基本推动力，将控制平面与数据平面在物理与逻辑上进行分离，两者依靠标准且开放的 API 进行交互[5,7,9-11]。ForCES 于 2003 年在 RFC3654 中被提出，并于 RFC3746 中得到确认，是 SDN 的早期架构之一。在 ForCES 架构中，控制平面与转发平面的分离主要体现在两个方面：一是通信的分离，CE 与 FE 的通信基于标准协议而非专用的 API；二是物理实施层面的分离，即 CE 与 FE 在物理设备上实质性的分离，但同时 CE 与 FE 又共同构成一个网元。需要注意的是，ForCES 的控制平面功能仍基于分布式协议[6]。随着 OpenFlow 的出现，SDN 架构的意义与实用性才正式起步。OpenFlow 使用流表与一组预定义的动作构成数据包处理逻辑的基石，并使用一组标准化的流表增删接口为控制平面提供流表与动作的配置能力。网络的控制平面通过北向接口接收高级网络范围的策略或意图，并通过数据平面提供的抽象配置能力将这些策略或意图编译下移至数据包处理策略级别，最后通过南向接口将这些策略配置到各个数据平面设备中。在 OpenFlow 体系中，数据平面的能力更纯粹，数据平面的设备不需要实现数据包转发策略的所有逻辑（比如运行路由协议来构建路由表），而是通过南向接口从控制平面获取这些策略。尽管 OpenFlow 为数据平面定义的可编程能力有限，但其意义在于正式推动了数控分离，使数据平面向着高可编程、易重配置的方向发展。数控分离的架构极大地增强了网络的可编程性，从本质上来说，数控分离将高层的策略意图与低层的数据包转发逻辑解耦，数

据平面负责为控制平面提供数据包处理的抽象表达能力，并基于控制平面下发的策略构建数据包转发逻辑，而控制平面负责接收来自北向的高层应用（如网络管理）意图并将其编译为数据平面级别的策略。几种不同可编程体系之间的比较见表 1。

表 1　几种不同可编程体系之间的比较

体系	特点	相同点
ForCES 体系	使用逻辑功能块构建网络功能；控制平面功能基于分布式协议；CE 与 FE 构成一个网元等	转发与控制在通信与物理层面进行分离
OpenFlow 体系	使用管道、流表以及预定义动作构建网络功能；基于流的数据包处理；集中控制器等	
协议无关编程体系	协议无关；更多的有状态处理；支持更丰富的管道构建方式以及更灵活的处理指令方式等	

数据平面通过执行一系列操作来处理网络数据包，基本功能包括解析、分类、修改、封装与转发。除此之外，一些数据平面还具备数据包调度、过滤、计量与流量整形等功能。可编程数据平面需要将这些基本功能抽象为数据包处理原语，并提供能够将这些原语组合为所需管道的抽象编程能力（如一种编程语言），从而提供功能的可重构能力。传统网络设备中，通常网络功能是固定的。这意味着一台传统的 L3 转发设备无法处理虚拟扩展局域网（virtual extensible local area network，VxLAN）数据包的转发，而可编程数据平面通过功能的重构可以对 VxLAN 进行快速的支持。通俗地说，数据平面的可编程的基本原理为，控制平面根据网络应用的需求，基于数据平面提供的可编程抽象（如数据平面语言），构建满足需求的网络功能的中间表示（如流表、表项以及指令块等），而数据平面能够快速地接收、编译这些从控制平面下发的中间表示，将控制平面构建的网络功能运行在实施平台之上，从而实现了网络功能的虚拟化。

数据平面的可编程能力代表了新功能被引入网络基础设施的方法、粒度与时间尺度[12]。尽管一些早期的研究人员认为基于硬件的数据平面不可编程，但随着一些可重配置硬件与配套的编程语言的发展，基于硬件实现的数据平面也被认为具有一定的编程能力。总的来说，数据平面的可编程能力指的是交换机将数据包的处理逻辑暴露给控制平面，从而能够达到的系统、快速以及全面的可重配置程度[3]。另外，数据平面的可编程能力在一定程度上影响了网络的灵活性[13]。因为高度可重构的数据平面能够及时响应网络中不同种类的新变化，包括需求的变化、约束的变化以及数据面状态的变化。

2　数据平面抽象

数据平面技术之间的差异的一个主要特征为提供给控制平面的数据包处理逻辑抽象，即数据包处理原语（指令集）和可用于组合原语以实现所需管道的能力（如编程语言）。具体来说，这种能力可以拆分为构成数据包处理整体架构的处理逻辑抽象以及用于实现这种架构以及数据包处理功能的领域特定语言（domain-specific language，DSL）。本节将讨论可编程数据包处理逻辑的两种架构以及用于描述和编程数据包处理逻辑的 DSL。

2.1　数据包处理逻辑抽象

可编程数据平面采用的数据包处理逻辑目前主要分为两种：数据流图（data flow graph）与匹配动作（match-action）管道（pipeline）。

2.1.1　数据流图抽象

数据流图主要在通用系统设计[14]与机器学习[15]中被广泛使用。在可编程数据平面的数据流图抽象中，整个数据包处理逻辑被描述成一张包含节点与有向边的图，每一个节点代表一个基本的数据包处理阶段，有向边指示了数据包接下来将移动到的处理节点的方向。数据流图抽象实质上将

数据包处理逻辑分割成不同的处理节点。这种模块化分割带来的好处是开发人员只需要开发一次处理节点，便能够得到高可复用的处理模块，这为复杂数据包处理流程的构建带来了极大的便捷。数据流图处理引入可编程数据平面的案例最早是 Click 软件路由器[16]。单个数据包在 Click 的流图中移动，节点表示简单的包处理操作，例如包头解析、校验和的计算与验证、字段重写、访问控制列表（access control list，ACL）等[3]。早期的数据流图抽象的输入是单个数据包，而目前实现于软件架构的数据流图抽象大多采用批处理机制，即数据流图的输入是成批的数据包[17]。目前这类抽象的具体实现有 ClickOS[18]、FastClick[19]、矢量数据包处理（vector packet processing，VPP）[17]、BESS[20]与 NetBricks[21]等。值得一提的是，尽管 ForCES 中的 FE 架构[22-23]没有明确表示，但从本质上来说，也属于数据流图抽象：FE 架构的数据包处理逻辑表示为逻辑功能块（logical functional block，LFB）与有向边组合成的有向图。ForCES 数据包处理逻辑结构如图 1 所示。LFB 可认为对应了数据流图的计算节点，有向边则代表数据包下一步应该前往的 LFB 的方向。

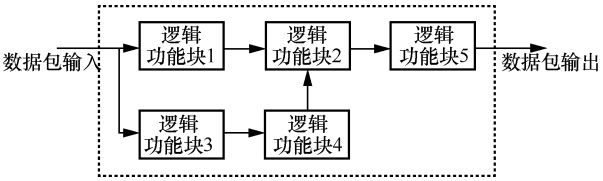


图 1　ForCES 数据包处理逻辑结构

尽管数据流图抽象给予了数据包处理逻辑的构建相当程度的灵活性与可扩展性[18,24]，但也有文献指出高灵活性会令最终的处理逻辑设计变得复杂零散，异构性使高级网络范围的抽象复杂化，同时也阻碍了性能优化[25]。

2.1.2　匹配动作管道抽象

匹配动作管道抽象的核心是匹配动作表（match-action table，MAT）[26]，一张 MAT 指定了一个数据包头字段的子集，而 MAT 的表项指定该包头字段子集的值范围。当数据包在包头子集上的值落在某个表项所指定的范围内时，称数据包匹配（match）该表项，此时数据平面会对数据包执行该表项所绑定的处理动作（action）。处理动作原语包括但不限于重写数据包内容、封装解封装隧道头、转发或者丢弃数据包以及将数据包处理推迟至后续流表（Goto-Table）处理等。匹配动作管道数据处理抽象如图 2 所示，早期的匹配动作管道抽象使用一张流表来完成数据平面的所有数据处理逻辑，但诸多包头字段的笛卡儿积造成的表项数量爆炸问题极大地提高了数据平面的管理难度。为了应对表项数量爆炸的问题，目前的匹配动作管道抽象使用一系列的匹配动作表构成数据包处理逻辑[7,9-10,27]，数据包通过处理动作 Goto-Table 在表之间进行跳转。实际上，此时匹配动作管道抽象与数据流图抽象的区别已经并不那么明显[7]。多匹配动作表的引入使单个表所表示的处理逻辑抽象更加模块化，而 Goto-Table 处理动作的引入则指示数据包下一步前往的匹配动作表，匹配动作表与有向边构成的有向无环图（directed acyclic graph，DAG）[28]与数据流图抽象不谋而合。

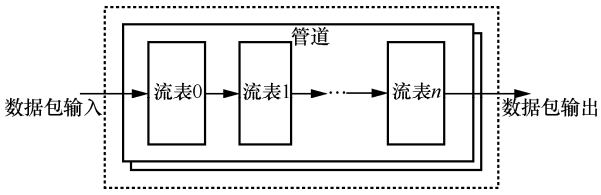


图 2　匹配动作管道数据处理抽象

匹配动作管道抽象的特点是程序员可以按照流的概念配置数据平面功能，而这些流的概念由匹配动作表所指定的包头字段子集定义。程序员能够轻松地配置流表内的表项，从而达到对不同流执行不同处理行为的目的，而这种执行行为的添加、删除或修改表项的操作被匹配动作管道抽象的标准化 API[29]所定义。

OpenFlow 最早将匹配动作管道抽象推广至可编程数据平面，并在后续的版本中引入多匹配动作表构建管道的规范[7]。匹配动作管道抽象基于流的概念构建数据包处理逻辑，但作为该抽象的推广者的 OpenFlow 在对数据包头字段的支持上却极为保守。从版本 1.0 到 1.5，OpenFlow 支持的数据包头字段仅从 12 个增加到 45 个，显然这种只能随协议演进增加的字段支持并不能满足灵活性的要求，并且对新匹配字段的支持都需要重新编写数据平面与控制平面两端的协议栈以及数据平面的数据包处理逻辑[30]。尽管如此，遵循 OpenFlow 规范的交换机，如 Open vSwitch[27]仍运行在有限的数据包头字段集合上。实际上，基于软件架构的数据平面可以通过自定义偏移与长度来灵活地支持自定义数据包头字段，如 Kangroo[31]、CAFE[32]等自定义包头解析器技术。然而尽管这类包头解析技术拥有比特级的解析精度，但这种高灵活度也带来了资源开销过大的问题[33]。另外，POF[10,34-35]和 P4[9]在匹配动作管道抽象中将支持的数据包头字段推广到任意包头字段，并提出了语义丰富的处理动作指令集以及灵活的匹配动作表构建方法，本文将在后续进行详细讨论。

总的来说，数据流图抽象通过较为独立的逻辑功能块构建网络功能，而匹配动作处理管道抽象则通过流表之间的跳转、表项与处理指令之间的组合实现网络功能，这意味着在数据流图抽象中，数据平面处理是以数据包为对象的，而在匹配动作管道处理抽象中，数据平面是以数据流为目标进行处理的。此外，匹配动作管道抽象更适配硬件流水线的实现，这一点已经得到产业界诸多成熟案例的支持[36]。从长远来看，以性能为目标的可编程数据平面通常依托可编程硬件实现，在这种背景下，匹配动作管道抽象凭借其在硬件流水线上的优良适配性拥有比数据流图抽象更广泛的应用前景。

2.2　可编程数据平面语言

适用于可编程数据平面的 DSL 可以分为数据平面描述语言与数据平面编程语言。数据平面描述语言拥有较高的抽象描述能力，能够描述数据平面架构及其构成组件。基于这种高度的抽象能力，数据平面描述语言利用不同的编译器可以支持多种硬件平台，从而实现良好的平台无关性。数据平面编程语言的抽象描述能力与平台无关性相对较差，但在数据平面上具备良好的数据结构（如指针、引用和队列等）以及灵活的算法实现能力，这是当前数据平面描述语言所不具备的。此外，数据平面编程语言在特定硬件平台上对硬件细节的编程粒度更细，这通常有益于性能提升。

2.2.1　数据平面描述语言

数据平面描述语言能够描述数据平面结构及其组件，如匹配动作管道中字段的定义、流表的顺序等）。P4[9]是目前流行的数据平面描述语言，提供了良好的数据平面的高级抽象，主要体现为如下 3 个设计目标。

- 可重配置性：在不更换实施硬件设备的前提下能够对数据包处理逻辑进行按需重配置。
- 协议无关性：数据包处理逻辑需要保证协议无关，保证快速灵活地支持各种新协议。
- 平台无关性：提供良好的高层抽象，使程序员无须关注底层实施平台，从而提供无缝的平台移植性[37-38]。

P4 通过将 P4 代码映射为通用的基于匹配动作管道的转发模型来抽象数据包处理逻辑。P4 代码从逻辑上可以分为 5 个部分。

（1）数据声明：定义需要解析的协议字段、自定义元数据、固有元数据、外部对象与结构定义[39]等，其中协议字段通过特定的顺序与字段长度声明。

（2）解析器逻辑：解析器逻辑部分指定如何、何时以及以什么顺序解析在数据声明中声明的数

据包头字段。这部分代码会被映射到转发模型中的解析器和逆解析器，解析器负责从数据包中将声明的数据包头字段的值提取出来，包头字段与数据包载荷分开存放。而逆解析器则将最终计算得到的值重新封装至数据包中对应的位置。

（3）表声明：P4 基于匹配动作管道抽象，所声明的表符合匹配动作表的结构，即匹配域（匹配字段）、匹配方式以及相应的执行动作等。

（4）动作声明：数据包处理动作的声明，可以通过 P4 提供的协议无关处理原语（指令集）的灵活组合对处理进行编程。

（5）控制流：控制流定义了数据包在多张匹配动作表之间的跳转方式[37]，将匹配动作表构造成 DAG，是构建匹配动作管道的核心逻辑。

P4 的通用转发模型从结构上来看分为解析器/逆解析器、出口/入口管道（egress/ingress pipeline）以及缓冲区。其中入口管道执行与出口无关的数据包处理（如复制、丢弃等），并指定数据包的出口意图（如转发的端口）。出口管道负责进一步的数据包头修改，如重写目标媒体访问控制（media access control，MAC）地址。

P4 作为数据平面高级语言，提供了描述数据平面的高级抽象，尽管 P4 提出了平台无关性的目标，也为不同的目标平台设计了相应的 P4 编译器，同时有一些技术如现场可编程门阵列（field programmable gate array，FPGA）上的高级综合（high-level synthesis，HLS）技术支持将 P4 程序转换为 Verilog 等硬件描述语言[38,40]，但 P4 程序不能在不同的架构上移植[41]，相应的架构包括 NetFPGA[42]的 Simple SumeSwitch 架构、BMv2[43]的 V1Model 架构以及 Tofino[36]的协议无关交换架构（protocol-independent switching architecture，PISA）。

2.2.2 数据平面编程语言

数据平面编程语言可以在数据平面实现新算法（如数据包调度机制、数据包分类算法）[44]，这类的语言有 POF[10,34]、Domino[45]、packetC[46]、Click[16]以及 PX[47]等，其中具有代表性的是 POF 与 Domino。

POF 是 OpenFlow 的直接演进，旨在与其互操作并在数据包处理定义中实现更大的灵活性。POF 可被看作 OpenFlow 的一个协议无关扩展，POF 中的字段使用{Type, Offset, Length}描述，其中 Type 字段可以描述包括数据包头以及元数据等多类型的数据域或状态域，从而支持不同应用场景的需求[26]。在 POF 原始版本中，Type 设置为 0 表示数据包头字段，为 1 表示元数据字段。Offset 是相对基准位置的偏移。Length 表示字段的长度，偏移与长度均以 bit 为单位。例如，目的 MAC 字段表示为{0, 0, 48}。需要注意的是，在 POF 中，Offset 的基准位置可以通过 POF 提供的 Move Packet Offset 指令进行移动，有研究利用该特性减小流表的占用空间[48]。POF 提供了一个协议无关的流指令集作为数据包处理逻辑的编程原语，允许对数据包头、匹配动作表和统计计数器等元素进行处理。程序员基于流指令集构建数据包处理逻辑，并将编写的代码编译成不同的可执行文件下发给数据平面，从而达到无须更改数据平面即可支持新协议的目的。值得一提的是，在 POF 的抽象转发模型中，并未设定专门的可重构解析器，这是因为 POF 通过表的重定向和专门定义的决策表将数据包头解析混合到匹配动作管道中。

Domino 面向基于 Banzai 模型的硬件交换机，是一种命令式语言，其语法类似于 C 语言，允许使用数据包事务来编写数据包处理逻辑。数据包事务（packet transaction）是一个按序执行的数据包处理代码块，该代码块是原子的且与其他代码块独立。Domino 数据包事务的执行是在可编程线速硬件交换机的机器模型上进行的，该模型称为 Banzai[45]。Banzai 是一个可编程的硬件数据包处理管道，它包含一系列在每个时钟周期同步执行的匹配动作阶段（match-action stage）。在每个阶段，传入数据包的包头字段与匹配表进行匹配，

然后采取相应的操作。Domino 程序由 Domino 编译器翻译成 Banzai 目标文件，该编译器基于全有或全无模型（all-or-noting model）实现：如果编译成功，程序将以线速运行；如果程序不能以线速运行，那么将无法编译。相对于软件交换机，Domino 面向的硬件交换机通过降低可编程性来保证线速性能[49]。

2.2.3　数据平面语言的局限

数据平面语言尽管为程序员提供了一定的数据包处理逻辑的抽象与编程能力，但目前仍存在很多局限。首先，P4 作为代表性的数据平面描述语言，平台无关性是其三大设计目标之一，然而使用 P4 程序对 FPGA 等平台编程虽然已经得到诸如 HLS 等技术的支持，但这些技术要么将 P4 程序转化为底层平台支持的代码，要么将 P4 程序直接编译成底层平台直接支持的配置文件。前者的问题在于转化的代码鲁棒性差，调试困难，而后者很难编译复杂代码。此外，P4 作为高级数据平面语言，在语法与功能上依然存在较多局限[37]。尽管 P4 也提出了比较完备的数据处理原语，并且也有初步研究[50]指出有限的指令集可以通过一些机制来解决观察到的网络约束，但当前 P4 的语法以及其原语（指令）集并不能提供许多图灵完备语言中的典型结构，如迭代、递归、动态内存分配、指针或引用、排队、排序、多路复用等，而这些结构在实现复杂的网络功能时是极为便利且有效的。尽管如此，P4 也并未因为这种灵活性的限制而考虑将混合架构作为目标平台。其次，数据平面编程语言能够在数据平面中灵活实现任意的报文处理算法，在功能扩展方面拥有很强的灵活性，但是这类语言的局限性在于对目标平台有着很强的依赖性。

总的来说，目前的数据平面语言在高级抽象描述能力与充足且全面的典型编程结构之间无法达到一个良好的兼顾，而后者对于网络功能的效率以及安全性有着直接影响。另外，平台无关性也会影响数据平台部署的灵活性，然而目前还难以做到在平台无关的前提下充分抽象目标平台的细节，而这会影响数据平面编程的粒度[44]。

3　目标实施平台

目前网络功能的实现广泛存在于通用 CPU 平台[27]、专用集成电路（application-specific integrated circuit，ASIC）[36]、FPGA[42]与网络处理器（network processor，NP）[51]以及一些混合架构[52-53]中。除此之外，智能网卡（smart NIC）与面向数据中心的数据处理单元（data processing unit，DPU）致力于在网络边缘增强数据平面的可编程性。通常来说，数据平面的高性能通过专用和专业的组件的硬件数据平面取得，如 ASIC 交换机等，而灵活性则以通用 CPU 平台的软件数据平面为代表。尽管目前许多硬件设备的可编程性已经取得了长足的进步，但高性能与灵活性仍然是目前数据平面实施平台难以兼顾的两个指标。尽管通常将软件与硬件分开讨论，但一些相关技术正在模糊这种分界，混合架构并不少见：硬件数据平面仍然可借用通用 CPU 来应对复杂的计算场景，如深度包检测（deep packet inspection，DPI）以及广域网（wide area network，WAN）优化等。目前软件数据平面也应用了许多特定领域的硬件能力，如接收侧缩放（receive side scaling，RSS）、数据直接输入输出（data direct input/output，DDI/O）以及传输控制协议段卸载（transmission control protocol segment offload，TSO）等。此外，越来越多的高性能网络功能也从 CPU 平台卸载到智能网卡或者其他硬件平台进行处理。

3.1　通用 CPU 平台

通用 CPU 平台上的数据平面通常部署在数据中心，以支持广泛的数据包处理功能。事实上，通常认为通用 CPU 平台具有最强的灵活性，能够实现各种复杂的数据包处理算法。在通用 CPU 平台上实现网络功能虚拟化的可编程数据平面已经

有许多成熟案例，如 VPP[17]、BESS[20]、FastClick[19]、NetBricks[21]、PacketShader[54]和ESwitch[55]。现代服务器硬件的复杂结构让网络数据包处理过程变得十分烦琐且缓慢，因此通用CPU平台上的主要研究集中在低时延I/O与高性能处理。目前加速数据包I/O方面有多项研究成果，典型案例有Netmap[56]、Intel数据平面开发套件（data plane development kit，DPDK）[57]、PacketShader[54]、FD.io[58]，以及带有扩展的伯克利数据包过滤器（extended Berkeley packet filter，eBPF）的快速数据路径[59]（express data path，XDP）等开发套件或工具。Netmap、DPDK等套件利用内核旁路（kernel bypass）技术直接将网卡读写数据包的内存映射到用户空间，从而消除了耗时的上下文切换和数据包复制，代价是不能使用任何内核网络栈。而XDP在传统网络栈的最低点（XDP位置）提供数据包处理，在提供高能数据包处理的同时允许使用一些传统的网络接口。数据包批处理技术广泛应用于各种数据平面开发套件之中，能够均摊开销[60]并提高数据的局部性，从而减小CPU缓存的失配率[17]。现代CPU的单指令多数据（single instruction multiple data，SIMD）流技术也被用来加速布谷鸟哈希查找[61]、流处理指令执行[62]等耗时操作。其他典型的技术包括非均匀存储器访问（non-uniform memory access，NUMA）、DDI/O、缓存行对齐（cache line align），以及内核亲和性等。

通用CPU平台主要部署于对转发性能要求不高但涉及复杂计算的应用场景，被广泛应用于前期的性能支持验证，为后续转移到专用硬件平台做前期工作，也用于与硬件交换机构成软硬一体化编排（如腾讯的TGW、微软的Azure云等公有云）。通用CPU平台的灵活性也适用于需求小众的校园网等场景下的快速功能迭代支持。通用CPU平台的灵活性也表现在不强制要求任何特定的处理模式，因此，将网络功能卸载至智能网卡或DPU等专用处理设备的这类混合设计已成为流行趋势，基于纯软件的网络功能将会越来越少。然而，通用CPU平台极强的灵活性使其仍在云计算等涉及复杂计算的应用中发挥着至关重要的作用，低时延I/O与高效率处理仍然是软件数据包处理技术的研究目标。

3.2 ASIC

数据平面领域的ASIC是一种专门为高性能数据包处理设计优化的芯片。早期数据平面的ASIC交换设备是不可编程的，为了摆脱低性能的软件数据平面，一些受精简指令集计算机（reduced instruction set computer，RISC）启发的硬件管道架构（如可重配置匹配动作表[63]模型）开始出现，打开了可编程ASIC的发展局面。目前最为知名且流行的ASIC交换机是Intel的Tofino[36]交换机，其包含4个并行的匹配动作管道并能维持6.4 Tbit/s的速率，而Tofino 2交换机则能维持12.8 Tbit/s。其他可编程ASIC有Intel FlexPipe、Cisco Nexus 3000、Xilinx SDNet、Broadcomm Trident、Innovium等。可编程ASIC的架构大多基于匹配动作管道抽象，例如Intel的Tofino的PISA。通常硬件的管道模型中，会将可重配置的解析器、逆解析器与匹配动作管道独立，以保证匹配动作管道的高效性与并行性。匹配动作管道由匹配动作表和用于执行指令动作的算术逻辑单元（arithmetic logic unit，ALU）组成。通常这些匹配动作表通过不同的硬件来实现不同的匹配能力。精确匹配（exact match，EM）表通常由静态随机存取存储器（static random access memory，SRAM）实现，而通配符匹配（wildcard match）表则由三态内容可寻址存储器（ternary content addressable memory，TCAM）实现。

ASIC拥有极高的数据包处理能力，特别适用于对数据包处理速度与转发速度要求较高并且不依赖复杂计算的转发场景。而对于一些复杂计算场景，如前文提到的DPI与WAN优化等，ASIC

则发挥有限。此外，尽管目前可编程交换芯片已经拥有较高的灵活性，但仍然存在开发周期长的问题。在实际网络场景中，ASIC 实现通常以集成单片系统（system on chip，SoC）的混合架构出现。在混合架构中，ASIC 不支持的复杂处理功能（如监控与检测等）运行在 ARM 处理器上的 Linux 系统中，从而构成了具有快速硬件路径和高度可编程软件路径的混合路径数据平面。

3.3　FPGA

FPGA 是由可编程逻辑块（configurable logic block，CLB）、输入输出模块（input/output block，I/OB）、内部连线（interconnect）以及其他内嵌单元组成的可编程半导体器件。与可编程的 ASIC 相比，FPGA 的开发周期更短且成本更低，此外，随着时钟频率和内存带宽的提高，FPGA 缩小了与 ASIC 的差距，这也导致大量高性能网络数据平面研究利用 FPGA 作为原型设计进行实验与验证。FPGA 在可编程数据平面上的典型实现是 NetFPGA。基于 Xilinx Virtex-7 芯片的 NetFPGA-SUME 板卡可以通过扩展连接器进行多板卡扩展，基于这种多板卡互联，可以实现 300 Gbit/s 的无阻塞交换[44]。其他实现有可编程网络平台 C-GEP[64]以及面向 100 Gbit/s 网卡开发的开源设计平台 Corundum[65]等。

通常 FPGA 通过 Verilog 或超高速集成电路硬件描述语言（very-high-speed integrated circuit hardware description language，VHDL）等硬件描述语言进行编程，尽管 HLS 或者专用编译器也允许使用像 C 或 P4 这样的语言对 FPGA 进行编程[38,40]，但存在代码鲁棒性差或复杂代码难以编译的问题。相对于 ASIC，FPGA 的优势在于开发周期短、成本低且可编程性更强，这使得基于 FPGA 构建的高性能网络数据平面原型在学术研究领域广受欢迎，并且相关研究众多。而相对于通用 CPU 平台，FPGA 凭借其高性能并行硬件流水线取得了处理性能的优势。与 ASIC 类似，FPGA 难以应对复杂计算问题，并受限于资源，通常以混合架构部署在实际网络环境中，如集成 SoC 或多处理器单片系统（multiprocessor system on a chip，MPSoC）[66]。此外，将数据包处理工作从通用服务器卸载至基于 FPGA 的智能网卡也是混合结构中较为常见的做法[67]。

3.4　NP 与 smart NIC

与可重构硬件相比，NP 用作 SDN 数据平面实施平台的范围小得多[44]。基于网络处理器的数据平面实现在功能布局和操作方面的灵活性较低，可编程性通常仅限于配置数据平面功能的参数，例如队列容量、调度机制、数据包头过滤器等。随着当前 FPGA 的时钟频率、内存带宽的提高，以及与 FPGA 相关研究的不断深入，NP 在性能与可编程性上已经难以与 FPGA 比肩。目前 NP 的相关应用主要集中于构建智能网卡。通常来说，智能网卡更偏向一个抽象网络设备概念，指的是以通用 CPU 平台设备为宿主，负责卸载 CPU 处理任务的 NP、FPGA 或 ASIC 可编程实现。智能网卡的功能通常有传输控制协议（transmission control protocol，TCP）卸载、VxLAN/通用路由封装（generic routing encapsulation，GRE）隧道报文卸载、安全套接字层（secure socket layer，SSL）/互联网络层安全协议（Internet protocol security，IPSec）加解密卸载、防火墙卸载等[68]。

4　主要流表算法

可编程数据平面的算法研究涉及多个方面，鉴于匹配动作管道抽象是目前主流的可编程数据平面抽象，而其中最重要的是可重配置匹配动作表方向的算法研究，因此，本节主要讨论可重配置匹配动作表相关算法的实现、加速与更新方面的算法。

4.1　流表实现算法

匹配动作管道抽象中匹配动作表的实现多种多样，比如 OpenFlow 实现的精确匹配表（exact match table）、最长前缀匹配（longest-prefix match，

LPM）表与通配符匹配表（wildcard match table）。其中精确匹配表要求数据包字段在每一比特上的值均与表项给定的值相同，而通配符匹配表则允许一些比特位上是任意值，而最长前缀匹配表则适用于 IP 地址前缀匹配。类似的还有 POF 实现中的直接表（direct table，DT）、精确匹配表、最长前缀匹配表、掩码匹配（mask-match，MM）表[10]。总的来说，在匹配动作表中，最为常用且重要的就是精确匹配表、最长前缀匹配表以及通配符匹配表。

精确匹配表通常基于哈希实现，因此无论在软件平台还是硬件平台，都拥有极高的查表效率。基于哈希实现的精确匹配表的问题在于哈希冲突、如何在保证高性能的同时支持海量表项数量以及如何实现高并发，因此精确匹配表方面的挑战主要集中在硬件平台上。软件平台上的研究集中在通过共享内存和消息传递设计并发和无锁哈希表[69]，以及利用现代 CPU 的 SIMD 技术（如 Intel AVX2 和 AVX-512 扩展）加速哈希表[61]。目前常见的实现是布谷鸟哈希[70-71]，而最近提出的 Ludo 哈希[72]能够实现更低的内存占用，并保证高性能的并发读写。硬件平台方面的高并发方案主要有 FPGA 上基于 SRAM 实现的 Parallel d-pipeline[73]、FASTHash[74]等。精确匹配表在数据中心中被广泛应用，在转发信息库（forwarding information base，FIB）的高效实现中发挥着重要作用[75]。

最长前缀匹配表在软件平台上主要基于 Trie 结构实现，如 Binary Trie、MultiBit Trie 等。此外，一些数据平面开发套件为 IPv4 与 IPv6 提供了针对性优化的高速 LPM 类库，如 DPDK 提供的基于多级哈希的 RTE_LPM 与 RTE_LPM6[76]。在可编程数据平面领域，可能会有一些与 IP 字段长度不一致的自定义字段需要使用 LPM，这类基于多级哈希实现的 LPM 类库在扩展性上存在一定的局限。在硬件实现方面，TCAM 由于能够存储除 0 与 1 以外的第三种状态通配符“*”，并且能在一个时钟周期内完成搜索，成为 ASIC 平台实现 LPM 表与通配符匹配表的首选技术。受此启发，FPGA 平台通过利用块随机存取存储器（block random access memory，BRAM）实现 TCAM 来构建 LPM 表与通配符匹配表，因此，相关的研究集中于如何在 FPGA 平台利用 BRAM 实现高性能且低资源占用的 TCAM[77-79]。

通配符匹配或者说是多维数据包分类算法，广泛应用于流量统计、路由、防火墙等网络服务中。与 LPM 表类似，通配符匹配表在硬件平台可以通过高性能 TCAM 实现。由于通配符匹配表的表项存在优先级，在 TCAM 中，每次更新表项都需要重新对所有表项进行优先级排序，更新开销较大。此外，TCAM 的容量有限，当数据包分类规则中的字段值域是一个有限区间时，将这种规则转换为与 TCAM 兼容的规则可能会导致规则数量爆炸[75]。因此，在硬件平台实现中，如何有效利用有限容量的 TCAM 进行高效多维数据包分类是该领域的热门研究内容。

多维数据包分类算法在软件平台中大致可以分为基于维度分解的算法和基于空间划分的算法。基于维度分解的算法将每条规则分解为一定长度的多个维度，对每个维度单独进行搜索，最后合并搜索结果。经典的维度分解算法有递归流分类（recursive flow classification，RFC）[80]、并行位向量（parallel bit vector，Parallel BV）[81]、聚合位向量（aggregated bit vector，ABV）[82]等。通常维度分解算法的内存占用较高，并随着规则集规模的增大而严重升高。基于空间划分的算法将整个规则空间划分为若干个子空间，然后将规则集划分为若干组，并将其放入每个子空间。空间划分算法主要有两类：基于元组空间的算法和基于决策树的算法。前者的经典算法有元组空间搜索（tuple space search，TSS）[83]、PartitionSort[84]等，后者的经典算法有 Hicuts[85]、HyperCuts[86]、EffiCuts[87]等。基于元组空间的算法的子空间分布

不均匀问题会在一定程度上影响分类性能，而基于决策树的算法在构建决策树时同样存在内存消耗大的问题，并且在规则更新时需要进行耗时严重的决策树重构过程。在规则频繁更新的可编程数据平面中，规则更新性能是一个重要的指标，这无论对于软件平台还是硬件平台的分类算法都是一个重大挑战。

4.2　缓存加速方案

网络中的流量通常具有一定的局部性，这导致可编程数据平面中的一小部分规则在一段时间内的匹配次数要明显高于其他规则，这就是所谓的齐夫定律。事实上，Open vSwitch 利用这一特性设计了多级缓存加速方案：Microflow Cache 与 Megaflow Cache[27]。基于 POF 与 P4 的软件数据平面也采取了类似的缓存方案[88-89]，不过相对于 OpenFlow，协议无关特性让缓存方案的设计更为复杂。缓存加速同样使硬件平台的数据平面实现收益：在容量受限的高速 TCAM 快速路径中只存放一小部分热点规则[90]，而其余规则存放在服务器或者 SoC 中的慢速路径。混合架构在这种缓存加速方案中颇具竞争力，已有研究将 Open vSwitch 的多级缓存方案部署到硬件平台，实现了真正的快速转发路径。

4.3　流表更新

在 SDN 架构中，数据平面的流表规并不像传统网络设备中那样在设备启动后保持不变，而是会发生频繁的变动与更新。在这种背景下，支持流表规则快速插入、修改以及删除等特性的在线算法是目前 SDN 架构的主流数据平面流表算法。在流表更新层面，软件平台能够保持相当程度的高性能。经典的算法有 PCIU[91]，更新流表规则时无须重建数据结构，能够保证较高的更新性能。Open vSwitch 采用 TSS[83]算法作为其流表实现算法的一个原因也是 TSS 算法的更新性能较高[27]。另外，像 ITOC[92]这样较新的软件平台流表算法，也能在维持一定流表查找性能的同时保证高性能的流表规则更新，但内存占用较高。硬件平台依赖 TCAM 维持高性能的流表查找性能，但在规则更新层面相对软件平台存在劣势。TCAM 不仅存在更新规则较慢的问题，如前文所述，使用 TCAM 实现的通配符匹配表在每次更新后都需要根据优先级对表项进行一次重排序，这无疑是雪上加霜。因此，部分研究[93]利用软件平台规则更新速度较高的特点，试图采用混合架构来解决 TCAM 更新较慢的问题。

总的来说，可编程数据平面上的算法研究涉及的方面相当多，但最为核心且关键的是流表的实现、查找速度与更新性能，其中涉及时间与空间效率方面的均衡。毋庸置疑的是，这方面的研究仍然在未来的很长一段时间内继续活跃。与此同时，尽管硬件平台拥有相当强势的性能优势，但存在多维资源受限等缺陷，相关算法集中在提升硬件资源的有效利用率与软件协同上。相对而言，软件平台有着灵活易部署、实现周期短、资源灵活可调度的优势，并且在数据包分类算法领域存在大量的最新研究，这些研究成果也证明了软件平台在可编程数据平面领域的研究价值。此外，混合架构相对于纯硬件实现，牺牲了部分性能优势，换来了更为全面的功能面的支撑，这是未来的一个重要发展方向。

5　目前的挑战

5.1　数据平面抽象

鉴于当前 ASIC、FPGA 这样的硬件实施平台在具备高性能的同时也拥有相当程度的可编程能力，数据包处理逻辑抽象的主流仍然是更加适配硬件流水线的匹配动作管道抽象。实际上，可编程性硬件平台在数据平面方向上的研究通常伴随着匹配动作管道抽象的发展。可编程硬件平台的特点是资源受限，且灵活性较低，因此在性能、功能以及简单性之间的权衡是硬件平台上设计数据包处理逻辑抽象的重点。通用 CPU 平台的优异

可编程性使其在数据平面实现上对数据包处理逻辑抽象并不挑剔，而在一些性能指标上则各有优劣。通用 CPU 平台的一个特点则是 CPU 缓存命中率对处理效率有着显著影响。较差的并行性是该平台性能的一个关键瓶颈。

目前，数据平面语言仍然存在许多问题。首先，目前仍然没有一种数据平面语言能够提供图灵完备编程语言中的常见数据结构，这无疑提升了复杂网络功能实现的难度与平台依赖性。一旦数据平面语言确定其定义的指令集，那么该数据平面语言提供的可编程性就已经确定，能否设计更为高效且安全的指令集是数据平面语言的一个关键研究方向。其次，目前的可编程数据语言对平台无关性的实现仍在起步阶段。就 P4 而言，依然无法做到跨架构的平台移植。尽管追求高性能实现难免对部分平台产生依赖性，但与平台无关的深度可编程网络架构仍然是未来的发展方向之一。最后，数据平面语言对数据平面各种状态的支持依然有限，而这类有状态的网络功能（如带内遥测、有状态负载均衡器、有状态防火墙等）并不少见。

5.2 有状态的高性能数据平面

目前无状态数据包处理方法相当可靠，并且在硬件平台实现中能够通过多个并行的管道显著提高数据包处理速率。然而，并行处理对于有状态数据包处理是一个严峻的挑战，程序员不得不在保证数据包处理性能的同时解决一致性等状态管理问题。MP5[94]是该领域的最新研究，该研究提出了一种新型的交换机设计，保证所有数据包处理程序的功能等同于逻辑上的单一数据包处理管道（即保证了状态一致性），同时也实现了接近理想的数据包处理速率。然而，该交换机设计仍然在功能等价、性能与可扩展性上存在一定的限制。如何实现带状态的高性能数据平面仍然是目前的一个热门问题。

5.3 数据包处理性能

在数据包转发处理方面，可编程硬件平台通过多个并行管道实现线速数据包处理，例如，Intel Tofino 3 使用 8 个并行管道实现了最高 25.6 Tbit/s 的处理速率。软件平台通常依靠提高 CPU 缓存命中率、增加处理逻辑核来提升数据包处理性能。然而，CPU 核性能停滞不前、运行频率不稳定、并行性较差等原因导致通用CPU平台的数据包处理性能不足。通用 CPU 平台上基于匹配动作管道抽象的数据平面尽管同样设计了所谓的数据包处理管道，但其单一管道的实现方式实际上仍然是“run-to-complete”模式，与可编程硬件平台上真正意义上的管道相去甚远。另外，虽然目前已经有相当多的研究集中于提升软件平台数据平面的数据包处理速率，但在不同输入流量速率下的处理时延不稳定仍然是需要解决的问题。此外，如何合理地设计混合架构、如何正确地卸载网络功能以及如何高效地利用软件平台在资源与扩展性上的优势也是未来需要解决的问题。

5.4 监测管理

对数据平面进行编程意味着人们可以灵活地重塑网络行为，这为网络检测实现更好的测量精度和粒度提供了可能性。带内网络遥测（in-band network telemetry，INT）[95]与带内操作管理和维护（in-band operation administration and maintenance，IOAM）是基于可编程数据平面的带内测量主要研究方向。这些方法的基本思路是：构建遥测数据包，并在遥测数据包经过交换机时，将交换机的目标实时状态数据存储至数据包的遥测报头上。在多个分布式设备上进行测量带来的挑战多种多样，例如，选择哪些交换机进行测量、如何构建遥测数据包、是否或者何时对数据包中的测量数据进行汇总统计等。

6 结束语

高性能可编程数据平面是SDN架构的重要构成要素。可编程数据平面提供的高可编程性与易可重构性为网络功能的实现提供了极大的灵活

性，能够更加快速与便捷地为需求方提供定制的解决方案。本文从可编程数据平面的可编程性技术发展逻辑出发，从数据平面抽象、数据平面实施平台、主要算法以及目前的挑战几个方面对可编程数据平面技术进行了梳理。尽管目前可编程数据平面在多方面存在限制与尚未解决的问题，但随着越来越多新型网络需求以及越来越短的实现周期要求的出现，可编程数据平面将在未来网络架构中发挥不可代替的重要作用。

参考文献：

[1] HORPÁCSI D, LAKI S, VÖRÖS P, et al. Asynchronous extern functions in programmable software data planes[C]//Proceedings of 2019 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS). Piscataway: IEEE Press, 2019: 1-2.

[2] GAO Y, WANG Z L. A review of P4 programmable data planes for network security[J]. Mobile Information Systems, 2021, 2021: 1-24.

[3] BIFULCO R, RÉTVÁRI G. A survey on the programmable data plane: abstractions, architectures, and open problems[C]//Proceedings of 2018 IEEE 19th International Conference on High Performance Switching and Routing (HPSR). Piscataway: IEEE Press, 2019: 1-7.

[4] MASOUDI R, GHAFFARI A. Software defined networks: a survey[J]. Journal of Network and Computer Applications, 2016(67): 1-25.

[5] DORIA A, SALIM J H, HAAS R, et al. Forwarding and control element separation (ForCES) protocol specification: RFC 5810 [R]. 2010.

[6] WANG Z, TSOU T, HUANG J, et al. Analysis of comparisons between OpenFlow and ForCES: draft-wang-forces-compare-openflow-forces-01[R]. 2012.

[7] MCKEOWN N, ANDERSON T, BALAKRISHNAN H, et al. OpenFlow[J]. ACM SIGCOMM Computer Communication Review, 2008, 38(2): 69-74.

[8] KANNAN P G, CHAN M C. On programmable networking evolution[J]. CSI Transactions on ICT, 2020, 8(1): 69-76.

[9] BOSSHART P, DALY D, GIBB G, et al. P4[J]. ACM SIGCOMM Computer Communication Review, 2014, 44(3): 87-95.

[10] LI S R, HU D Y, FANG W J, et al. Protocol oblivious forwarding (POF): software-defined networking with enhanced programmability[J]. IEEE Network, 2017, 31(2): 58-66.

[11] FEAMSTER N, REXFORD J, ZEGURA E. The road to SDN: an intellectual history of programmable networks[J]. ACM SIGCOMM Computer Communication Review, 2014, 44(2): 87-98.

[12] FARHAD H, LEE H, NAKAO A. Data plane programmability in SDN[C]//Proceedings of 2014 IEEE 22nd International Conference on Network Protocols. Piscataway: IEEE Press, 2014: 583-588.

[13] ZILBERMAN N, WATTS P M, ROTSOS C, et al. Reconfigurable network systems and software-defined networking[J]. Proceedings of the IEEE, 2015, 103(7): 1102-1124.

[14] STEVENS W P, MYERS G J, CONSTANTINE L L. Structured design[J]. IBM Systems Journal, 1974, 13(2): 115-139.

[15] ABADI M, BARHAM P, CHEN J M, et al. TensorFlow: a system for large-scale machine learning[C]//Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation. New York: ACM Press, 2016: 265-283.

[16] KOHLER E, MORRIS R, CHEN B J, et al. The click modular router[J]. ACM Transactions on Computer Systems, 2000, 18(3): 263-297.

[17] BARACH D, LINGUAGLOSSA L, MARION D, et al. Batched packet processing for high-speed software data plane functions[C]//Proceedings of IEEE INFOCOM 2018 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS). Piscataway: IEEE Press, 2018: 1-2.

[18] MARTINS J, AHMED M, RAICIU C, et al. ClickOS and the art of network function virtualization[C]//Proceedings of 11th USENIX Symposium on Networked Systems Design and Implementation. Berkeley: USENIX Association, 2014: 459-473.

[19] BARBETTE T, SOLDANI C, MATHY L. Fast userspace packet processing[C]//Proceedings of 2015 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS). Piscataway: IEEE Press, 2015: 5-16.

[20] HAN S J, JANG K, PANDA A, et al. SoftNIC: a software NIC to augment hardware[R]. 2015.

[21] PANDA A, HAN S J, JANG K, et al. NetBricks: taking the V out of NFV[C]//Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation. New York: ACM Press, 2016: 203-216.

[22] HALPERN J, SALIM J H. Forwarding and control element separation (ForCES) forwarding element model[R]. 2010.

[23] RAMIREZ P L G, LLORET J, CORDERO S M, et al. Design and implementation of ForCES protocol[J]. Network Protocols and Algorithms, 2017, 9(1-2): 1-27.

[24] LAUFER R, GALLO M, PERINO D, et al. CliMB[J]. ACM SIGCOMM Computer Communication Review, 2016, 46(4): 17-22.

[25] LI B J, TAN K, LUO L, et al. ClickNP: highly flexible and high performance network processing with reconfigurable hardware[C]//Proceedings of the 2016 ACM SIGCOMM Conference. New York: ACM Press, 2016: 1-14.

[26] JING L N, CHEN X, WANG J L. Design and implementation of programmable data plane supporting multiple data types[J].

Electronics, 2021, 10(21): 2639.

[27] PFAFF B, PETTIT J, KOPONEN T, et al. The design and implementation of Open vSwitch[C]//Proceedings of 12th USENIX Symposium on Networked Systems Design and Implementation. Berkeley: USENIX Association, 2015: 117-130.

[28] SUN X Y, NG T S E, WANG G H. Software-defined flow table pipeline[C]//Proceedings of 2015 IEEE International Conference on Cloud Engineering. Piscataway: IEEE Press, 2015: 335-340.

[29] RÉTVÁRI G, MOLNÁR L, ENYEDI G, et al. Dynamic compilation and optimization of packet processing programs[Z]. 2017.

[30] 耿俊杰, 颜金尧. 基于可编程数据平面的网络体系架构综述[J]. 中国传媒大学学报(自然科学版), 2019, 26(5): 38-43.
GENG J J, YAN J Y. Overview of network architecture based on programmable data plane[J]. Journal of Communication University of China (Science and Technology), 2019, 26(5): 38-43.

[31] KOZANITIS C, HUBER J, SINGH S, et al. Leaping multiple headers in a single bound: wire-speed parsing using the kangaroo system[C]//2010 Proceedings IEEE INFOCOM. Piscataway: IEEE Press, 2010: 1-9.

[32] LU G H, SHI Y F, GUO C X, et al. CAFE: a configurable packet forwarding engine for data center networks[C]//Proceedings of the 2nd ACM SIGCOMM Workshop on Programmable Routers for Extensible Services of Tomorrow. New York: ACM Press, 2009: 25-30.

[33] 申涓, 段通, 兰巨龙. 面向全可编程网络数据平面的资源优化方法[J]. 电子学报, 2018, 46(10): 2423-2429.
SHEN J, DUAN T, LAN J L. The resource optimization towards fully-programmable network dataplane[J]. Acta Electronica Sinica, 2018, 46(10): 2423-2429.

[34] SONG H Y. Protocol-oblivious forwarding: unleash the power of SDN through a future-proof forwarding plane[C]//Proceedings of the 2nd ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking. New York: ACM Press, 2013: 127-132.

[35] YU J Z, WANG X Z, SONG J, et al. Forwarding programming in protocol-oblivious instruction set[C]//Proceedings of 2014 IEEE 22nd International Conference on Network Protocols. Piscataway: IEEE Press, 2014: 577-582.

[36] GUREVICH V. Programmable data plane at terabit speeds[Z]. 2017.

[37] 林耘森箫, 毕军, 周禹, 等. 基于 P4 的可编程数据平面研究及其应用[J]. 计算机学报, 2019, 42(11): 2539-2560.
LIN Y S X, BI J, ZHOU Y, et al. Research and applications of programmable data plane based on P4[J]. Chinese Journal of Computers, 2019, 42(11): 2539-2560.

[38] WANG H, SOULÉ R, DANG H T, et al. P4FPGA: a rapid prototyping framework for P4[C]//Proceedings of the Symposium on SDN Research. New York: ACM Press, 2017: 122-135.

[39] BUDIU M H, DODD C. The P416 programming language[J]. ACM SIGOPS Operating Systems Review, 2017, 51(1): 5-14.

[40] O'LOUGHLIN D, COFFEY A, CALLALY F, et al. Xilinx vivado high level synthesis: case studies[Z]. 2014.

[41] KAUR S, KUMAR K, AGGARWAL N. A review on P4-programmable data planes: architecture, research efforts, and future directions[J]. Computer Communications, 2021(170): 109-129.

[42] ZILBERMAN N, AUDZEVICH Y, COVINGTON G A, et al. NetFPGA SUME: toward 100 Gbps as research commodity[J]. IEEE Micro, 2014, 34(5): 32-41.

[43] CONSORTIUM P L. Behavioral model (BMv2)[EB]. 2018.

[44] KALJIC E, MARIC A, NJEMCEVIC P, et al. A survey on data plane flexibility and programmability in software-defined networking[J]. IEEE Access, 2019(7): 47804-47840.

[45] SIVARAMAN A, CHEUNG A, BUDIU M H, et al. Packet transactions: high-level programming for line-rate switches[C]//Proceedings of the 2016 ACM SIGCOMM Conference. New York: ACM Press, 2016: 15-28.

[46] DUNCAN R, JUNGCK P. packetC language for high performance packet processing[C]//Proceedings of the 2009 11th IEEE International Conference on High Performance Computing and Communications. New York: ACM Press, 2009: 450-457.

[47] BREBNER G, JIANG W R. High-speed packet processing using reconfigurable computing[J]. IEEE Micro, 2014, 34(1): 8-18.

[48] 凌致远, 陈晓, 宋磊. 基于匹配动作表模型的可编程数据平面流表归并[J]. 计算机与现代化, 2022(7): 74-78, 84.
LING Z Y, CHEN X, SONG L. Flow table merging on programmable data planes based on MAT model[J]. Computer and Modernization, 2022(7): 74-78, 84.

[49] DOBRESCU M, ARGYRAKI K, RATNASAMY S. Toward predictable performance in software packet-processing platforms[C]//Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation. New York: ACM Press, 2012: 11.

[50] SHARMA N K, KAUFMANN A, ANDERSON T, et al. Evaluating the power of flexible packet processing for network resource allocation[C]//Proceedings of the 14th USENIX Conference on Networked Systems Design and Implementation. New York: ACM Press, 2017: 67-82.

[51] WANG W M, DONG L G, ZHUGE B, et al. Design and implementation of an open programmable router compliant to IETF ForCES specifications[C]//Proceedings of Sixth International Conference on Networking (ICN'07). Piscataway: IEEE Press, 2007: 82.

[52] KATTA N, ALIPOURFARD O, REXFORD J, et al. CacheFlow: dependency-aware rule-caching for software-defined networks[C]//Proceedings of the Symposium on SDN Research. New York: ACM Press, 2016: 1-12.

[53] HAMADI S, SNAIKI I, CHERKAOUI O. Fast path accelera-

tion for open vSwitch in overlay networks[C]//Proceedings of 2014 Global Information Infrastructure and Networking Symposium (GIIS). Piscataway: IEEE Press, 2014: 1-5.

[54] HAN S J, JANG K, PARK K, et al. PacketShader: a GPU-accelerated software router[C]//Proceedings of the ACM SIGCOMM 2010 Conference. New York: ACM Press, 2010: 195-206.

[55] MOLNÁR L, PONGRÁCZ G, ENYEDI G, et al. Dataplane specialization for high-performance OpenFlow software switching[C]//Proceedings of the 2016 ACM SIGCOMM Conference. New York: ACM Press, 2016: 539-552.

[56] RIZZO L. Netmap: a novel framework for fast packet I/O[C]//Proceedings of the 2012 USENIX Conference on Annual Technical Conference. Berkeley: USENIX Association, 2012: 101-112.

[57] ZHU H Q. Data plane development kit (DPDK)[EB]. 2020

[58] FD.io - the universal dataplane[EB]. 2022.

[59] HØILAND-JØRGENSEN T, BROUER J D, BORKMANN D, et al. The eXpress data path: fast programmable packet processing in the operating system kernel[C]// Proceedings of the 14th International Conference on Emerging Networking Experiments and Technologies. New York: ACM Press, 2018: 54-66.

[60] TANG X X, ZENG X W, SONG L. A forwarding latency optimization method for software data plane based on spin-polling[J]. Applied Sciences, 2022, 12(17): 8758.

[61] SHANKAR D, LU X Y, PANDA D K D. SimdHT-bench: characterizing SIMD-aware hash table designs on emerging CPU architectures[C]//Proceedings of 2019 IEEE International Symposium on Workload Characterization (IISWC). Piscataway: IEEE Press, 2020: 178-188.

[62] LING Z Y, CHEN X, SONG L. Flow processing optimization with accelerated flow actions on high speed programmable data plane[J]. IEICE Transactions on Communications, 2023, E106.B(2): 133-144.

[63] BOSSHART P, GIBB G, KIM H S, et al. Forwarding metamorphosis: fast programmable match-action processing in hardware for SDN[C]//Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM. New York: ACM Press, 2013: 99-110.

[64] VARGA P, KOVÁCS L, TÓTHFALUSI T, et al. C-GEP: 100 Gbit/s capable, FPGA-based, reconfigurable networking equipment[C]//Proceedings of 2015 IEEE 16th International Conference on High Performance Switching and Routing (HPSR). Piscataway: IEEE Press, 2016: 1-6.

[65] FORENCICH A, SNOEREN A C, PORTER G, et al. Corundum: an open-source 100-Gbps NIC[C]//Proceedings of 2020 IEEE 28th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM). Piscataway: IEEE Press, 2020: 38-46.

[66] SHA M, GUO Z C, GUO Y F, et al. A high-performance and flexible architecture for accelerating SDN on the MPSoC platform[J]. Micromachines, 2022, 13(11): 1854.

[67] HUANG X Y, GUO Z C, SONG M G, et al. AccelSDP: a reconfigurable accelerator for software data plane based on FPGA SmartNIC[J]. Electronics, 2021, 10(16): 1927.

[68] 张昕怡, 潘恒, 谢高岗. 可编程网络数据平面技术进展[J]. 电信科学, 2022, 38(6): 42-50.
ZHANG X Y, PAN H, XIE G G. Progress in programmable network data plane[J]. Telecommunications Science, 2022, 38(6): 42-50.

[69] METREVELI Z, ZELDOVICH N, KAASHOEK M F. Cphash[J]. ACM SIGPLAN Notices, 2012, 47(8): 319-320.

[70] PAGH R, RODLER F F. Cuckoo hashing[J]. Journal of Algorithms, 2004, 51(2): 122-144.

[71] ZHOU D, FAN B, LIM H, et al. Scalable, high performance Ethernet forwarding with CuckooSwitch[C]//Proceedings of the 9th ACM Conference on Emerging Networking Experiments and Technologies. New York: ACM Press, 2013: 97-108.

[72] SHI S Q, QIAN C. Ludo hashing: compact, fast, and dynamic key-value lookups for practical network systems[J]. Proceedings of the ACM on Measurement and Analysis of Computing Systems, 2020, 4(2): 1-32.

[73] PONTARELLI S, REVIRIEGO P, MAESTRO J A. Parallel d-pipeline: a cuckoo hashing implementation for increased throughput[J]. IEEE Transactions on Computers, 2016, 65(1): 326-331.

[74] YANG Y, KUPPANNAGARI S R, SRIVASTAVA A, et al. FASTHash: FPGA-based high throughput parallel hash table[C]//Proceedings of the 2020 International Conference on High Performance Computing. Cham: Springer, 2020: 3-22.

[75] MICHEL O, BIFULCO R, RÉTVÁRI G, et al. The programmable data plane: abstractions, architectures, algorithms, and applications[J]. ACM Computing Surveys, 2022, 54(4): 1-36.

[76] 曾理, 叶晓舟, 王玲芳. DPDK 技术应用研究综述[J]. 网络新媒体技术, 2020, 9(2): 1-8.
ZENG L, YE X Z, WANG L F. Survey of research on DPDK technology application[J]. Journal of Network New Media, 2020, 9(2): 1-8.

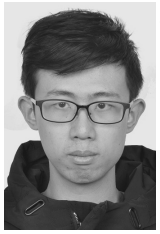
[77] IRFAN M, SANKA A I, ULLAH Z, et al. Reconfigurable content-addressable memory (CAM) on FPGAs: a tutorial and survey[J]. Future Generation Computer Systems, 2022(128): 451-465.

[78] ABDELHADI A M S, LEMIEUX G G F, SHANNON L. Modular block-RAM-based longest-prefix match ternary content-addressable memories[C]//Proceedings of 2018 28th International Conference on Field Programmable Logic and Applications (FPL). Piscataway: IEEE Press, 2018: 243-2437.

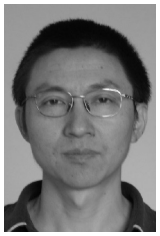
[79] IRFAN M, YANTıR H E, ULLAH Z, et al. Comp-TCAM: an adaptable composite ternary content-addressable memory on FPGAs[J]. IEEE Embedded Systems Letters, 2022, 14(2): 63-66.

[80] GUPTA P, MCKEOWN N. Packet classification on multiple fields[C]//Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication - SIGCOMM'99. New York: ACM Press, 1999: 147-160.
[81] LAKSHMAN T V, STILIADIS D. High-speed policy-based packet forwarding using efficient multi-dimensional range matching[J]. ACM SIGCOMM Computer Communication Review, 1998, 28(4): 203-214.
[82] BABOESCU F, VARGHESE G. Scalable packet classification[J]. IEEE/ACM Transactions on Networking, 2005, 13(1): 2-14.
[83] SRINIVASAN V, SURI S, VARGHESE G. Packet classification using tuple space search[J]. ACM SIGCOMM Computer Communication Review, 1999, 29(4): 135-146.
[84] YINGCHAREONTHAWORNCHAI S, DALY J, LIU A X, et al. A sorted-partitioning approach to fast and scalable dynamic packet classification[J]. IEEE/ACM Transactions on Networking, 2018, 26(4): 1907-1920.
[85] GUPTA P, MCKEOWN N. Classifying packets with hierarchical intelligent cuttings[J]. IEEE Micro, 2000, 20(1): 34-41.
[86] SINGH S, BABOESCU F, VARGHESE G, et al. Packet classification using multidimensional cutting[C]//Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications. New York: ACM Press, 2003: 213-224.
[87] VAMANAN B, VOSKUILEN G, VIJAYKUMAR T N. EffiCuts[J]. ACM SIGCOMM Computer Communication Review, 2010, 40(4): 207-218.
[88] TANG X X, ZENG X W, SONG L. Accelerating protocol oblivious forwarding programmable data plane with flow cache[J]. IEEE Transactions on Network and Service Management, 2023, 20(1): 578-594.
[89] ZHANG C, BI J, ZHOU Y, et al. B-cache: a behavior-level caching framework for the programmable data plane[C]// Proceedings of 2018 IEEE Symposium on Computers and Communications (ISCC). Piscataway: IEEE Press, 2018: 84-90.
[90] YANG J L, LI T, YAN J L, et al. PipeCache: high hit rate rule-caching scheme based on multi-stage cache tables[J]. Electronics, 2020, 9(6): 999.
[91] AHMED O, AREIBI S, FAYEK D. PCIU: an efficient packet classification algorithm with an incremental update capability[C]//Proceedings of the 2010 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS'10). Piscataway: IEEE Press, 2010: 81-88.
[92] LI Y F, WANG J L, CHEN X, et al. ITOC: an improved trie-based algorithm for online packet classification[J]. Applied Sciences, 2021, 11(18): 8693.
[93] BIFULCO R, MATSIUK A. Towards scalable SDN switches[J]. ACM SIGCOMM Computer Communication Review, 2015, 45(4): 343-344.
[94] SHRIVASTAV V. Stateful multi-pipelined programmable switches[C]//Proceedings of the ACM SIGCOMM 2022 Conference. New York: ACM Press, 2022: 663-676.
[95] KIM C, SIVARAMAN A, KATTA N, et al. In-band network telemetry via programmable dataplanes[C]//Proceedings of the 2015 ACM Conference on SIGCOMM. New York: ACM Press, 2015.

[作者简介]



唐鑫新（1995– ），男，中国科学院大学博士生，主要研究方向为可编程数据平面、协议无关交换机、高性能数据包处理。



曾学文（1968– ），男，博士，中国科学院声学研究所研究员，主要研究方向为网络安全、软件定义网络和多媒体通信。

凌致远（1997– ），男，中国科学院大学博士生，主要研究方向为软件定义网络、协议无关交换机。

宋磊（1986– ），男，博士，中国科学院声学研究所研究员，主要研究方向为可编程数据平面、信息安全。