面向多模态网络的可编程数据处理方法及系统设计

王劲林[1,2,3]，井丽南[1,2]，陈晓[1,2,3]，尤佳莉[1,2,3]

（1. 中国科学院大学电子电气与通信工程学院，北京 100049；2. 中国科学院声学研究所，北京 100190；3. 鹏城实验室，广东 深圳 518055）

摘 要：针对多模态网络面临的兼容多种路由标识、网络协议和网络应用数据处理的问题，提出了一种兼容多模态的可编程数据处理方法。通过模态识别与分流模块识别不同网络模态的数据包，利用数据类型、偏移和长度表示多种类型的数据，并利用可选择匹配算法和处理动作的匹配动作数据表组成模态数据处理管道，处理不同类型网络的数据。利用所提方法，实现并验证了一个多模态网络数据处理原型系统。实验结果表明，模态数据处理管道可以隔离不同模态的网络并保证性能独立，匹配算法的类型、处理数据的长度会影响系统转发性能，而与数据类型无关。

关键词：信息中心网络；多模态；数据面编程；数据处理

中图分类号：TP393

文献标志码：A

DOI: 10.11959/j.issn.1000−436x.2022070

Programmable data processing method and system design for polymorphic network

WANG Jinlin[1,2,3], JING Li'nan[1,2], CHEN Xiao[1,2,3], YOU Jiali[1,2,3]

1. School of Electronic, Electrical and Communication Engineering, University of Chinese Academy of Sciences, Beijing 100049, China
2. The Institute of Acoustics of the Chinese Academy of Sciences, Beijing 100190, China
3. Peng Cheng Laboratory, Shenzhen 518055, China

Abstract: Aiming at the problems of compatible with multiple route identification, network protocols and network application data processing faced by multimodal networks, a programmable data processing method compatible with multiple modes was proposed. Through the mode recognition and classify module to identify packets of different network modes, the data type, offset and length were used to represent multiple types of data, and the modal data processing pipeline was formed by using the matching action data table with selectable matching algorithm and processing action to process data of different types of network. A prototype system for data processing of polymorphic network was implemented and verified by using the proposed method. Experimental results show that the modal data processing pipeline can isolate the network of different modes and ensure the performance independence, the type of matching algorithm and the length of the processed data will affect the forwarding performance of the system, while the type of data will not.

Keywords: information-centric networking, polymorphic, data plane programming, data processing

0 引言

多模态网络是我国邬江兴院士团队为了解决当前网络结构僵化、IP 单一承载、难以抑制未知网络威胁等问题而提出的新型网络架构，它将传统的 7 层网络结构整合为功能可定义的服务面、控制面和数据面 3 个层次[1]。服务面负责智慧化的网络资源调度和业务部署，控制面针对不同网络需求灵活地定制多样化的路由策略，数据面为整个网络的全维可定义特性、多网络模态异构兼容特性提供支撑，它由可定义功能的基础平台组成。

收稿日期：2021−12−29；**修回日期**：2022−03−17

基金项目：国家重点研发计划基金资助项目（No.2020YFB1806402）

Foundation Item: The National Key Research and Development Program of China (No.2020YFB1806402)

对于支撑多模态网络的数据面，首先，其应具有灵活可编程的特性，使网络不再被僵硬封闭的设备、固定的功能模块绑定，开放网络并增强网络的可扩展和自演化能力。其次，数据面应兼容多种不同网络模态数据包的处理，支持 IP、内容标识、身份标识、地理空间标识等多种模态的路由机制。最后，数据面的网络节点还需具有一定的计算能力，支持以“内容”为中心的网络数据在数据面自适应地流动和处理，满足多样化、个性化的网络用户业务需求。

然而，若要满足多模态网络数据面的上述要求，对基础平台的数据处理能力提出了极大的挑战。一方面，不同模态的网络在数据包格式、路由机制和传输协议上存在巨大差别。例如，传统网络采用固定长度、反映地理位置的 IP 地址作为标识。路由器使用包含目的地址和下一跳端口的路由表转发数据包。信息中心网络（ICN, information-centric networking）采用与位置分离的内容名字作为标识，其标识长度不再完全固定[2]。信息中心网络中的数据包格式和路由策略也发生了变化，例如命名数据网络（NDN, named data networking）中路由器使用内容缓存表、兴趣包缓存表和转发表处理包[3]。发布订阅互联网路由模式（PSIRP, publish-subscribe Internet routing paradigm）网络中路由器甚至不再需要存储类似路由表的数据，直接根据数据包头中的链路标识即可转发包[4]。

另一方面，基础平台需要具有不局限于数据包处理的、丰富的数据处理能力，并支持控制器以可编程的方式部署数据处理逻辑，以满足不同网络的数据处理需求。除了基本的路由功能，基础平台还需要支持动态网络应用（如拥塞控制[5-7]、调度[8]、网络测量[9-11]、主动队列管理[12-14]、安全性[15]和负载平衡[16-17]等），从而满足多模态网络中多样化、个性化的用户业务需求。

综上，面对多模态网络数据面需要兼容处理多种网络数据的挑战，本文提出面向多模态网络的数据处理方法。首先，使用数据类型、偏移和长度兼容表示数据面的多种模态类型数据；其次，使用可选择匹配算法的匹配动作数据表和一组原子性动作，组合出满足不同网络模态数据包处理需求的匹配表；再次，通过包括多个匹配动作数据表和本地数据空间的模态数据处理管道形成适用于不同网络模态的数据处理单元；最后，利用模态识别与分流模块对数据包进行分类并转发到合适的管道中，利用封装转发与多播模块将管道处理后的数据包转发到物理链路上。控制面通过可编程的方式配置数据面中的匹配动作数据表、模态数据处理管道等模块，满足多模态网络灵活性、可扩展性的要求。

本文主要的贡献如下。

1) 提出一种兼容不同网络模态数据处理的数据面处理架构。使用模态识别与分流模块分类数据包，使用模态数据处理管道处理不同网络模态的数据包，使用封装转发与多播模块转发数据包，提供了控制面通信接口用于对上述模块的编程配置。

2) 提出一种使用数据类型、偏移和长度表示数据的方法。在数据面转发设备中兼容表示不同网络模态的数据，以及转发设备内不同类型的数据。所提方法可以将数据匹配和动作处理过程与数据所属的模态或类型解耦，有助于各功能模块独立优化。

3) 提出一种与网络模态无关的匹配动作数据表，以及一套原子性动作用于网络数据处理。其中匹配动作数据表不再局限于数据包字段的匹配，还可用于转发设备内其他数据的匹配。匹配动作数据表还与具体的匹配算法解耦，网络编程人员可以选择匹配算法库中的算法以达到目标功能和性能的需求。

4) 在通用服务器上，基于 Intel 的数据平面开发套件（DPDK, data plane development kit）实现面向多模态网络的数据处理原型系统，并从匹配算法、数据类型、模态数据处理管道的数量等方面对转发性能的影响进行了测试。

1　多模态网络

全维可定义的多模态智慧网络（PINet, polymorphic smart network）是一种具有多模态功能呈现、全方位覆盖、全业务承载、智慧化管理控制和内生安全特性的新型网络体系架构[1]。PINet 针对现有网络架构存在的结构僵化、IP 单一承载、难以抑制未知威胁等问题，从网络结构的角度来提升网络的功能、性能、效能、安全等性质，建立从底层到上层全维度可定义的灵活、通用的网络架构。PINet 将传统网络的 7 层参考模型整

合为：1) 网络功能基础平台组成的数据面；2) 支持 IP、内容、身份和地理空间位置等多模态标识路由的控制面；3)“感知−决策−执行”一体化管理、传输与控制闭环的服务面。

多模态网络系统形态[1]如图 1 所示。图 1 中，数据面是支撑控制面多模态路由策略和服务面网络业务的基础平台，拟合了路由与资源，数据面的物理节点中拥有状态信息（如节点位置、类型、可信度、故障率等）和资源服务能力（如算术逻辑运算、存储、传输带宽等）。多模态网络的软件定义数据面处理流程[1]如图 2 所示，包括帧解析、分组解析、适配处理、分组封装、交换调度与队列管理。软件定义的 PINet 数据面具有兼容性和可扩展性，可以支持多种类型的数据包格式、多种路由标识、网络协议栈支持标准协议和各种自定义协议，从而实现各种形态的模态共存。

然而不同网络模态在数据包格式、路由标识和路由机制等方面存在巨大差异，如表 1 所示。首先，路由标识的长度和形式不再固定，出现了类似于统一资源定位符（URL, uniform resource locator）的层次化名字、人类不可读的哈希值等多种路由标识；其次，数据包类型和路由机制也有多种形式，例如，PSIRP 网络中转发设备不再需要保留路由表，而是直接根据内容分组中的链路标识路由。

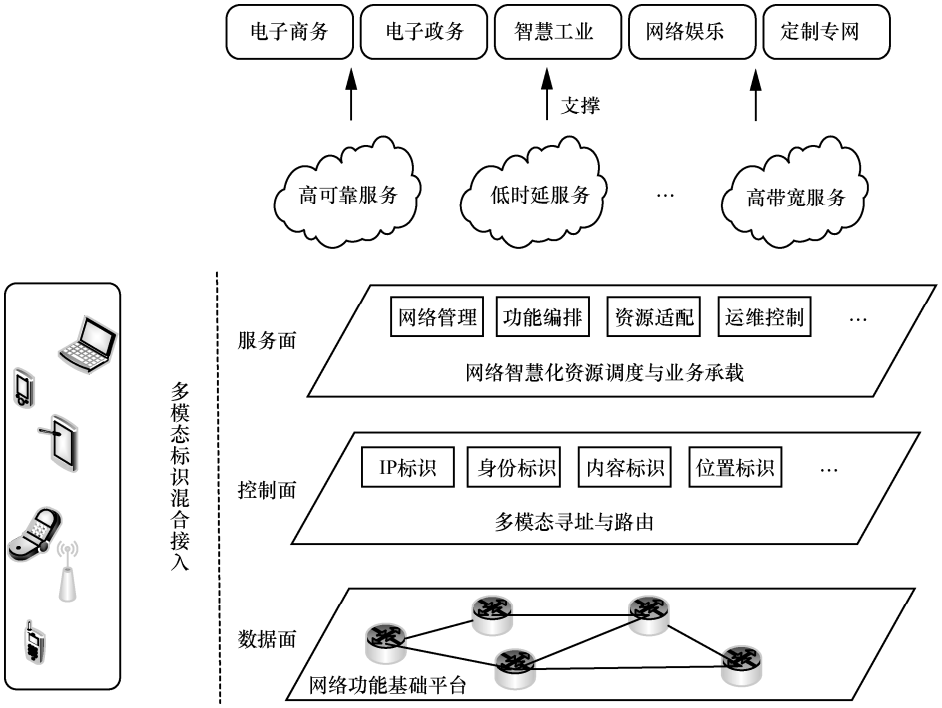

图 1　多模态网络系统形态

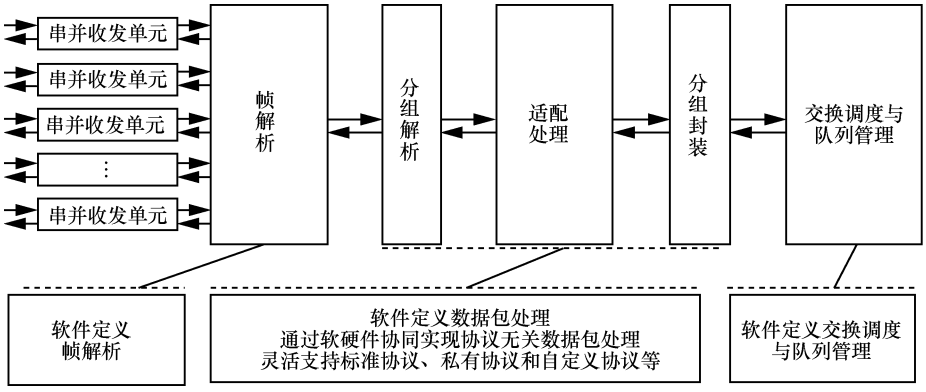

图 2　多模态网络的软件定义数据面处理流程

表 1　　**不同模态网络特性对比**

网络模态	路由标识	数据包类型	路由机制
IP[18]	IP 地址	IP 数据包	基于 IP 地址路由
NDN[3]	层次化名字	兴趣包，NDN 数据包	基于名字路由
DONA[19]	扁平化名字	注册消息，请求消息	基于名字路由
PSIRP[4,20]	扁平化名字	内容请求，事件通知	名字解析
NetInf[21]	扁平化名字	数据对象	名字解析/基于名字路由
SEANet[22]	扁平化名字（ID）和网络地址（NA）	ID/NA 协同报文，NA 报文	基于名字和网络地址的协同路由

在同一物理网络中兼容上述不同网络模态对转发设备的能力提出了挑战，针对不同网络模态增加相应的功能模块，只能暂时满足兼容性，面对未来可能出现的新型网络模态缺乏可扩展性。

2　面向多模态网络的数据处理方法

2.1　多模态网络数据面处理机制

本文提出一种面向多模态网络的数据面处理机制，可以在同一物理网络域中通过可编程的方式支持多种网络模态，如图 3 所示。其中，数据面转发节点通过模态识别与分流、模态数据处理管道、封装转发与多播以及控制面通信接口这 4 个模块相互配合完成数据包的处理。

模态识别与分流模块包括一张模态识别表和若干队列，控制面配置模态识别与分流、管道与队列的关系以及管道内部数据包处理逻辑如图 4 所示。控制器利用控制面通信接口配置修改模态识别的规则以便分离不同网络的数据包。这些规则由网络编程人员根据不同模态数据包在协议字段、进入端口等多个方面的区别制定，并利用如 P4 等网络编程语言描述。因为网络模态数据包的格式相对固定且有限，所以模态识别的规则相对固定。因此，数据面交换机可以利用可编程网卡加速模态识别，即在网卡中配置规则表，当网卡接收数据包时直接查表分类，然后通过队列进入不同的模态数据处理管道等待处理。这样可以避免利用 CPU 等通用硬件识别模态带来的时间损耗[23]。

数据面支持控制器配置管道和队列之间的关系，为多模态网络数据的处理提供足够的灵活性。例如，对于数据流量大的网络可以在识别数据包后将其轮询到多个队列中（如图 4 中的模态 3），每个队列分别连接一个管道，通过并联的多个管道提高吞吐量。对于逻辑功能复杂的网络，可以串联多个管道用于解耦功能模块。

封装转发与多播模块根据管道的处理结果将数据包重新封装（如替换包的某些字段）并转发到

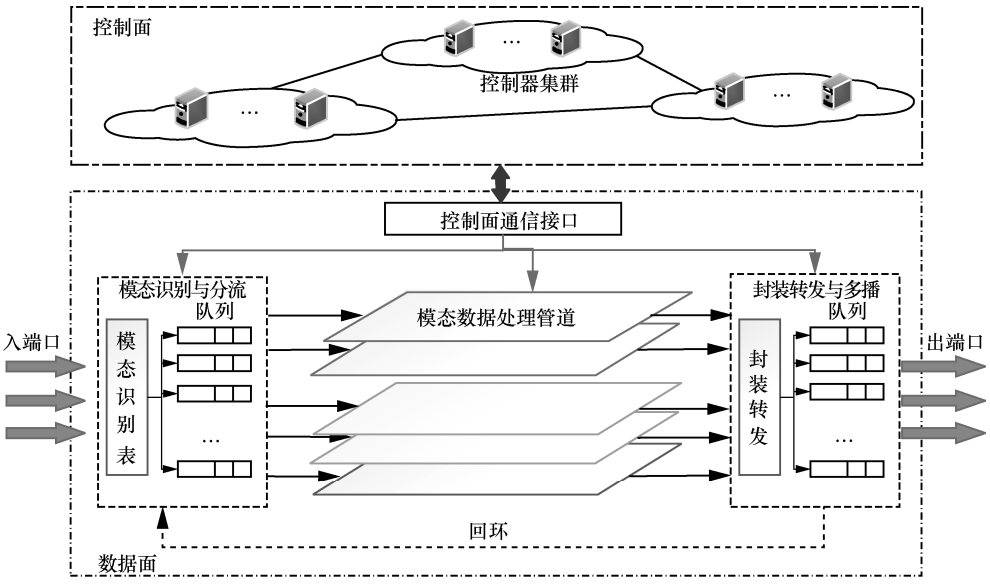

图 3　多模态网络数据面处理机制

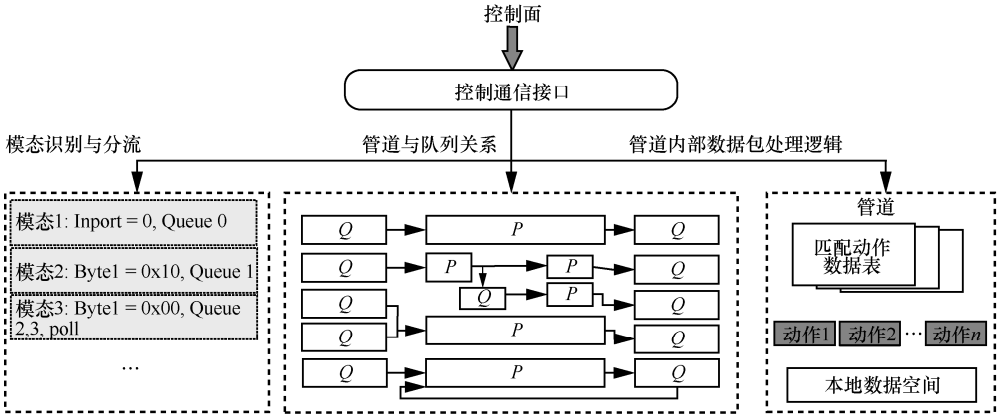

图 4　控制面配置模态识别规则、管道与队列的关系以及管道内部数据包处理逻辑

物理链路上。此模块为数据包转发方式提供了多种选择，如单播到某个端口、在多个端口组播、在全部端口泛洪，或者回环到模态识别与分流模块对数据包进行二次处理。

模态数据处理管道是网络数据包处理的基本单元，具有网络模态无关、协议无关、数据类型无关的特点，多个管道之间可以通过串联/并联的方式连接。管道仅为用户提供可以组合和编程的基本功能模块，网络运营商可以通过编程管道，将其定制为适用于不同网络模态的处理单元。支持的网络模态只需增加模态识别规则，配置相应的处理管道即可。此时，网络设备的处理能力不再局限于封闭固定的功能模块，网络设备内的资源可以得到充分的利用。本文将在 2.2 节详细介绍管道的内部结构和功能。

控制面通信接口向控制器提供了实时、可编程配置网络设备的通信接口。如图 4 所示，控制器可以利用控制面通信接口实时地编程配置模态识别规则、管道和队列连接关系、管道内部数据包处理逻辑以及数据包转发方式。控制器在配置管道 P 与队列 Q 或管道之间的连接时，需说明数据包进入和流出的方向，如图 4 中箭头所示。

本文所提的处理机制使数据面的网络设备不再与特定的网络功能捆绑，允许控制器以编程的方式灵活地配置数据面网络设备内的功能模块，根据网络需求调度网络设备内的资源。可编程的模态识别与分流模块，以及可定制功能、可连接组合的模态数据处理管道可以支撑在同一物理网络内融合多种网络模态的需求。

2.2　兼容多模态的数据处理方法

模态数据处理管道是兼容处理不同网络模态的关键模块，本节首先介绍管道的组成结构，然后通过介绍数据表示方法和匹配动作数据表说明管道如何兼容处理不同模态的数据。

2.2.1　模态数据处理管道

模态数据处理管道如图 5 所示，管道内部使用一个或多个匹配动作数据表描述数据包处理逻辑，并提供了可存储网络状态数据的本地数据空间。

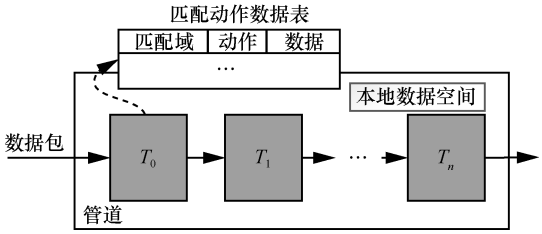

图 5　模态数据处理管道

管道提供了 2 种匹配动作数据表的配置方式：1) 由控制器初始化匹配动作数据表，并在网络运行中安装/删除规则；2) 由控制器初始化匹配动作数据表，由表内的动作在网络运行中安装/删除规则，其中动作是在控制器初始化时配置的。这 2 种方式可以同时使用，控制器可以要求数据面汇报表内的规则情况。这种灵活的转发规则配置方式具有以下优势：1) 控制器可以掌握网络全局视野；2) 控制器不需要安装全部规则，可以缓解负载压力；3) 数据面自动安装规则可以在网络事件发生时快速响应。

管道中的本地数据空间为响应网络事件、改变包转发逻辑等提供了状态信息存储空间。例如，网络应用可以在本地数据空间中记录需要跟踪的网络状态，控制器也可以将响应网络事件的消息提前存储在本地空间中。

2.2.2　数据表示方法

允许管道适应不同网络模态数据处理的关键在

于管道内使用了一种灵活的数据表示方法。具体而言，本文提出使用数据类型、偏移和长度表示网络设备中可能存在的不同网络模态数据。其中数据类型说明了数据在网络设备中的记录形式，类型可以是数据包、本地存储数据、匹配动作数据表中的数据，或者是网络设备内记录的节点属性等状态数据。偏移和长度说明了数据的相对起始位置的偏移和长度。

虽然网络模态和数据类型可能有无限多种，但是在网络设备中数据的记录形式是有限的。通过管道对不同模态数据的隔离，就可以使用有限的数据类型表示无限多种网络模态数据。如图 6 所示，管道 A 和 B 分别处理 IP 网络和 DONA 网络的数据。管道 A 中{数据包, 34, 16}表示相对 IPv6 数据包起始位置偏移 34 byte、长度 16 byte 的目的地址；管道 B 中{数据包, 56, 40}表示相对 DONA 请求消息（本文把消息和数据包划为同一类型）起始位置偏移 56 byte、长度 40 byte 的内容名。

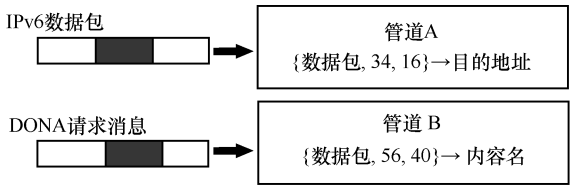

图 6　使用数据类型、偏移和长度兼容表示多种网络模态数据

这种数据表示方法与网络模态、协议类型无关，是为了便于网络编程人员描述数据处理的抽象逻辑。而且这种数据表示方法与网络设备的底层细节解耦，可以用于网络功能的跨平台快速部署，也有利于网络设备的性能独立优化。

2.2.3　匹配动作数据表

匹配动作数据表是管道中处理数据包的核心部分，本节通过介绍匹配动作数据表的 3 个组成部分，即匹配域、动作和数据，来解释其可以适应不同网络数据包处理的原因。

匹配域说明了数据包查表所需的匹配字段，匹配字段使用{数据类型,偏移,长度}表示。通过第 2.2.2 节可知，利用这种数据表示方法和模态数据处理管道，可以将无限类型的网络模态数据，转换为有限类型的数据。因此，匹配字段通过这种方法表示后，数据表可匹配的数据不再局限于某种固定的网络模态或固定类型（如数据包）。转发设备内存储的应用数据（如元数据、流状态、全局数据等）也可以参与到匹配过程中。这种方法提高了匹配的灵活性，扩大了匹配表的使用范围。

更进一步，为了适应不同网络的数据处理需求，管道将匹配动作数据表与匹配算法解耦。转发设备中的匹配算法库为匹配动作数据表提供多种匹配算法。如图 7 所示，网络编程人员可以选择匹配动作表使用的匹配算法。例如，选择最长前缀匹配（LPM, longest prefix match）算法用于查找 IP 地址；选择最长名称前缀匹配（LNPM, longest name prefix match）算法用于查找层次化内容名；选择直

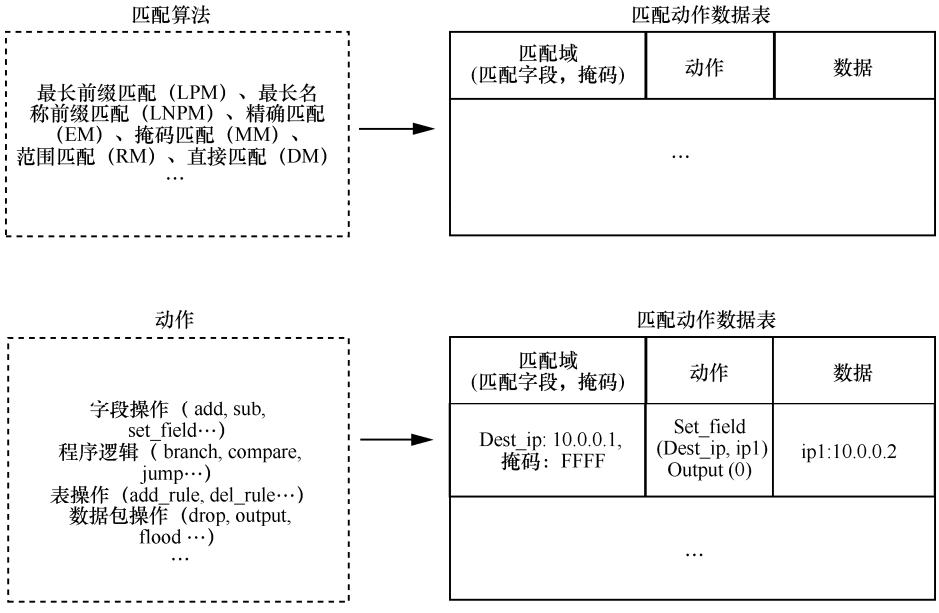

图 7　匹配动作数据表

接匹配（DM, direct match）算法，跳过提取字段匹配的环节，直接使用动作处理数据包。DM 算法适用于 PSIRP[4]、SRv6[24]等将转发操作直接记录在数据包字段内的网络。

数据包与规则匹配后，网络编程人员可以通过管道提供的具有基本功能的原子性动作组合出数据包处理逻辑。如图 7 所示，这些动作包括字段操作类，如加（add）、减（sub）、赋值（set_field）等；程序逻辑类，如分支（branch）、比较（compare）、跳转（jump）等；表操作类，如添加规则（add_rule）、删除规则（del_rule）等；以及数据包操作类，如丢弃（drop）、转发（output）、泛洪（flood）等。

值得注意的是，表操作类动作可以使网络设备在没有控制器的协助下自动增删表项。以添加规则为例，网络编程人员可以提前将所需增加的规则内容记录在本地数据空间中，利用字段操作类动作根据网络状态实时修改已记录的规则内容，最后使用 add_rule 动作将规则添加到某个表中。删除规则更加简单，只需要描述规则所在表和它的匹配值即可。这种自动增删规则的方式不仅缓解了控制器的压力，还可以及时响应网络事件。通过每隔一段时间向控制器同步表的状态，保证控制器仍然了解网络全局状态。

匹配动作数据表中的数据字段为网络应用提供了记录流状态的空间。匹配动作数据表中每个表项不仅包括匹配域和动作，还包括数据字段。网络编程人员可以利用字段操作类动作读写数据字段中的内容。数据字段还可以用于记录动作参数，避免因动作参数的变化，导致表项频繁更新。以图 7 匹配动作数据表中的规则为例，网络编程人员可以将动作中使用的目的地址（10.0.0.2）记录在表项的数据字段中，当替换地址发生变化时，不需要修改动作，只需要替换数据字段的内容即可。而数据字段的内容可以通过编程的方式，利用比较和赋值动作根据网络状态在转发设备内实时更新。

综上，本文提出的匹配动作数据表具有如下优点。1) 扩大了匹配范围，匹配动作数据表不局限于处理固定网络模态或固定类型的数据。匹配动作数据表可以应用于多种网络模态数据处理，转发设备本地的数据也可以参与到匹配过程中，增加了匹配的灵活性。2) 匹配动作数据表与匹配算法解耦，匹配算法可以被独立优化，网络编程人员可以通过更新匹配算法库，将最新的匹配算法应用于数据包处理中，以满足新型网络应用的需求，通过组合多种不同匹配算法表示复杂的匹配逻辑。3) 原子性动作可以兼容多类型数据处理，表操作类动作为网络应用提供了在数据面操作表项的能力，缓解了控制器的压力，提高了网络事件响应速度。4) 匹配动作数据表中的数据字段为网络应用提供了记录流状态的空间，还可避免因动作参数变化导致的表项频繁更新的问题。

3 实验评估

3.1 实现

为了验证本文提出的面向多模态网络的数据处理方法，基于 Intel 的 DPDK[25]框架，本文实现了数据面处理原型系统。系统中的模态识别功能通过支持数据包入端口匹配和掩码匹配的查找表实现。系统中的模态识别、管道、队列等配置可通过 POF[26]南向接口由 POX[27 控制器配置，也可以利用配置文件在本地配置。管道内的匹配动作数据表可使用 4 种匹配算法，即精确匹配（EM, exact match）、LPM、掩码匹配（MM, mask match）和直接匹配（DM, direct match），可使用的动作如表 2 所示。

表 2 管道支持的原子性动作

分类	动作
字段操作	set_field, insert_field, del_field, calculate_checksum, add, sub, srl, sll, and, or, xor, nor, not
数据包操作	goto_table, output, flood, drop
程序逻辑	jump, compare, branch
表操作	add_rule, del_rule

3.2 实验

本节实验从不同匹配算法的性能对比、不同数据类型对性能的影响、系统可支持的模态数据处理管道的数量和性能等方面对实现的原型系统进行测试。实验中对管道等模块使用配置文件进行本地配置，测试系统运行的服务器为 DELL R740，服务器使用的处理器为 Intel Xeon CPU E7-4809 v4 @2.10 GHz，支持的缓存为 32 KB 的 L1i 和 L1d、256 KB 的 L2、20 MB 的 L3，使用的内存为 128 GB 的 DDR3@1 333 MHz 4 Channels，使用的网卡为 Intel XL710 PCI-E 3.0x8 40 GB 和 Intel I350 PCI-E 3.0x8 4 GB，使用的 DPDK 版本为 19.11，使用 Sprient TestCenter 产生测试数据包。

实验 1　不同匹配算法的性能对比

匹配算法可以让匹配动作数据表灵活地适应不同类型数据包的查找需求，例如 LPM[28]算法可以用于 IP 地址的路由查找；EM[29]算法可以用于扁平化哈希值的查找；利用了字典树算法的 MM[30]算法可以用于范围查找和层次化名字的查找；本文所提 DM 算法跳过了匹配过程，直接执行动作处理数据包，适用于 PSIRP、SRv6 等将路由标识记录在数据包头的网络。本文实现了上述 4 种匹配算法，实验测试不同的匹配算法对管道转发性能的影响。

通过本文提出的数据表示方法，匹配算法可以与具体的网络模态解耦，不同模态的数据包可以复用相同的匹配算法。例如 DONA 数据包和 SEANet 数据包都使用扁平化的名字，它们都可以使用 EM 算法匹配名字。SEANet 数据包作为 ID/NA 协同数据包，数据包中还会携带网络地址。那么可以使用 MM 算法匹配网络地址。本文提出匹配算法库的思想，允许网络编程人员根据需求组合不同的匹配算法以满足需求。

如图 8 所示，不同的匹配算法在功能和性能上存在差异。此实验中使用一个 CPU 逻辑核，处理一个 10 Gbit 网口的流量，CPU 的单个逻辑核运行含有一张匹配动作数据表的管道，并分别：1) 使用 EM 算法处理名字为 128 bit、98 byte 长的 DONA 数据包；2) 使用 MM 算法处理名字为 128 bit（例如，/sort/videos/movies/test.mp4/version1/....）、98 byte 长的 NDN 数据包；3) 使用 LPM 算法处理名字为 128 bit、98 byte 长的 IPv6 数据包。实验还测试了在匹配动作数据表中安装 500～10 000 等不同数量规则对转发性能的影响。

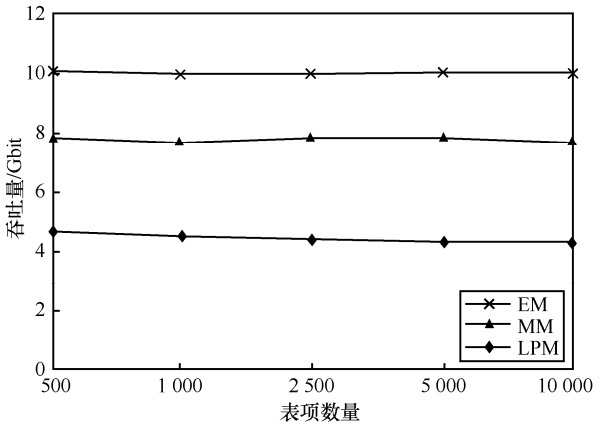

图 8　匹配动作数据表使用不同匹配算法对转发性能的影响

实验结果表明，基于哈希的 EM 算法性能更好且实现了线速转发，掩码匹配的 MM 算法次之，最长前缀 LPM 算法性能最差。在本文实验中使用一个 CPU 逻辑核运行一个模态数据处理管道，在实验 5 中本文扩展使用多个 CPU 逻辑核运行多个管道，测试转发性能是否具有可扩展性。

网络数据包的处理常需要匹配多个字段，选择合适的匹配方式可以有效地提高性能。例如，实验中使用一张 MM 表（10 万条表项）匹配 IPv6 五元组处理 98 byte 长度的数据包（一个 CPU 逻辑核运行管道），转发性能为 7.822 Gbit。转发同样的数据包，使用一张 EM 表（100 条表项）匹配协议字段，使用一张 MM 表（5 万条表项）匹配 IPv6 的源目的地址、端口，转发性能提升为 8.240 Gbit。使用 DM 表直接用表 2 中的分支比较动作匹配 10 个常用的协议字段，再使用一张 MM 表（5 万条表项）匹配 IPv6 的源目的地址、端口，转发性能提升为 8.664 Gbit。

通过不同匹配算法性能对比的实验可知，仅通过调整使用的匹配策略就可以提高转发性能。由于 IPv6 五元组中的协议字段值固定且有限，因此相比于 MM 算法，使用 EM 算法可以获得更高的性能。而 DM 表直接省略了匹配过程，利用指令比较有限的字段，避免了因读表带来的开销，可以进一步提升性能。

综上，本文提出的数据表示方法使匹配算法与具体的网络模态解耦，实现匹配算法在不同模态间的复用。而匹配算法库的思想允许网络编程人员根据需求选择合适的匹配算法进行组合，以达到最佳性能。在未来的工作中，本文将使匹配算法插件化，从而使匹配算法库快速地支持新的匹配算法。

实验 2　动作性能

动作负责在数据包与规则匹配后处理数据包，管道原子性动作的性能如表 3 所示，其中，f 为字段，imm 为立即数。本文在实验平台上测试了表 2 中各动作执行所需的 CPU 时钟周期。实验还对比了处理不同类型的数据对动作性能的影响，例如 sf (f, imm_64) 为使用 64 bit 的数据对一个数据包首部字段赋值，而 sf (f, f)则是用一个数据包首部字段对另一个相同长度的数据包首部字段赋值。为控制变量，实验中相同的动作使用的测试数据长度相同。实验中记录每个动作指令 1 000 000 次所需的 CPU 时钟周期，然后计算单次动作执行所需时间开销。

表 3 管道原子性动作的性能

动作	字段类型	CPU 时钟周期/个	动作	字段类型	CPU 时钟周期/个	动作	字段类型	CPU 时钟周期/个
sf	f, imm_64	7	srl	f, f	18	cmp	f, imm_32	10
sf	f, f	13	sll	f, imm_64	16	cmp	f, f	16
ins	f, imm_64	12	sll	f, f	12	jump	imm_32	8
ins	f, f	21	and	f, imm_64	10	br	f, imm_32	10
del	f	5	and	f, f	16	br	f, f	16
cksum	f, f	67	or	f, imm_64	10	gte	f, imm_32,imm_32	44
add	f, imm_64	8	or	f, f	16	gte	f,f,f	56
add	f, f	17	xor	f, imm_64	10	gte	f, f,imm_32	50
sub	f, imm_64	14	xor	f, f	16	ate	f, imm_32	758
sub	f, f	10	nor	f, imm_64	10	ate	f, f	764
srl	f, imm_64	12	nor	f, f	16			

由表 3 可知，动作中 2 个参数都为字段类型时所需时间更长，因为增加了一个字段的加载时间。根据表 3 中的数据，执行 sf（f, imm_64）需要 7 个 CPU 时钟周期，执行 sf（f, f）需要 13 个 CPU 时钟周期，可计算出加载一个字段所需时间为 6 个 CPU 时钟周期（在本文实验平台上约为 3 ns）。在所有动作中，增加表项动作所需的时间最长，约为 382 ns。进一步进行测试，利用 POX 控制器每秒可以向管道添加 8 624 条表项，即每条表项添加时间为 0.115 ms。显然，如果表项内容不需要控制器协助即可获得，那么使用动作添加表项所需的时间更短，可用于在数据面有动态处理表项需求的网络模态（如 NDN、SEANet 等）。

实验 3 不同数据类型对性能的影响

为了适应不同模态的数据包，本文提出使用数据类型、偏移和长度表示网络设备内的数据。用网络设备内有限的数据类型（立即数、数据包字段、元数据、表项数据和全局数据）代替可能无限多种不同模态的数据包类型。

在实验 2 中，测试动作使用的字段全部为数据包首部字段，但是本文提出的动作适用的数据类型并不局限于数据包字段，这为不同模态数据处理提供了灵活性。实验 3 测试数据类型是否会对动作性能产生影响。实验选用 set_field(p1,p2)动作进行测试，测试中 set_field 动作的 p1 参数始终为 64 bit 的数据包首部字段，变换 p2 参数分别为 64 bit 的立即数、数据包首部字段、元数据、表项数据、全局数据。在实验平台上不同参数的 set_field 动作分别执行 1 000 000 次，计算平均每次执行所需的 CPU 时钟周期数。

动作处理不同类型数据所需的 CPU 时钟周期如图 9 所示，动作中使用数据包字段、元数据、表项数据和全局数据的表项转发数据包的时延比使用立即数的高。原因是这些动作需要额外加载一次数据，而立即数可以直接使用。但是，加载不同类型数据（如数据包字段、元数据、表项数据、全局数据），数据包的转发时延是相同的。因为在所有动作执行前，管道会统一获取所有类型数据的基址，然后利用偏移和长度获取数据，所以数据类型不会对性能产生影响。

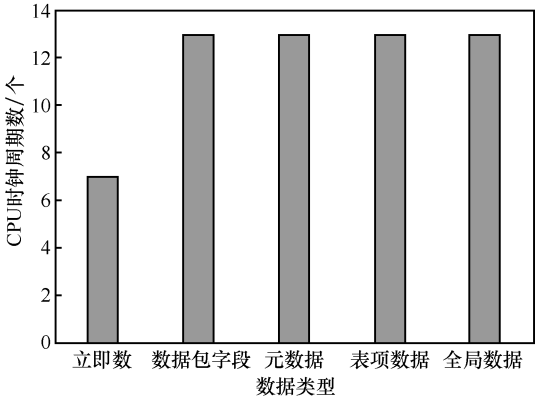

图 9 动作处理不同类型数据所需的 CPU 时钟周期

使用数据类型、偏移和长度表示网络设备中的数据，一方面，可以屏蔽不同网络模态数据包的差异，将它们统一认作数据包类型，由网络编程人员根据各模态数据包的区别，填写不同匹配动作数据表所需匹配和处理的字段位置；另一方面，在不降低动作性能的情况下，扩展了动作的能力，使动作不再局限于数据包字段处理，提高了数据面的编程能力。

实验 4　数据长度对动作性能的影响

不同模态的数据包所需处理的数据长度可能不同，如传统网络中存在 32 bit 和 128 bit 这 2 种地址长度，使用层次化命名的网络（如 NDN、CCN）其名字长度则更加多变。数据包处理中常需要进行名字替换、字段修改等操作。此实验测试不同长度的字段对动作性能产生的影响。

实验 4 中使用 set_field(p1,p2)动作测试，p1、p2 为长度相同的数据包字段。实验中测试了 set_field 动作处理 16～192 bit 的数据，包括字节对齐的数据和非对齐的数据。实验中对不同参数长度的 set_field 动作分别执行 1 000 000 次，计算平均每次执行所需的时间。

动作处理数据的长度对性能的影响如图 10 所示。图 10 中，符号<表示非对齐数据，例如，<32 表示长度在 16 bit 和 32 bit 之间的非对齐数据。从图 10 可以看出，1) 在没有对数据加载功能优化前，动作的执行时间随着处理数据的长度增加，在数据长度超过 64 bit 时增加明显；2) 处理 64 bit 的数据所需时间最短，这是由于实验平台处理器的总线位宽为 64 bit，相比于加载其他长度的数据速度更快；3) 动作处理非字节对齐的数据所需时间更长；4) 在使用因特网数据流单指令序列扩展技术对数据加载功能优化后，动作执行时间减少，在数据长度超过 64 bit 时，性能提升明显。

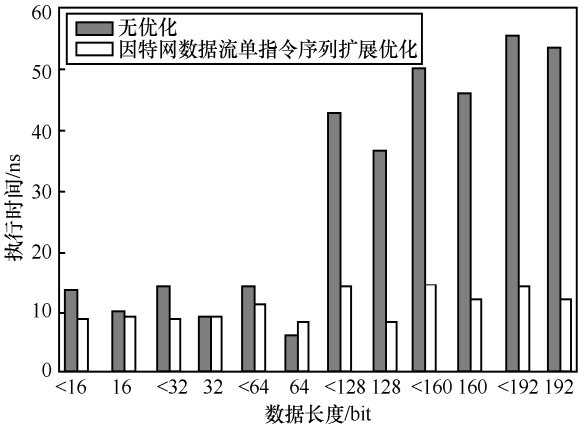

图 10　动作处理数据的长度对性能的影响

综上，在本文的实验平台上，数据长度对动作性能有明显的影响。网络编程人员在编写网络数据处理功能时可以据此选择合适的数据长度，例如，当需要在网络设备内记录临时数据时，长度为 64 bit 的数据在未来的动作处理中性能更好。

实验 5　模态数据处理管道性能测试

本文提出的面向多模态网络的数据处理方法中，模态数据处理管道是适应不同网络模态的关键部分。网络编程人员可以使用多个管道处理不同网络的数据，也可以通过多个管道提高单个网络的吞吐。

实验 5 测试同时运行多个管道对转发性能的影响。实验中每个管道都使用一个 CPU 逻辑核运行，每个管道内部都安装一张使用 LPM 算法的匹配数据动作表，匹配数据动作表中匹配 IPv6 数据包五元组。实验中使用 Sprient TestCenter 产生 98 byte、40 Gbit 的 IPv6 数据包进行测试。

运行管道数量对转发性能的影响如图 11 所示。从图 11 可以看出，当运行 1～8 个管道时，吞吐量是线性增长的。这说明各管道的性能平均，没有受到其他管道的影响。但是当运行的管道数量从 8 增加到 16 时，吞吐量增长幅度较小，没有达到 40 Gbit 的吞吐量。这是因为实验平台使用的 PCI-E 3.0x8 无法满足 40 Gbit 数据的双向转发，其 PCI-E 的极限性能为 32 Gbit。

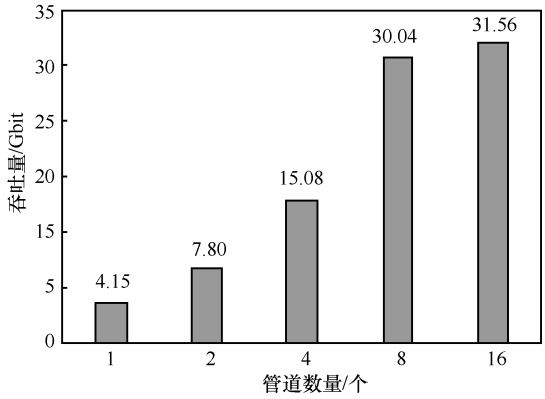

图 11　运行管道数量对转发性能的影响

实验 5 说明，通过管道隔离不同模态的网络，并保证性能独立不受其他网络的影响是可行的。但是需要注意网络设备的其他部分是否存在性能瓶颈。

4　结束语

针对多模态网络在数据面中面临的兼容多种路由标识、网络协议和网络应用数据处理的问题，本文提出了一种兼容多种模态的可编程数据处理方法。该方法利用模态识别与分流模块、模态数据处理管道、封装转发与多播模块和控制面通信接口 4 个部分，使数据面可以通过编程的方式完

成数据包的网络模态识别、不同模态数据包的定制化处理、灵活的数据包封装与转发。其中，模态数据处理管道可以适应不同网络数据处理的关键在于，本文使用数据类型、偏移和长度兼容表示多种类型数据，并使用可以选择匹配算法、利用原子性动作处理数据、支持记录流状态的匹配动作数据表，组成管道的数据包处理逻辑。本文在通用服务器上基于Intel的DPDK框架实现了一个多模态网络数据处理原型系统，并从匹配算法、数据类型、执行的动作、动作处理数据的长度和多管道并行对系统性能的影响等方面进行了实验。实验结果表明，模态数据处理管道可以隔离不同模态的网络且性能独立，匹配算法的类型、处理数据的长度会影响系统转发性能，而不同的数据类型不会对转发性能产生影响。未来的工作将在可编程硬件（如 FPGA）上实现本文所提出的模态识别与分类模块，以及以插件的方式在匹配算法库中扩展匹配算法。

参考文献：

[1] 胡宇翔, 伊鹏, 孙鹏浩, 等. 全维可定义的多模态智慧网络体系研究[J]. 通信学报, 2019, 40(8): 1-12.
HU Y X, YI P, SUN P H, et al. Research on the full-dimensional defined polymorphic smart network[J]. Journal on Communications, 2019, 40(8): 1-12.

[2] AHLGREN B, DANNEWITZ C, IMBRENDA C, et al. A survey of information-centric networking[J]. IEEE Communications Magazine, 2012, 50(7): 26-36.

[3] ZHANG L X, ESTRIN D, BURKE J, et al. Named data networking (NDN) project NDN-0001[R]. 2010.

[4] TARKOMA S, AIN M, VISALA K. The publish/subscribe Internet routing paradigm (PSIRP): designing the future Internet architecture[J]. Towards the Future Internet: A European Research Perspective, 2009: doi.org/10.3233/978-1-60750-007-0-102.

[5] TAI C H, ZHU J, DUKKIPATI N. Making large scale deployment of RCP practical for real networks[C]//Proceedings of IEEE INFOCOM 2008 - The 27th Conference on Computer Communications. Piscataway: IEEE Press, 2008: 2180-2188.

[6] HONG C Y, CAESAR M, GODFREY P B. Finishing flows quickly with preemptive scheduling[J]. ACM SIGCOMM Computer Communication Review, 2012, 42(4): 127-138.

[7] ALIZADEH M, GREENBERG A, MALTZ D A, et al. Data center TCP (DCTCP)[C]//Proceedings of the ACM SIGCOMM 2010 Conference on SIGCOMM - SIGCOMM'10. New York: ACM Press, 2010: 63-74.

[8] SIVARAMAN A, SUBRAMANIAN S, AGRAWAL A, et al. Towards programmable packet scheduling[C]//Proceedings of the 14th ACM Workshop on Hot Topics in Networks. New York: ACM Press, 2015: 1-7.

[9] YU M, JOSE L, MIAO R. Software defined traffic measurement with OpenSketch[C]//Proceedings of the 10th USENIX Symposium on Networked Systems Design and Implementation. Berkeley: USENIX Association, 2013: 29-42.

[10] ESTAN C, VARGHESE G. New directions in traffic measurement and accounting[J]. ACM Transactions on Computer Systems, 2003, 21(3): 270-313.

[11] ESTAN C, VARGHESE G, FISK M. Bitmap algorithms for counting active flows on high speed links[C]//Proceedings of the 3rd ACM SIGCOMM Conference on Internet Measurement. New York: ACM Press, 2003: 153-166.

[12] FENG W C, SHIN K G, KANDLUR D D, et al. The BLUE active queue management algorithms[J]. ACM Transactions on Networking, 2002, 10(4): 513-528.

[13] FLOYD S, JACOBSON V. Random early detection gateways for congestion avoidance[J]. IEEE/ACM Transactions on Networking, 1993, 1(4): 397-413.

[14] KUNNIYUR S S, SRIKANT R. An adaptive virtual queue (AVQ) algorithm for active queue management[J]. IEEE/ACM Transactions on Networking, 2004, 12(2): 286-299.

[15] BILGE L, KIRDA E, KRUEGEL C, et al. EXPOSURE: finding malicious domains using passive DNS analysis[C]//Proceedings of the 2011 Network and Distributed System Security Symposium. Piscataway: IEEE Press, 2011: 1-17.

[16] ALIZADEH M, EDSALL T, DHARMAPURIKAR S, et al. CONGA: distributed congestion-aware load balancing for datacenters[J]. ACM SIGCOMM Computer Communication Review, 2014, 44(4): 503-514.

[17] BARBETTE T, CHEB T, YAO H R, et al. A high-speed load-balancer design with guaranteed per-connection-consistency[C]//Proceedings of the 17th USENIX Conference on Networked Systems Design and Implementation. Berkeley: USENIX Association, 2020: 667-684.

[18] POSTEL J. Internet protocol-DARPA Internet program protocol specification[R]. 1981.

[19] KOPONEN T, CHAWLA M, CHUN B G, et al. A data-oriented (and beyond) network architecture[J]. ACM SIGCOMM Computer Communication Review, 2007, 37(4): 181-192.

[20] FOTIOU N, NIKANDER P, TROSSEN D, et al. Developing information networking further: from PSIRP to PURSUIT[C]//Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering. Berlin: Springer, 2012: 1-13.

[21] DANNEWITZ C, KUTSCHER D, OHLMAN B, et al. Network of Information (NetInf) - an information-centric networking architecture[J]. Computer Communications, 2013, 36(7): 721-735.

[22] YOU J L, QIAO N N, WANG J L, et al. An on-site elastic autonomous service network with efficient task assignment[C]//Proceedings of 2016 IEEE 41st Conference on Local Computer Networks Workshops. Piscataway: IEEE Press, 2016: 42-49.

[23] WATANABE Y, KOBAYASHI Y, TAKENAKA T, et al. Accelerating NFV application using CPU-FPGA tightly coupled architecture[C]//Proceedings of 2017 International Conference on Field Programmable Technology (ICFPT). Piscataway: IEEE Press, 2017: 136-143.

[24] FILSFILS C, CAMARILLO P, LEDDY J, et al. SRv6 network programming[R]. 2017.

[25] PONGRÁCZ G, MOLNÁR L, KIS Z L. Removing roadblocks from SDN: OpenFlow software switch performance on Intel DPDK[C]//Proceedings of 2013 Second European Workshop on Software Defined Networks. Piscataway: IEEE Press, 2013: 62-67.

[26] 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 - HotSDN'13. New York: ACM Press, 2013: 127-132.
[27] KAUR S, SINGH J, NAVTEJ S G. Network programmability using POX controller[C]//Proceedings of the International Conference on Communiction, Computing and Systems (ICCCS 2014). Piscataway: IEEE Press, 2014: 134-138.
[28] KOBAYASHI M, MURASE T, KURIYAMA A. A longest prefix match search engine for multi-gigabit IP processing[C]//Proceedings of 2000 IEEE International Conference on Communications. Piscataway: IEEE Press, 2000: 1360-1364.
[29] PAGH R, RODLER F F. Cuckoo hashing[J]. Journal of Algorithms, 2004, 51: 122-144.
[30] 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.

[作者简介]

王劲林（1964− ），男，北京人，中国科学院声学研究所研究员、博士生导师，主要研究方向为新型网络技术、分布式系统等。

井丽南（1995− ），女，吉林松原人，中国科学院声学研究所博士生，主要研究方向为可编程数据面、状态数据面等。

陈晓（1964− ），男，北京人，中国科学院声学研究所研究员、博士生导师，主要研究方向为高性能网络、数据面编程等。

尤佳莉（1982− ），女，吉林白山人，中国科学院声学研究所研究员、硕士生导师，主要研究方向为新型网络技术、分布式智能等。