西安交通大学

博士学位论文

网络数据平面可编程硬件的研究

学位申请人：乔思祎

指导教师：邹建华 教授

合作导师：邹建华 教授

学科名称：控制科学与工程

2020 年 9 月

Research on Programmable Hardware for Network Data Plane

A dissertation submitted to

Xi'an Jiaotong University

in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

By

Siyi Qiao

Supervisor: Prof. Jianhua Zou

Associate Supervisor: Prof. Jianhua Zou

Automation Science and Engineering

September 2020

博士学位论文答辩委员会

网络数据平面可编程硬件的研究

答辩人：乔思祎

答辩委员会委员：

西安交通大学教授：嗷嗷 (主席)

西安交通大学教授：宝宝

西安交通大学教授：纯粹

西安交通大学教授：蛋蛋

西安交通大学教授：尔尔

答辩时间：2020 年 12 月 34 日

答辩地点：地点

摘 要

网络通信是支撑当今社会的重要基础设施，当前的发展方向主要集中于建设高性能、高可创新性的网络环境。最近 10 年，软件定义网络 (SDN) 和可编程网络 (SDN2.0) 概念的提出很好的解决了过去网络创新难度大的不足。但随着流量和网络功能复杂度的快速增长，这种新的网络体系结构也带来了性能和鲁棒性两方面的挑战。性能和功能方面: 基于 CPU 的转发平台性能发展逐步减慢，基于 ASIC 的智能网卡硬件可编程性差。鲁棒性方面: 数据平面和控制平面分离的 SDN 网络架构带来了稳定性不足和安全性、效率低的问题。

将问题从网络的三个维度进行分析：1）主机侧网络，在服务器网卡层面，基于 CPU 的智能能网卡的性能难以满足目前虚拟化技术和网络监管细粒度化的发展需求。2）交换侧网络，在核心网骨干网层面，基于 ASIC 的转发平面不足以提供网络网络处理的高灵活性，由于其与成本、性能之间平衡困难，网络工程师的创新空间受到了限制。3）控制层-数据层交互，硬件流表是一种高效且昂贵的实现网络转发抽象的核心部件，在软件定义网络时代流表稀缺性更加突出。由于流数目和流量的快速增长，控制平面针对流表的操作导致数据平面和控制平面的大量协议开销，导致网络鲁棒性差，易形成安全隐患。近年来，现场可编程门阵列（FPGA）器件得到快速发展，以可编程硬件技术为首的异构架构已经大量融合到网络领域，带来高用户可定制能力的同时也能保证了一定的处理性能，这也为此论文的研究内容得到了基础的保障。

本文主要探索基于可编程硬件的高性能网络数据平面。本文研究在软件定义的网络编程框架内如何将这种可编程硬件抽象层融入整体系统，并设计与其配套的控制平面软件和协议，使整体网络系统的软硬件有机结合，在增强网络处理性能、灵活性的同时保证安全性和效率。论文从理论抽象分析中提出了体系架构，最后给出了系统实现并进行验证。本文将从以下三方面阐述：

1）研究可编程设备加速主机侧网络方法。本文提出利用基于 FPGA 的智能网卡卸载操作系统层部分网络功能，以达到扩展网络接入层的性能的目的。探讨了不同场景下网络功能的构成，分析并提出一种基于可编程硬件的网络功能定义模型（Data-Computing，DC 抽象）。本文把服务器网络功能任务中可转化为 DC 抽象的计算密集型功能通过合理转换下放到网卡的 FPGA 可编程器件中。论文基于可编程网卡设计了一套网络流量捕获，统计分析和回放系统。在满足网络功能不受改变的前提下，证明利用基于 FPGA 的智能网卡能有效地提升服务器的网络性能（100x）、抖动 (降低 10^4x) 和效率 (10x)。

2）研究可编程设备加速网络硬件交换层方法。本文提出一种硬件异构型的可编程网络数据平面架构，将 FPGA 与 ASIC 交换芯片有机结合，以增强 ASIC 报文处理报文的灵活性，同时满足性能需求。论文设计了 ASIC 面向硬件可编程扩展的接口，将数据包头拆分并通过高速数据互联载体发送给 FPGA，利用 FPGA 可重配特性实现完全可

编程的报文处理数据平面；同时，本文基于 DC 抽象，将网络随路计算（network-centric computing）模式引入可编程网络体系架构；本文通过分析流量模型在 FPGA 中设计了一种并行化处理单元，在资源消耗可控的前提下大规模提高系统的可扩展性能；另外本文提出了一套基于可编程硬件混合网络架构的软件定义语言编程框架，实现了软件定义需求和可编程硬抽象层分离，以及针对底层数据平面的一种高效自适应的并行单元流分配算法，在可编程性与 FPGA 同等的条件下，比目前 FPGA 交换机性能提升 120x。

3）SDN 硬件流表可扩展性研究。本文针对不同层面网络设备的控制，进行全局优化、分布式优化。在可编程网卡和交换机组成的网络系统中，数据平面内最重要的资源是流表资源（瓶颈资源），本文从全局视野角度，结合可编程硬件的特性，在全网约束的条件下，对流表资源进行优化，以满足未来可扩展性需求。本文分析不同的流量规模和特征，以及系统多模块直接独特的互联协议，提出一种 SDN 网络流表空间全局共享机制。实现了在流量大规模扩展的情形下，保证数据平面稳定性，对受影响的流转发 RTT 时间和安全通道消息风暴数量的优化均达到至少 2 个数量级。

此外，为支持本文提出的相关设计概念，本文实现了一套基于 FPGA 的转发平面设备，包括智能网卡和交换机原型平台。此套平台资源容量大，外设接口丰富，可以满足本文在各类网络架构下实验验证需求。

关 键 词：软件定义网络；网络数据平面；可编程硬件；现场可编程门阵列

论文类型：应用基础

ABSTRACT

英文摘要正文每段开头不缩进，每段之间空一行。

The abstract goes here.

LaTeX is a typesetting system that is very suitable for producing scientific and mathematical documents of high typographical quality.

KEY WORDS: Xi'an Jiaotong University, Doctoral dissertation, LaTeX template

TYPE OF DISSERTATION: Application Fundamentals

目　录

CONTENTS

1 绪论

1.1 研究的背景

1.1.1 研究的意义

21 世纪的新 20 年，网络正以前所未有的速度越来越紧密地参与到民生社会中，对满足国家民生需求、新基建拉动内需和产业升级起到了至关重要的作用。从“百度一下”到网红全民直播带货，从实现“三网通”到发展“新基建”的国家战略，小到优化社会资源效率的办公数字化，大到勾勒出智能交通、智慧城市和万物互联的 5G 海洋，无一不是构建在网络基础设施的快速发展之上。思科公司预计，到 2023 年全球家用互联网总带宽将达到 $5.85Ebps$[①]（是现在的 3.27 倍），移动互联网用户预计达到 57 亿，其总流量可达 $11.3Ebps$ (将达到目前的 5 倍)，其中 5G 流量将占据移动互联网总带宽的 76.5%（0.6%，2019 年）[1-2]。由于深度学习、AI、大数据、云计算、物联网的快速发展，这些新技术将催使新零售、新金融、新医疗、新教育、新制造、云视频和云游戏等行业“云化”，海量的数据会在数据中心内部服务器间网络中及对外网关中交互，这些关键应用将会改变数据中心算力和数据中心内部网络结构特性。

IDC 报告称，2019 上半年中国公有云服务整体市场（Iaas/PaaS/SaaS）达到 54.2 亿美元，并预计在未来 5 年间内以年均复合 46% 的速度快速增长[3-4]。数据中心内服务器计算力呈现异构化趋势，GPU, AI Chip, FPGA 等使用非通用类型指令集和特殊体系架构计算单元已成为目前分布式计算领域的热点话题。现在超大型数据中心一般可容纳数十万台终端服务器，内部网络链接数量多、拓扑规模大、传送海量数据，这使得现有的网络将变的异常复杂。同时，新的数据包类型层出不穷也使得现有网络变得异常脆弱。

传统网络技术已经无法满足当前的网络环境的需求，最近十年来网络技术和架构经历了快速地演进和变革，针对数据中心网络尤为明显。传统网络的互连包含了经典的二三层网络。为增强交换机的扩展能力，二层网络增加了广播，桥接等复杂功能。这种网络架构在小规模应用时可以展现强大的智能性与可扩展性，但当网络规模进一步增加，网络中容易出现的广播风暴、链路收敛等一系列尖锐问题变得难以解决。现代的大型网络设计思想摒除了略显冗余看似小聪明的功能设计，事先规划好网络拓扑层次，完整地保留网络的第三层，从而将网络扁平化。网络拓扑结构演化出可进行大规模扩展的 CLOS 型架构，为了降低系统复杂性，在各个层次之间的网络设备功能也逐步变得统一透明。网络设备统一化，可降低网络功能开发部署的难度。通常，研究人员需要持续地投入对网络进行测量、监控、容错、提升效能的工作。由于思想的创新和技术的推进，设备厂商不断开发出具备各种高级功能交换芯片。设备、芯片功能强大的同时，复杂的网络功能不断地对于网络的管理层又提出了新的挑战。

① $1Ebps = 10^6 Tbps = 10^{18} bps$

为解决设备制造复杂和设备管理复杂的问题，软件定义网络（Software Defined Network，SDN）概念的提出拨开了笼罩在网络体系结构发展道路上的迷雾。SDN 将数据平面和控制平面解耦。在数据平面上，对数据包的处理统一做查找-转发（Match-Action）抽象。控制平面复杂建立网络拓扑，控制并下发流表。这样所有的数据包转发行为都又控制平面的软件逻辑完成，数据平面可以支持任意一种网络协议的处理。由于软件具有强大的灵活性以及开发的敏捷性，SDN 大大加速了网络创新和智能化进程。数据平面和控制平面的安全通道由 OpenFlow 协议进行规范，将数据平面统一化、简单化，使得网络交换设备向白盒化方向发展。

大规模网络无论是在底层设备架构还是运维方式上仍不能停止变革的脚步，这为可编程硬件的发展带来了巨大空间。随着云服务概念和大规模机器学习的落地，近年来以云计算为代表的数据中心网络规模指数增长。网络功能虚拟化在数据中心内部是关键一环。虚拟交换机则是主机内各虚拟机之间数据包转发的核心软件。随着众核 CPU 架构快速发展，服务器内虚拟机布置资源大幅扩张，促使主机出口吞吐量从 40GbE 向 100GbE 甚至 400GbE 演进。不但如此，复杂的网络安全规则、流量监控等模组进一步导致 CPU 过多地消耗在处理网络功能上面。研究人员可能需要花费大量时间去解决目前网络架构的大规模扩展的方案。（创新变的异常艰难）。传统 x86CPU 架构适合于处理灵活多变的计算控制任务，对于做重复、常规流式数据处理，通用指令集架构并不能得到最优的效率，厂商往往不得不依靠大量部署 server 来解决。为缓解主机内 CPU 消耗过大，目前提出的智能网卡是一种新思路。智能网卡采用 FPGA，网络处理器，ARM 等器件，或以他们的组合形式形成在网卡端的新的算力集合，这种算力集合对于处理网络流量会有更高的效率。我们可以把转发动作，网络安全规则等功能下放进来，以削减服务器 CPU 的额外消耗。ASIC 具有最好的性能和最高的能量效率，但每次大批量的部署消耗时间长，投入研发资金大。对于运营商来说，设备、仪器等一次性支出都叫做 CapEx（Capital Expenditure，资本性支出）。对于目前快速发展的网络环境架构，设备的更新换代周期也在变短，在优化 CapEx 时已经不能把固定设备投入当做一次性支出。在探索新一代网络架构时，CapEx 也会成为重要的参考因素。

随着创新性和需求的进一步发展，让底层硬件拥有灵活的可控制能力才能满足目前行业变革的需求。因此，网络领域提出了编程协议无关（Programming Protocol-Independent Packet Processors，P4）概念。P4 协议不但支持 SDN 网络控制和管理的可编程性，还提出了数据平面可编程的概念。数据包在数据平面内的处理模型遵循解析-查找-匹配的抽象模式。P4 规定了一种编程语言[5]，它可以控制数据平面对数据包的任意解析行为，也可以自由配置查找表的数据位宽和多级流表之间的查找流水线[6-7]。这种更高阶的数据平面可编程模型使交换机设备更加白盒化，交换机与任意网络协议解绑，带来了具备灵活性的创新实践。除此之外，端到端大带宽、低时延的网络需求引申出了网络功能硬件卸载、网络随路计算等概念，这进一步增强了对高性能的网络数据平面可编程性的需求。

综上所述，现代网络在向软件定义、数据平面可编程的方向发展。网络架构的变迁

的核心是有一套可以映射上层可编程逻辑的硬件数据平面。本文主要探索一种面向网络数据平面的可编程硬件，能够满足快速迭代的网络创新性需求，同时能够提供与目前主流设备相仿的处理性能，以及可扩展性高的全局优化方法。

1.1.2 技术简介

软件定义网络的基本设计概念是将数据平面与控制平面分离[8-9]。其中，网络数据平面是指完成计算机之间通信数据包的匹配、修改、传送、转发的软硬件设备。数据平面的可编程性要求网络管理员拥有对数据平面的各个特性做快速个性化定制。网络的控制平面维护全网视野数据，配置针对流的转发条目，控制平面中的应用程序几乎都由软件构成。当前数据平面的设计思想如图1-1所示，主要有软件方法实现，专用硬件实现和新设计的可编程硬件。

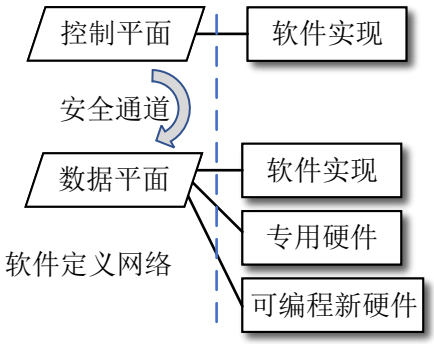


图 1-1 软件定义网络结构及其实现方案

在不同场景下，网络对于数据平面的需求千差万别，研究人员一般根据场景的流量大小，处理过程复杂度来思考并选取数据平面的实现方案，本文将在第2章详细介绍各类数据平面的实现方案的优缺点，并着重于可编程性的分析。目前两种最重要的数据平面是“软件交换机”和“专用硬件交换机”。两者在功能上都是针对数据包做一系列处理，包括匹配、查找、统计、传送、转发和安全校验等等，其中“流表”是实现数据平面核心功能的函数（器件）。数据平面、都包含一个可以与远端控制器沟通的软件代理，这部分功能着重于通信协议的实现以及通道安全性加解密，主要由轻量级通用处理器完成。其二者的主要区别在于处理数据包的性能以及交换容量。数据包处理性能主要看数据吞吐量（字节每秒）和包吞吐量（包每秒），目前软件交换机做高性能的包转发几乎可以达到 60G/60Mpps[10]。当数据包处理复杂度增加时，软件交换机的性能会直线下降，几乎与操作步骤数成反比。专用硬件交换机有接口数目多，交换容量大的特点，一般能满足 64 口乘以每口 25*Gbps* 的总交换容量。而且硬件交换机的性能与数据包处理步骤几乎无关，它拥有良好的性能稳定性，低转发时延等特性。虽然在核心网络和高性能网关领域主要使用硬件交换机，但是硬件交换机的功能固定，更换成本高昂，如果需要修改网络或者更改网络功能那么选用专用硬件交换机的场景将无从下手。所以目前在数据中心网络或服务器 NFV（网络功能虚拟化）等场景中，软件交换机依然占据很大份额。由于软件交换机的灵活性高，开发人员能够快速迭代部署新功能，且

传统单机 CPU 通信速率需求不高，软件交换机尚能满足在数据处理时延高、吞吐率低的前提下，提供足够的可编程灵活性。但随着人工智能领域、5G 的发展，数据中心网络内通信容量需求快速增长，转发时延需求快速收紧，软件交换机性能瓶颈快速到来，将不得不面对大量无谓堆叠 CPU 的情形。本文将主要侧重于研究主机侧网络和核心交换网络中使用可编程硬件来大大提高交换机的性能瓶颈。针对控制平面，本文将从单点优化开始用分布式优化和全局优化的思想，实现对网络中的瓶颈资源（如流表资源）的可扩展性和安全性提升。

1.1.3 国内外应用与研究现状

为增强数据平面的可编程性，工业界学术界互相促进、广泛研究并已经提出了许多方案。

1）基于软件的数据平面

这类技术着重于开发便捷，价格低廉，无需在网络中部署专用设备场景，是快速实现功能的首选方案。目前在虚拟化的云服务系统中，已经部署了大量基于软件的功能：a）转发层，华为 CE1800V[11] 是专为数据中心云计算虚拟化环境部署的一种分布式虚拟交换机。其支持标准 Open Flow1.3 控制协议，以及 Open vSwitch 数据库管理协议（OVSDB），基于英特尔 DPDK（Data Plane Development Kit）技术提供每核 12Gbps 的转发吞吐，比业界平均水平高出 20%。b）流量监管，Activelogic[12] 是一个提供安全可靠、流量分类、提高 QoE（Quality of Experience）能力的网络管理工具。它基于软件可自动化部署，依靠超大规模性能、人工智能技术以及云计算场景优化的能力，在数据平面解决流量监管的问题。基于软件的数据平面功能可以依靠堆叠 CPU 核数来实现大规模的性能扩展，但由于计算复杂度过高、基于指令的图灵机在高速内存共享和海量数据处理场景中效率低下，即使简单转发的性能达到 100Gbps 线速也需要占用 8 个核心以上[10, 13]。综上所述，我们发现单纯地依靠软件处理器扩张来增加网络性能边界收益将越来越小。

2）基于白盒交换机和 P4 专用芯片的数据平面

在网络性能方面大幅超越基于通用服务器的 NFV 数据平面[5, 13]。符合 OpenFlow 规范的白盒交换机可将控制平面移交给远端软件层，从而大幅提升设备的再开发能力，在 DDoS 防护、负载均衡等基础网络转发设备的智能化和可定制化方面给出了比较好的灵活性。阿里巴巴在其云计算网络场景中，通过可编程硬件交换机和通用服务器结合来实现公有云的网关服务。此架构既享受到芯片带来的网络转发性能提高（6.4Tbps，400ns 延迟）和可编程能力带来的网络功能快速部署迭代，又能实现软件所擅长的复杂网络调度功能[14]。这样同时兼顾了性能、灵活性，在大规模扩展网络体系结构时达到降低成本，满足业务需求和简化网络架构同时提升服务稳定性。数据平面可编程芯片提供了硬件层面上的可编程包头抽取器、可编程流表以及可编程执行器，他们的设计思想是依靠快速查表（TCAM，SRAM）法，或经过后期编程选取特定的冗余逻辑模块（在 ASIC 芯片内部的空间上堆叠的可编程单元）法，来完成专用电路（ASIC）的直接

描述逻辑[6, 15]。不过这类可编程芯片架构提供的可编程执行器是不完备的，前后堆叠的流表限制了流表的宽度、深度范围，会造成逻辑资源浪费以及流水线处理延迟过长。同时，ASIC 设计定型之后无法增加新的用户特性（状态转发、随路计算、监测计数和包调度特性），导致这类 P4 专用芯片的可编程性是大大受限的。

3）基于 FPGA 的自主设计的数据平面

现场可编程门阵列（FPGA）是一种灵活性可以与软件媲美可编程硬件，性能和效率与专用硬件比较接近。现代高速度云架构依赖于每个专用硬件（ASIC）网络节点的支持，随着网络功能需求多变与复杂化，ASIC 类型的网络处理芯片已经不能提供足够的可编程性，然而 CPU 核心无法提供高的处理性能。业界已经开始将网络堆栈向基于 FPGA 的自研网卡中卸载[16-17]。为了推广可编程硬件，学术界牵头推出了基于 FPGA 的智能网卡开源项目 NetFPGA[18]，业界龙头企业 Xlinx、Intel 等也纷纷推出了基于 MPSoC/FPGA 的自适应计算加速平台 Alveo[19-20] 系列智能网卡和 N3000[21]。目前，基于 FPGA 的可编程数据平面已经广泛应用在 5G 接入边缘网络[22]、数据中心计算存储[23]、核心网络低延迟加速器[24] 以及高性能高可靠性高安全性的数据中心防火墙[25] 加密通信[26] 等领域。FPGA 的高灵活性由全可编程的逻辑门带来，目前一般用硬件描述语言 Verilog、VHDL 等开发。一个合格的硬件工程师的培养周期要远大于软件工程师，这也是目前网络领域硬件卸载最难所在。为了解决这种不足业界也推出了一系列类似 C 语言的高层次综合工具 HLS[27]，但使用这类工具必须学习 1000 多页的开发文档[28]。并不是所有代码都可以直接被工具转译，而且还需要考虑到硬件细节，降低 FPGA 资源消耗；需要自主决定并行区块；需要在代码中融入这种编译器的特性标记字符，总体来看，目前并没有从本质上改善对硬件编程的困难程度。除此之外，由于在 FPGA 中复杂逻辑对并行总线宽度的时延敏感度高，一个大型工程的主频一般不会超过 200MHz，即使每个时钟节拍都可以处理一个数据包，那么 FPGA 流水线在处理最小包时的最高吞吐量也只有 134Gbps②，这对进一步需求性能的核心网包交换场景也形成了瓶颈。

1.2 研究内容

本文主要探索基于可编程硬件的高性能网络数据平面。论文提出基于可编程硬件的网络数据平面，对主机侧网络和交换层网络的数据平面实现加速，并研究在软件定义网络（SDN）概念下控制平面对全网核心流表资源的全局优化方法。如图1-2所示，论文把操作系统软件网络堆栈的大负载的网络存储和计算功能向网卡硬件卸载，利用 FPGA 与交换芯片使能交换网络数据平面的高性能高可编程性，把数据平面的主机侧网络、交换层网络的普通转发设备替换为具有硬件可编程特性的网络设备。流表资源是网络转发数据包的核心指令依据，本文基于软件定义网络控制面数据面分离的特点，对全网的流表资源进行了全局效率、可扩展性和安全性优化。

1）研究可编程设备加速主机侧网络方法

本文提出利用基于 FPGA 的智能网卡卸载操作系统层部分网络功能，以达到扩展

② 134Gbps=200Mpps*(64+20)*8bits

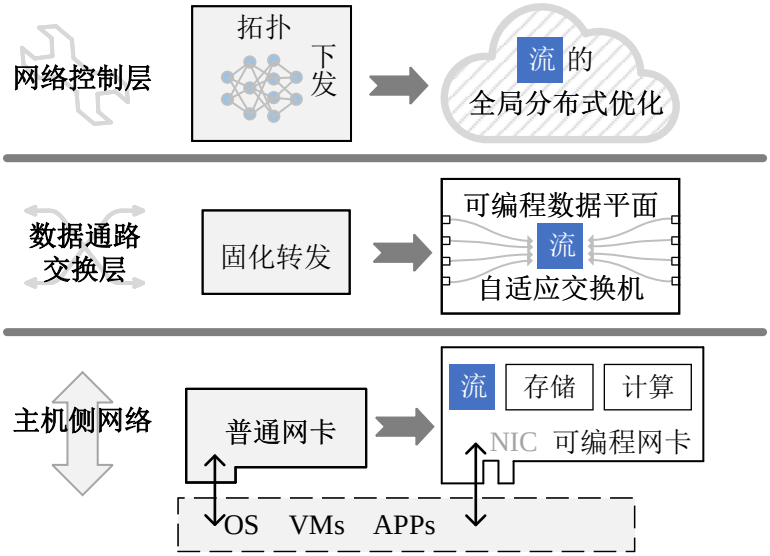


图 1-2 基于可编程硬件的 SDN 数据平面研究框架

网络接入层的性能的目的。探讨了不同场景下网络功能的构成，分析并提出一种基于可编程硬件的网络功能定义抽象（Data-Computing，DC 抽象）。本文把服务器网络功能任务中可转化为 DC 抽象的计算密集型功能通过合理转换下放到网卡的 FPGA 可编程器件中。论文针对网络流量捕获，统计分析和回放等功能场景，将其功能利用 DC 抽象方法，合理化地卸载到硬件网卡。在满足网络功能不受改变的前提下，证明利用基于 FPGA 的智能网卡能有效地提升服务器的网络性能、时延和效率。

2）研究可编程设备加速网络硬件交换层方法

本文提出一种硬件异构型的可编程网络数据平面架构，将 FPGA 与 ASIC 交换芯片有机结合，以增强 ASIC 报文处理报文的灵活性，同时满足性能需求。论文设计了 ASIC 面向硬件可编程扩展的接口，将数据包头拆分并通过高速数据互联载体发送给 FPGA，利用 FPGA 可重配特性实现完全可编程的报文处理数据平面；同时，本文基于 DC 抽象，将网络随路计算（network-centric computing）模式引入可编程网络体系架构；本文通过分析流量模型在 FPGA 中设计了一种并行化处理单元，在资源消耗可控的前提下大规模提高系统的可扩展性能；另外本文提出了一套基于可编程硬件混合网络架构的软件定义语言编程框架，实现了软件定义需求和可编程硬抽象层分离，以及针对底层数据平面的一种高效自适应的并行单元流分配算法，可以稳定实时地保障系统交换层的高性能。

3）SDN 硬件流表可扩展性研究

本文针对不同层面网络设备的控制，进行全局优化、分布式优化。在可编程网卡和交换机组成的网络系统中，数据平面内最重要的资源是流表资源（瓶颈资源），本文从全局视野角度，结合可编程硬件的特性，在全网约束的条件下，对流表资源进行优化，以满足未来可扩展性需求。本文分析不同的流量规模和特征，以及系统多模块直接独特的互联协议，提出一种 SDN 网络流表空间全局共享机制。实现了在流量大规模扩展

的情形下，保证数据平面稳定性，降低系统中关键通信通道失效风险。

1.3 关键科学问题

1）精度高、性能可扩展性强的软件网络流量功能卸载方法

面对当前数据量庞大复杂的操作系统网络环境，业界一般会使用专门的软件传输加速工具库（例如，DPDK[29]），也会使用到例如 SR-IOV[30] 的专有硬件加速。新一代的网卡还会支持 VXLAN、GENEVE 等封装技术的卸载，同时基于硬件的远距离直接内存访问（RDMA[31-32]）技术大有取代 TCP 协议栈的趋势。然而这些基于固定转发平面的卸载技术只能将虚拟化的转发层或者 TOE（TCP Offloading Engine[33]）卸载下去得到硬件加速，一些基于随路流量的有状态计算、并行计算以及灵活的流量工程却依然难以享受硬件加速带来的优势。目前基于 FPGA 硬件可编程网卡同时提供了高性能收发和足够强大的灵活性已经可以满足主机侧网络的性能需求，为更复杂功能的卸载提供了有力支持[34-35]。如何利用可编程网卡实现高精度、高性能保障的网络功能硬件卸载，并且提出网络功能抽象、合理部署、合理划分任务是本文要解决的第一个问题。

2）高资源利用率、高动态性的高性能硬件可编程数据平面设计方法

在云、服务器–客户端的计算网络体系结构下，由于新兴的内容应用（社交，虚拟/增强，混合现实）以及工业网络应用（移动性，大数据，机器学习）导致网络追求高的实时性、可扩展性和可靠性。网络设备数量和多样性随着数据中心、边缘设备的发展而壮大，因此，现在学界对交换层、核心网场景快速创建灵活解决方案的需求也愈发强烈。可编程数据平面交换机拥有很高的灵活性，可以快速重新定义新的数据包处理协议，为应对新形态网络发展提供了良好前景。其有三类典型设计架构但目前都存在缺陷：1）软件交换机性能普遍低下，2）基于 ASIC 的交换机无法拥有完全可编程性，3）基于 FPGA 的交换机资源有限，交换性能无法满足业界需求。综上所述，本文第二个研究问题：如何设计一款转发性能强，而又拥有硬件可编程性的交换机设备？如果这种设备所需求的资料是目前产业界无法提供的，有没有一种对现有设备进行科学合理的具有最小改动可能性的方法？如何实现高资源利用率、高灵活性的高性能硬件可编程数据平面设计方法？

3）流表关键资源的全局优化方法

网络数据包的转发动作依赖于数据平面内查找表的匹配结果，SDN 架构下亦是如此，当前 SDN 数据平面内将网络数据包的处理流程抽象为 Match-Action（匹配-执行）。在此基础上还交换机内增加了多种匹配域、多级流表结构，绝大多数平台中都视转发表为最核心以及成本占用最大的模块。以 OpenFlow 协议为代表，为更好的服务动态的新流，一般规定控制器与交换机之间流表安装流程为 Reactive 模型：交换机收到一条新流首先会上报控制器，随后控制器计算路径并下发流表到数据平面设备。基于硬件的高性能 TCAM（三态内容地址查找表）拥有单周期流水、掩码匹配等优秀性能，然而昂贵的价格使得用户无法购置容量足够大的表。因此，交换机内极易引发流表溢出的

现象，若此时新流到达此交换机节点并按照 Reactive 模型处理，由于可能需要频繁更替活跃流表内容，这会进一步直接引发控制平面和数据平面之间安全通道的消息风暴，否则会造成丢包或服务任务中断等异常现象。本文第三个研究问题：如何在维持交换机中原有流表容量的前提下，缓解流表溢出所带来的危害？在保持 SDN 网络平面分离优点的条件下，如何利用其全局化优势高效利用网络设备资源？

1.4 主要研究成果

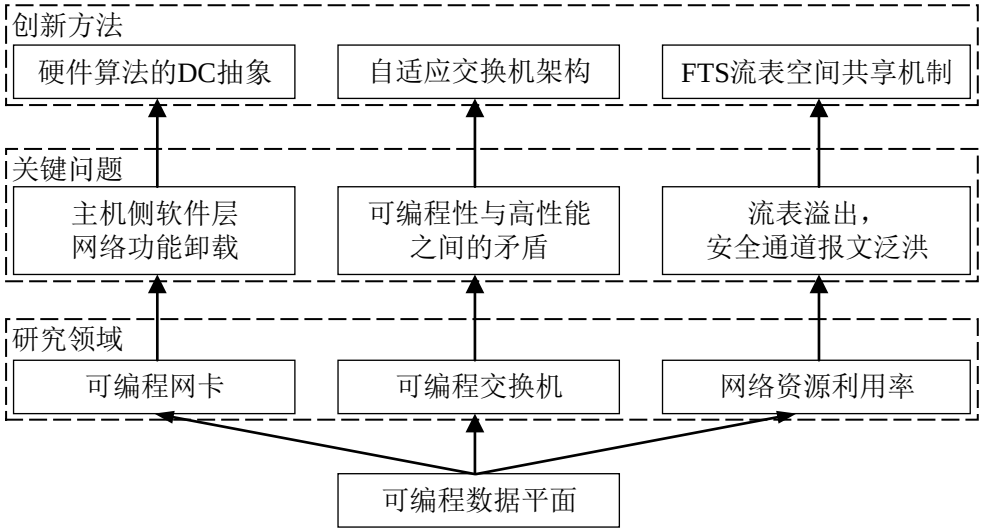


图 1-3 论文主要研究内容以及成果

论文针对可编程网卡卸载有状态计算和流量工程、基于 FPGA 可编程硬件性能不足，流表溢出威胁风险大网络资源利用率低等问题展开分析和创新方法设计。如图1-3具体研究成果概况如下：

1）提出了针对流量随路计算的网络功能卸载抽象模型

本文提出一种适用于网络功能硬件卸载的抽象模型：数据—计算抽象（DATA—COMPUTING, DC 抽象）。根据 DC 抽象，分离软件中适用于硬件加速的繁杂计算，使原本经 X86 计算架构需要频繁访存的任务，转换到硬件中做流水线式流计算，可在不影响功能精度的前提下，释放 CPU 资源，大规模扩展性能，提升系统效率。同时，再配合数据包的分类—查找抽象（Classification—Matching，CM 抽象），论文在可编程硬件的网卡中实现了更高精度、更高性能、资源利用率更好的流量捕获-统计-回放应用。在满足网络功能不受影响的前提下，证明利用可编程硬件能使原有软件性能效提升 100x、抖动降低 4 次方数量级、能源效率提升 10x。

2）提出了 FPGA 与交换芯片（Switching ASIC）结合的自适应交换机架构

本文提出一种高性能的可重配交换层数据平面架构：自适应交换结构（Adaptive Switch, AS）。通过 FPGA 与交换芯片联合的设计思想，AS 架构将 FPGA 的高灵活性与交换芯片的强大性能同时对外表现。论文在前述 DC 抽象的基础上，继续研究 FPGA 可

编程硬件中高度并行的大规模性能扩展方法。为了保证 FPGA 低资源消耗，论文设计了一种基于硬件的灵活负载均衡机制。综上，AS 架构解决了 FPGA 性能差与资源少的限制，与交换芯片的有机连接更进一步增强了 AS 架构的整体性能。综上在可编程性与纯粹 FPGA 等同的条件下，论文将目前基于 FPGA 的可编程数据平面性能提升 120x。

3）提出了一种针对流表资源不足场景下的网络内流表共享机制

网络转发层核心资源不足的问题，本文提出一种全局流表共享方法（Flow Table Sharing, FTS）。本文分析目前 OpenFlow 协议中有关 Table-Miss（流表缺失）的处理过程，并论证即使单纯依靠增加流表容量的资源堆叠方案，并不能使流表溢出的概率降低为零。本文在维持 SDN 网络控制面悬离特性不变的前提下，提出新的 Table-Miss 处理机制。FTS 方法通过控制器层面、交换机数据层面的软硬件联合设计方法，使得新的 Table-Miss 机制能实现对原先受影响的转发流量 RTT 时间和安全通道消息风暴数量的优化均达到至少 2 个数量级，并且能够容易回退、向下兼容现阶段的传统方案。

1.5 论文组织结构

根据主要研究内容的讨论，本文的组织结构安排如下：

第 2 章对相关工作进行调研，主要介绍网络中主要数据平面，以及其可编程化发展趋势，分析应对网络软件定义化的主要挑战。

第 3 章网络计算、流量工程卸载方法。

第 4 章自适应交换机，可编程数据平面，网络交换层。

第 5 章网络资源全局优化方法，table-miss 处理方法。

第 6 章总结。

2 相关工作综述

2.1 本章引论

本章综述了国内外网络基础设施技术的演进，主要分析其主要技术特征和局限短板，重点关注了现阶段实际情况下 SDN 可编程数据平面的灵活性与性能矛盾点，以及 SDN 网络架构下网络设备硬件资源匮乏的现状，为本文研究工作指明了方向和意义所在。

2.2 网络可编程的发展历程

2.2.1 软件实现—早期网络基础设施

网络对于业务的基本价值是网络实现了数据在计算机之间的任意传输。在早期[①]，由于用户数量、计算机算力、存储、硬件性能都过于微弱，作为连接所有终端、服务与用户的管道，网络的主要特点集中在连通性、可行性和初期探索性上。在一个简单的星型拓扑中，一个路由器其实就是一台普通计算机。在学术和产业界的初期，人们并没有意识到网络需要单独拎出使其成为一套独立系统的价值。这在侧面也体现出软件作为网络实施载体的特点：“灵活性”。即：对于处理并实现一个新兴事物，软件可以发挥其巨大的灵活性优势，使其可以作为一种为数不多的手段，快速实现工程师学者的任意的新的思想。

后期随着社会生活、技术进步，步入信息时代之后逐渐发现人与人之间数字信息交互的需求和价值越来越大。因而研究重点开始关注在如何实现快速的包交换、路由查找。为此人们开始提出各种快速交换的数据结构：Cache 优化、哈希表、Radix Tree(树查找) 等。很长时间基于软件的转发设备核心架构都没有变化，唯一变化的是跟随摩尔定律成长的芯片技术。CPU 和存储每 18 月性能翻番，网络设备的性能也顺势而上，人们对网络的发展信心十足。网络处理从单 CPU 向多 CPU 并行，向分布式存储 cache 结构进行了小小扩展，但也好像失去了创新的动力。然而人们对信息量需求的增长却大大快于摩尔定律。到 2011 年底，我国互联网入户带宽平均接近 20Mbps[36]。从最初 14.4Kb 的拨号上网，网络容量的发展几乎是以每 18 个月翻 10 倍的速度在增长。在数百兆的路由性能要求下用软件作为转发设备基础比较合适，但如果核心网要升级到 1G 或数十 G 以上更高的带宽就会面临技术、成本等多方面的瓶颈。

2.2.2 向硬件过渡

数据包交换对于 CPU 来讲是一种很累的工作。虽然数据包转发算法既简单、又高效，但面对无穷无尽的任务量，依靠指令集的软件转发架构存在访存效率差、CPU 无法

① 上世纪 90 年代中期以前

批处理等劣势。这时研究人员抛弃了基于指令集的软件架构，开始思考基于专用硬件电路（Application-specific integrated circuit, ASIC）的数据包处理模型。此时硬件转发的发展目标是如何增大交换设备的交换容量、以及研究具有更好的可扩展性的设计方案。电路交换 Crossbar（交叉开关）[37] 架构追求 N 队列输入到 N 队列输出的无阻碍转发，其思想的本质是使用一种二维电子开关（Switching）矩阵来增强交换设备的转发能力。矩阵中有 N^2 个开关交点，可以实现任意的 N_i 输入映射到 N_j 输出，也易实现多对一、一对多映射。因报文长度不固定开关数量大导致控制器硬件算法难度高[38-39]，以及传输冲突等问题[40]，此后的一系列技术创新集中在如何降低 Crossbar 的管理时延、提高理论吞吐容量[41-42]。人们也在思索如何在扩展交换容量时节约芯片面积，其中重要的思想是由单模块交叉开关联结为多交叉开关结成的网（fabric）[43]。有专用硬件电路的加持，业界把单芯片交换能力提升至 25.6Tbps[44]。能够支持在一个大规模数据中心内可以支持 256 台配置有 100G 网的卡服务器形成一个小区进行高速互联，这样的组网计算机的并行处理能力已经足够一个通常规模大数据算法使用。单芯片容量升高会使晶体管面积成 $O(N^2)$ 规模增长从而变得不再划算。如果想支持 1024 台服务器，网络架构商可以选用两级 Spine&Leaf(骨干与边缘) 架构，使用 12[②]块 25.6Tbps 的交换芯片组成一个 102.4Tbps 的扩展规模网络。

当设备被大规模组网时，网络的管理问题变得尖锐。早期互联网的发展非常迅速，因为设备的扩展就是简单对接，每个设备独立控制自己，具备对外扩展的策略。随着时间的流逝，网络内产生了成百上千个新 IETF RFC(网络工程备忘录) 和 IEEE 标准。设备制造商需要用同一个产品向各类运营商提供服务，这导致在同一款路由设备产品中堆叠的功能特性也越来越多。一些 ISP 路由设备的源代码甚至超过 1 亿行，是最复杂电话交换机的 10 倍以上，要知道电话交换机也曾需要支持上百种协议[9]，即使大多数客户只需要其中某一种功能。互联网也为这种高复杂度付出了代价：设备臃肿部件数量庞大、不节能效率低下、价格昂贵、API 的设计随意。由于路由设备行业门槛较高，初创企业难以进入市场并发挥创新能力。此时大的路由器供应商也为路由器的可靠性、高复杂性、安全性等问题苦恼，网络的创新速度又变慢了。

2.2.3 软件定义网络演进—软、硬任务划分，物理隔离

1）数据平面的统一化与精简控制软件。

软件定义网络（Software Defined Networking,SDN)[8] 的概念赋予运营商集中式或半集中式程序控制的便利。网络设备控制面和数据面的物理隔离，给这种体系架构带来经济学层面的优势：能将复杂的数据平面管理功能软件集中在少数几个地方，具有统一设计的数据平面抽象。最开始人们发现，每一个运行在网络系统里的数以千计的交换机和路由器都运行着一个程序处理器。大量这种分布式控制设备的数据平面运行的软件其实是一样的，但却需要设备数量十分之一[45] 的网络管理员去不停地确保网络正常运转。相比于运维效率低，不确定性才是最危险的。由于传统网络的控制平面是分布

② 12=4(Spine)+8(leaf)，每个 leaf 节点对外暴露一半的接口容量，最终扩展 4 倍到达 102.4Tbps

式的，在正常运行状态下没有人能够有一个清新的网络运行图，因而在网络出现问题时管理人员很难调试。对于数据平面的设计思想也很直接，数据平面必然完全接受控制平面的控制策略，而数据平面输入输出都是数据报文，那么数据平面内的所有操作都可以由 Matching–Action（匹配—执行）模型抽象出来。网络的功能就由远端控制平面上的软件来定义，这将有助于网络的“创新力”。因为网络功能、协议的定义不再只能由设备供应商提供，而能够由真正维护和使用网络的操作人员现场定义/修改。同时，操作人员能够拥有网络全局视图，对保障网络运行和安全控制也有极大的促进。

数据平面与控制平面之间的交互称为“南向接口”，目前南向接口的事实标准是 2008 年由斯坦福大学提出的 OpenFlow 协议。OpenFlow 协议由最开始的 OpenFlow1.0，快速发展到现在的 OpenFlow1.6。几年时间，OpenFlow 协议已经逐步完善到网络的各个细分领域：流量调度[46-47]，光适配[48]，广域网[49-50]，超转发③[51] 等。

2）云和虚拟交换。

随着云计算的持续发力，虚拟化成为其中重要技术。虚拟机内部互相通信需求增高，基于 SDN 的 OpenVSwitch 同样也令虚拟交换机编程更容易、转发更方便。软件定义网络加速云虚拟化的创新，软件定义网络能够提供非常复杂的虚拟网络语义，支持快速迭代。数据中心网络性能的提升需求远远快于 CPU 的处理能力的增长，通常来讲，CPU 一个核心能够支持 10Gpbs 的转发性能。对于未来数据中心服务器百 G 带宽需求，也许需要消耗 CPU 总体性能的 20%④。

2.2.4 协议无关数据平面可编程演进—可编程性层次划分，逻辑隔离

1）扩充报文编码与设备快速更新

如果说 SDN 给出了控制层的全局视野，那么这种协议无关可编程的数据平面给出了设备层的全局视野。SDN 已经将数据平面高度抽象，操作人员可以灵活的定义什么样的流，以及对这种流进行怎样的操作。但是在数据平面内数据包头的匹配域却是预先规划好的。固有转发平面的设计思想会引起如下两个问题：其一，添加新特性需要跟业界讨论、以及等待很长的设备研发时间；其二，在数据平面内固化现实中可能出现的每一个网络协议字段造成宝贵计算资源的巨大浪费。满足新阶段的网络创新需要比 SDN 更好的灵活性、动态性。因此斯坦福大学提出了 P4[5] 编程语言框架，这种语言有能力重新定义数据平面的包解析模式。P4 源代码通过前端编译器编译为中间表示层代码，这个编译过程将提出源代码中的语义逻辑。之后需要根据不同的目标器件再进行后端编译，这个过程最终会生成目标器件对应的机器码，硬件可直接读取。目前 P4 的目标设备已经有基于 ASIC 的交换芯片、CPU、GPU 和 FPGA 等多种实现。

P4 是与流表式编程不同，它是另外一种维度的高层次可编程概念。在 P4 框架中，网络操作者可以根据新的设计，创造性地自行设计一种结构的数据包头字段。通过 P4 源代码，将新的包头结构编译到数据平面形成新的指令。这就实现了灵活定义数据平

③ Super Packet Transport Network, SPTN。一种硬件功能组件可分解的高效可编程网络框架。 ④ 以常见 Intel 志强 48 核心 CPU 处理器为例。

面解析过程。P4 的目标是让已经部署的硬件网络设备数据平面实现软件定义升级，可以达到在线无插拔地更换新设备的效果。P4 的出现也首次实现了数据平面不同逻辑层面上的可编程性。

2）可编程硬件的未来

数据平面可编程概念引发了众多新技术和为解决不同问题所提出的创新实践，如图2-1, 本文从不同方向架构梳理这些工作。

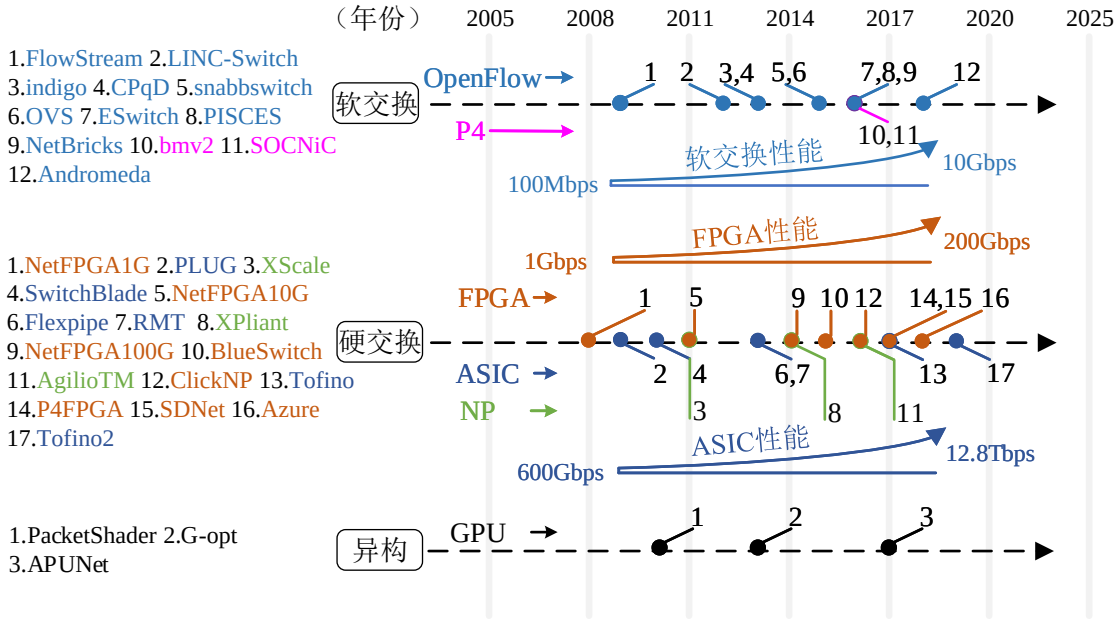


图 2-1 可编程数据平面各界发展历史

当设备处理接收进来的每个数据包时，数据平面是网络当中最关键的环节。通常需要用到专用的硬件设施，或者经过复杂优化后的软件加速方案。在硬件方面，数据平面可以在 ASIC[6, 15, 52-55]，FPGA[18, 56-62]，网络处理器[63-65]，外挂有三态内容地址查找器件（TCAM⑤）的系统[66]，在软件方面有基于快速包分类算法[67-69] 的在 CPU 系统上实现[10, 70-80]。从 2008 年提出软件定义网络，由于真实环境性能的需求，业界从未间断地开发基于硬件的可编程数据平面。在 P4 概念提出来之前，也有类似于半 P4 的混合型可编程数据平面，受限于设计架构，他们对于短长度域可以实现任意匹配，基本可实现常见协议的数据平面编程，但不能够有效支持宽域[52]。

由于硬件可编程技术的加持，外加比虚拟机更轻量级的容器、高速分布式存储、无服务架构、AI 对 I/O 响应速度的要求，使得网络体系架构设计发展繁荣、爆炸增加。相信在未来业界将会出现更多的应用场景，这些场景也将会不断催生出功能更强大的可编程网络、以及更强大的性能。

⑤ Ternary Content Addressable Memory, TCAM

2.3 网络可编程性的“图灵完备”

2.3.1 通用可编程性和可编程网卡

上一章提到，我们需要使用智能网卡来卸载操作系统内的网络功能，以期望获得比 CPU 更好的效能，同时还可以兼顾网络设计中不断变化的革新需求。智能网卡也叫做可编程网卡，相比于普通网卡，一种认知认为[81]：智能网卡不但可以完成网卡最基本的作用（主机与网络间通信），还应该有如下特征：输入输出多队列、TCP 卸载、流量整形、规则过滤、虚拟化等。从而增强一些通用场景下的网络性能：带宽扩容、优化 QoS⑥、降低 CPU 利用率、降低通信时延等。如图2-2，是一个典型的 ASIC 智能网卡通路，易见，网卡将各种网络处理过程（流分类、流量工程、协议）硬化到专用硬件逻辑上，使处理效能增加。不难发现，基于 ASIC 的智能网卡本质类似于一个操作系统的外挂交换机，只是他与主机侧链接的延迟更短，主机拥有其完整的控制平面管理能力。

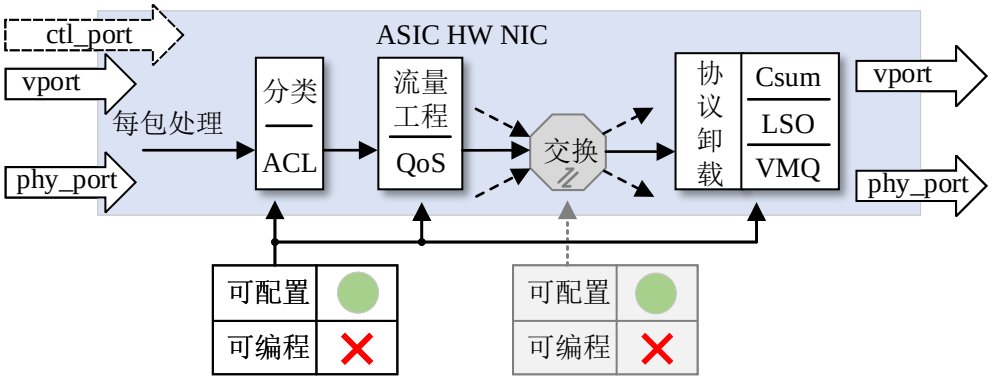


图 2-2 基于 ASIC 的智能网卡架构

这种开放控制平面的网卡架构，还称不上真正的智能网卡，因为它无法提供“核心部件”和“辅助部件”两方面的客制化的编程能力。首先，网卡的核心功能是一个数据包交换结构，完成“匹配—执行”操作，然而基于 ASIC 的网卡芯片出厂后就无法修改包头域的设置，核心部件不能实现可配置的交换，因而无法满足新协议的处理需求。第二，流水线中包括基于硬件电路的 QoS、访问控制（ACL）、协议卸载等辅助部件。这些卸载功能如果无法支持新的网络协议栈，那么此类功能只能从网卡重新回到通用 CPU 中处理，几乎失去智能网卡的性能优势。ASIC 的研发周期一般都比较久，并不能很好的适应目前快速迭代的网络架构需求，是缺乏适应性和可扩展性的。

随着时间的推移，人们还发现如果能够将计算[82-83]、随路功能聚合[84-85]、缓存[86]甚至 AI[87-88] 都卸载到网络上，有能力显著提高分布式应用的处理效率。目前能够支持这种将更复杂计算卸载到网络中的网卡，都要求此智能网卡具有通用型的可编程能力。

1）通用可编程的智能网卡

基于网络处理器（Network Processor, NP) 的数据平面，拥有完全的可编程能力。如

⑥ Quality of Service (QoS)，服务质量

图2-3上部所示，NP 芯片内部一般包括基于硬件的拥塞控制、队列调度、QoS 等协处理逻辑，还包括一组并行微码处理器。处理器按任务可分为核心处理器和转发引擎。处理器通过预先编制的微码来控制处理过程和内容。NP 编程模式简单，一旦有新的技术或者需求出现，可以通过软件语义重新定义数据平面。值得注意的是 NP 中的众核一般使用数据平面专用精简指令集，为了达到节能与节约面积，像浮点运算等复杂的处理指令是不支持的。NP 的每个内核处理性能一般较差，NP 的高性能主要靠结合使用专用外挂电路。一旦处理的内容无法映射到专用电路那么 NP 的性能会弱于通用软件。另外，NP 编程开发门槛较高，NP 运行软件无操作系统扶持。NP 的代码移植性差，开发人员需要深入理解 NP 的处理模型。因此 NP 始终只在一些狭窄的领域空间内发挥作用。

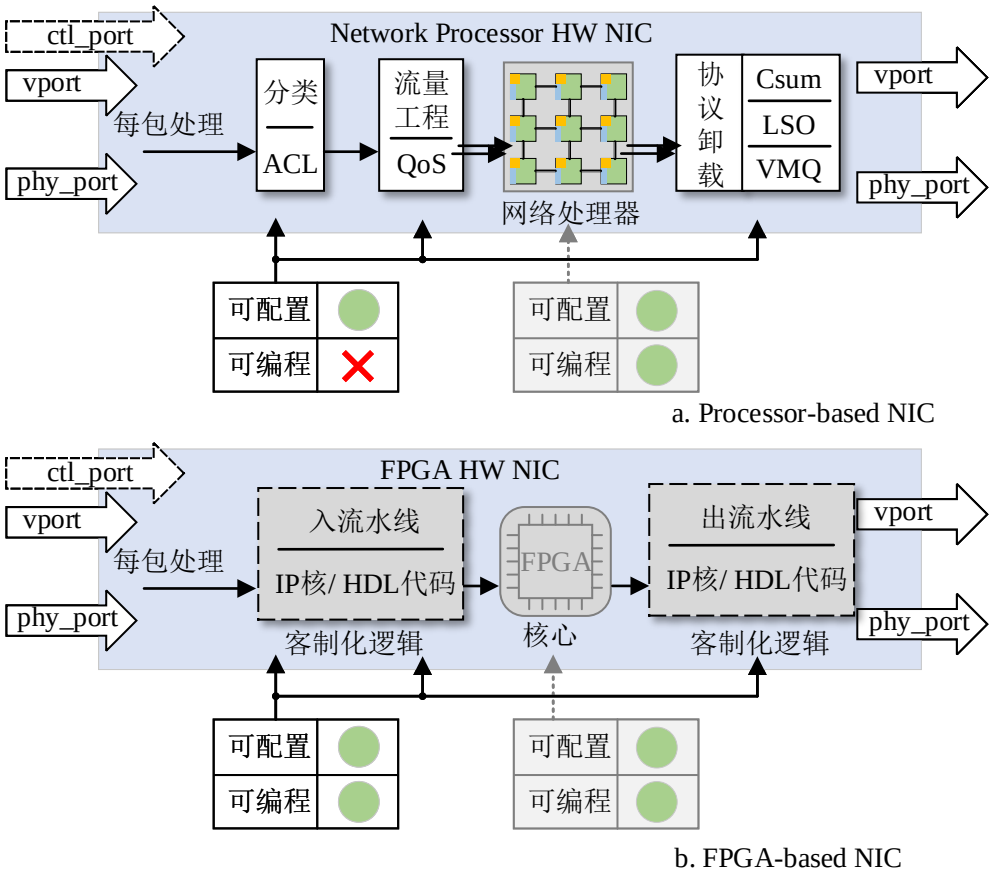


图 2-3 具有通用编程能力的智能网卡架构

基于 FPGA 的智能网卡拥有更为广阔何灵活的编程空间。FPGA 内部有大量 LUT 门电路，以及分布式片上互联网络，基于此结构的 FPGA 可以实现任何客制化的逻辑电路。FPGA 使用硬件描述语言开发（HDL），HDL 不直接体现门电路的拼接方式而只是一种行为描述语言，从而屏蔽了底层细节。如图**??**下部所示，FPGA 可以方便地移植程序，我们可以将 HDL 代码打包成 IP 核，只要按照规定好输入输出接口位宽和时序就可以任意复用。在设计电路模组时，我们一般会使用标准的总线接口来连接不同的功能模块，以增强开发的灵活性。如今 FPGA 厂商也会在 FPGA 中加入专用功能电路来增加芯片集成度、增强 FPGA 的处理某些任务时的性能。如 ARM 核、分布式 DSP 核、

PCIe 收发器、分布式片上存储。

2）灵活性与性能

如图2-4所示，基于目前业界的技术，为设计更灵活的数据平面，我们一般选取如下两种类型的系统做比较：其一，基于 NP 或 CPU 众核的智能网卡，拥有比较好的可编程性和灵活性，是具有“图灵完备”一类型设备，我们可以将其当做 CPU（计算）系统的延伸。但是他们的缺点也很明显：性能低，效率不足。其二，基于 FPGA 的智能网卡由于可以任意制定处理逻辑，也属于“图灵完备”的一系列设备。虽然 HDL 语言是高级描述语言可编程性强，但需要程序员基于硬件电路的思想来完成设计，学习成本高。这种思想层面中的“不灵活”作为一种挑战，又阻碍了 FPGA 的适用性。性能和可计算性如何更好地折中，或者如何选取一个更合适演进的线路图则成为本文主要考量之处。

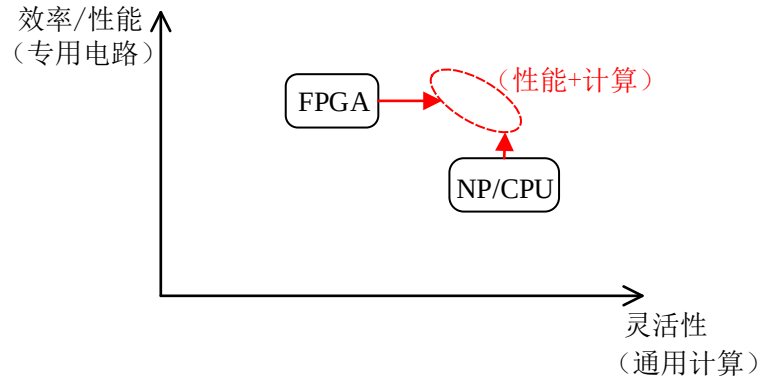


图 2-4 性能与灵活性如何更好的折中

2.3.2 领域内可编程性和可编程转发设备

1）领域内可编程性在不同的信息技术领域内有不同信息处理需求，从信息技术蓬勃发展的过去的几十年到现在，随着微电子行业的诞生，一直不断地涌现出各种类型基于某种专业硬件的处理器，往往这些设备都兼具有某种软件的可编程性。如图2-5所示，下面简要介绍历史各个类型可编程处理器。第一，中央处理器（CPU）。CPU 解决的是通用类型的计算问题，工作生产生活中，人们往往会遇到各种各样的数学计算任务，使用 CPU 去辅助人们完成这类枯燥且量大的工作可以极高的提升社会生产效率。CPU 采用冯诺依曼结构，是一种图灵机。它将处理通用计算任务抽象为控制-计算-存储模型。控制模块从存储器内读取程序指令和数据指令，并把他们按逻辑分配给计算模块。人们预先可以将需要处理的任务和数据编写到可重复擦写的存储器中，实现各种灵活的任务需求。操作人员就从繁琐的计算当中隔离开来，只需要去关注如何设计控制逻辑已经对应的需要计算的数据。第二图形处理器（GPU）。图形从本质上是一组二维矩阵数据，像素数据量一般都在百万级别。视频信号又是由一帧一帧的图像先后排列形成，导致处理图形的过程中产生大量的数据量。这些数据由 CPU 处理往往需要占用很长的处理时间，消耗大量的计算能力从而效率低下。GPU 架构提出，图像处理没有先后依赖关系，处理器对于每一帧图像处理的方式完全一致，因而可以利用多 CPU 并行处理

以达到加速目的。所以 GPU 就是众多微小的 CPU 的堆叠，同时增加了片上存储密度以应对并发的指令读取需求。第三，信号处理器（DSP）。与 CPU 类似，但 DSP 增加了专门为信号处理设计的指令集，使 FFT 运算更快速。DSP 一般是数据地址与内存地址分开的双总线结构，支持灵活的编程。第四，神经网络处理器（NPU）。神经网络的训练过程需要进行大量的张量运算，适合于使用并行度高的处理器做运算，例如使用 GPU。但在神经网络的计算中数据位宽往往比较低（8bits）如果使用通用处理器会有比较大的资源浪费，能源效率也比较低。随着 AI 技术发展，业界对算力的需求持续增高，研究人员专门为神经网络计算任务设计了一种专用处理芯片，TPU 与 GPU 相比在同样能源消耗下，计算完成时间可缩短 70 倍左右[89]。第五，协议无关交换架构（PISA）。网络包处理流程一般比较封闭，开发人员一般使用设备厂商固化好的网络设备进行数据包传输处理等。但由于网络功能应用环境的快速变化，研究人员发现固化的网络处理芯片无法满足增加新协议的需求，这严重制约了网络的创新。最近业界提出基于硬件的 PISA 模型，定义了可编程数据包处理的规范。它提出了开源的数据平面功能描述语言，为开发人员提供了可编程的包头描述能力，以及对应的包头信息抽取。在查找和匹配方法上，此类芯片使用多级查表法来实现任意的匹配和查找操作。

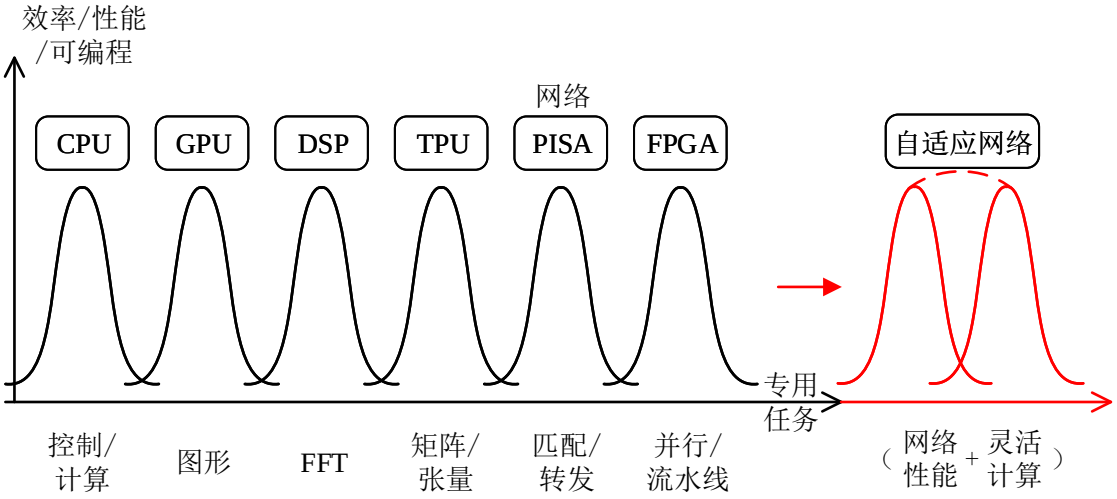


图 2-5 器件在不同的专用任务中性能和可编程性指标

在网络领域的可编程虽然较新，但已经引发业界很大的关注。不少工作都指出，可编程网络在超大规模数据中心网络、企业级交换网、网络内测量、负载均衡等领域都有广泛的实践以及优势。PISA 有能力只使用单一器件就可支持种类繁多的网络功能，这种大的灵活性可为企业节约更换设备的成本，统一结构的数据平台也使操作人员维护复杂网络的成本降低。综上所述，网络可编程也已经作为可编程专用任务中的重要一分子，在未来值得持续投入研究。

2）可编程网络交换芯片架构

网络设备的最基本功能是解析数据包头，并对数据包头信息进行转发、丢弃、修改动作。最初的基于硬件的网络设备对于一个数据包包头定义是固化的，每当增加新协议字段，这种固化的网络设备都无法胜任新工作。但如果所有操作都由 CPU 处理，则

性能十分低下，无法满足核心网、骨干网中性能的需求。后来人们针对处理数据包灵活性不足和性能问题对 CPU 进行架构优化，设计出采用辅助硬件流水线增强和多核并行方式的网络处理器。网络处理器拥有 CPU 般的灵活性，但由于本质还是基于 CPU 的指令循环操作，以及众核架构的内存、数据总线复杂度高数据搬运压力大，NP 最终难以持续优化，目前业界顶级的基于 NP 的交换芯片性小于有 1Tbps，这极大地限制了 NP 的使用范围。新兴的基于 ASIC 流水线指令集的可编程处理器为 NP 的性能不足带来了一种解决思路（PISA）。

首先，需要解决基于 ASIC 的可编程包头解析器。交换机中的查找表必须查找固定协议字段，所以每个交换机都会包括一个包头解析器，它能够标识当前数据包包头的协议名称。如图2-6所示，数据包是一组串行数据，从数据包的开始位置起，包头协议依次串行排列。每个协议段长度固定，每个协议末尾会有标识码标明下一个协议字段名称。本协议字段的长度一般都存储在协议字段内部。包头中的协议字段是一种有向图关系，图的节点代表协议，向量字段代表转移关系。一个包的包头不一定包含图中所有的节点，和对应关系，但当数据包到来时，包头所包含的协议只能是有向图中的唯一一种路径。一般使用有限状态机（FSM）就可以提取出包头内的所有协议字段。FSM 内部存储完整的图关系，只要按照图对应关系来给包头不同字段打上协议名称标签即可。

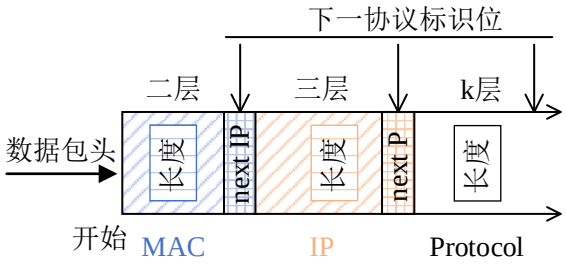


图 2-6 数据包包头结构

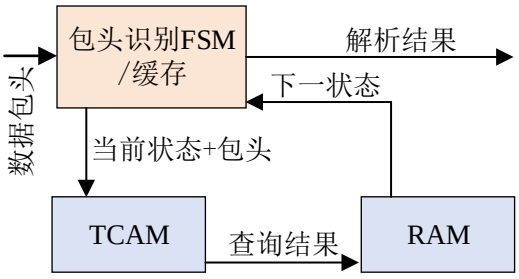


图 2-7 基于 ASIC 的可编程包头协议解析器架构

可编程解析器须实现灵活可运行时配置的协议图。可编程解析器中的状态转移图可以通过状态查找表来实现。状态查找表可以由 RAM 和/或 TCAM 存储器组成，RAM 是一种基于地址的内容访问存储器；CAM 是一种基于内容的地址访问存储器。CAM 中在不同地址存储有内容，当 CAM 接收到一个内容输入请求（key）时，可以并行搜索所有位置，并返回内容等于 key 的地址位置。TCAM 则是在请求 key 中可以定义“不考虑”bit 位，在判断二者内容是否相等时所有“不考虑”位都认为是相等的。如图2-7所示，TCAM 的 key 的宽度与一个完整的包头相等，在 TCAM 的一个表项中，存储着某一个协议的包头标识数据，这个数据的位置与真实数据包包头中此协议的位置相同，但是其他位置都属于“不考虑”bit 位。因而只要包头可以匹配此 TCAM 表项，就代表包头中有这个协议。当然由于包头域是个有向图，在不同阶段所需要看的标识位置是不一样的。下一跳状态信息就存储在 RAM 中，得到新的状态后，电路再去查找 TCAM，直到有向图走完。只要我们有足够宽的 TCAM 表，我们可以通过任意修改表中存储的

状态图信息来实现运行时可编程的包头解析器。

第二，需要解决基于 ASIC 的可编程流表匹配。在传统交换机中，当数据包完成包头域的解析，交换机通过查询流表来得到数据包的执行指令。如图2-8所示，处理每个协议的节点组成了协议有向图，节点一般可以抽象为“匹配–执行”的模式。由于包头协议状态转移图是固化的，所以交换机可操作的数据包的种类、数量也是固定的，其他数据包会被交换机当做未知类型而丢弃。在设计交换机之初，就需要根据各个协议字段不同位宽，不同流表匹配方法，制定固定深度的流表。因而目前固化交换机的数据处理核心都会设计异常复杂，需要适配各种有可能的协议，然而在某一个应用场景中只会存在其中一部分数据包类型，导致了很大能耗和经济的开销。

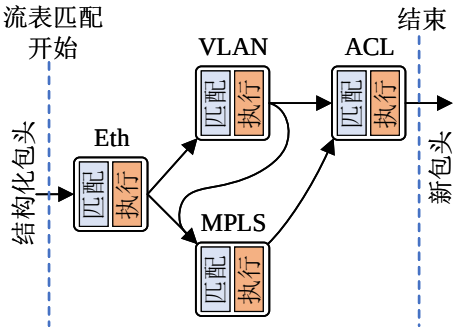


图 2-8 传统交换机中的查找匹配过程

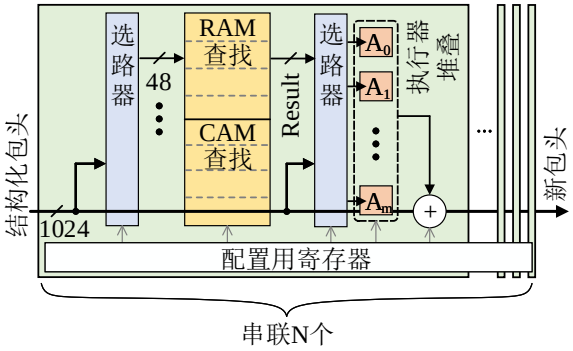


图 2-9 基于 ASIC 的可编程匹配模型

可编程流表须实现灵活配置“查找–执行”逻辑结构。如图2-9所示，目前的设计思路，可编程匹配流水线由 N 个“物理块”（Phy_Stage）串联而成，每个 Phy_Stage 可以独立配置查找表的位宽、深度和类型。而且在每个 Phy_Stage 中堆叠了所有类型的“执行器”（Action）模块，在运行时这些部件依次按流水线背靠背方式处理。选路器可以由多路复用器（MUX）或交叉开关（Cross_Bar）组成。由于这些机构都可以被后期在线配置，在这样的流水线中一个 1024bits 的并行包头数据进入物理块后由第一个选路器（MUX）选出“待匹配域”送入所需的存储器接口，匹配之后的结果和包头域信息被第二个选路器（Cross_Bar）送往所需的执行器中进行操作。最后执行器将新的域插入（修改/删除）包头内形成新包头。多个“物理块”可以先后呼应形成一个更复杂的“逻辑块”，最终通过运行时配置这些“物理块”可表达任意类型的协议有向图。

2.3.3 可编程数据平面的应用与问题

协议无关（PISA）处理器从诞生至今已经覆盖了广泛的网络应用场景：

1）替代传统网元（服务负载均衡[90]、安全控制[91]、流量控制[92]、测量[93]），使云网络自身成为一个软硬任务分配均衡且可编程的系统。

2）增加网络随路专用功能，如键值查询（key-value store）[94]。旨利用网络高速以及可编程交换机特点，使特殊功能的性能大幅提升。

在此之外，也有很多特性是 PISA 架构无法实现的：

1）包头长度有限，目前数据平面可编程的流水线紧紧局限于处理宽度受限的包头。

2）非图灵完全，只有有限个数的固化的“执行器”。

3）没有讨论包调度问题。

4）无法对数据包进行可编程的带状态处理。

而这些问题目前被认为是因追求高性能而带来的设计折中[95]。

2.4 网络资源优化

2.4.1 软件定义网络安全通道机制

在前文提到，软件定义网络（SDN）的诸多优势是由于网络结构逻辑分层带来的。SDN 将控制平面抽离出来，形成对网络分布式数据平面的集中控制的结构。作为控制平面与数据平面交换机通信接口 OpenFlow 协议是目前最具影响力的，已经成为了业内事实标准。SDN 将网络业务抽象为网络操作系统（控制面）上的不同应用程序。如图2-10于控制平面与数据平面二者为远距传输，通信成本相对增大，主要体现在：第一，SDN 强调网络的快速变化，然而控制器对数据平面的控制都依赖于容易成为瓶颈的安全通道。安全通道一般通信速率较低，而且控制信令传输延迟大。这与快速变化的网络结构成为矛盾。第二，成为中间瓶颈的安全通道容易承载来自内部、外部的大流量，而遭受攻击。

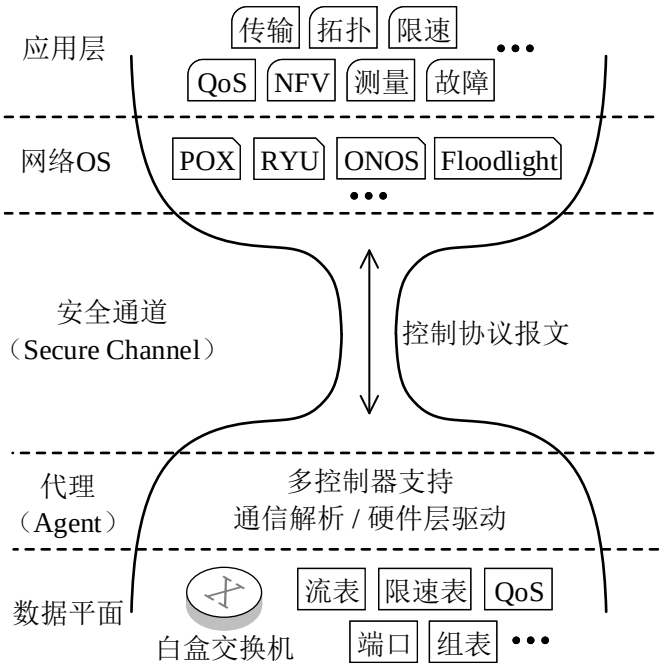


图 2-10 SDN 架构的瘦腰问题

SDN 网络中存在需要快速变化的流表项信息。SDN 的控制平面包括一个符合南向接口协议的网络操作系统，以及运行在其上的众多应用。SDN 网络操作系统与下层数

据平面通过安全通道相连。网络操作系统基本的任务是管理与配置。管理包括，发现交换机、发现拓扑、发现端口、故障测量等任务。配置包括，流表配置、执行集配置，组表以及限速表等配置。在支持数据平面可编程（PISA）的交换机中还会有包头解析器配置和数据平面逻辑块的配置。由于网络动态性强的特征，上述配置内容均可能快速发生。针对于流表配置任务，如处理新流到达，SDN 有两种策略，一种是动态响应（Reactive）。Reactive 的核心思想是被动实时处理数据平面内出现的新流量。交换机如果发现这是一条无法匹配到结果的流，那么交换机会将次信息上报控制器，控制器认证、处理后将新的流表项下发到数据平面，从而完成此流后续转发。Reactive 的缺点就是控制平面与数据平面之间信息交互频繁，对安全通道造成很大压力，若遇到瞬时流量突发（burst），还有可能会耗尽控制器的能力资源，使数据面服务中断。另一种是规划响应（Proactive）。Proactive 的核心思想是控制平面根据网络拓扑、传输任务意图，提前将所有可能出现的流量信息全都下发到控制平面。这样可以避免后续实时配置过程中的不确定性，减小安全通道遭受大流量冲击的概率。但 Proactive 的缺点也很明显，他需要大容量的硬件流表转发表来预先安装可能还用不到的流表项，另外它还与流表项更替的一般思路“最近最少使用替换”（LRU) 算法相冲突。

2.4.2 数据平面流表资源与问题

传统的表项查找方法都是基于 SRAM 的软件查找方法，共同特点是查找速度慢。线型查找法需要遍历表中的所有表项；二叉树查找法需要遍历树中大多数节点，而且查找速度受树的深度影响较大。目前最好的基于 Linux 内核的软件查找性能大约只有 1Mpps[96-97]（64 字节最小包 1Gbps 吞吐）左右⑦，是远不能满足核心路由器的处理需求的。数据平面交换机设备为支持大量流快速查找，一般需要使用基于 RAM/CAM 等硬件的快速存储器。

1）SDN 数据平面查找模型

交换机的本质工作是找到对某一数据包的处理方式，并执行这种处理。目前我们把这种处理抽象为查找和执行。查找在计算机学科内是一类最基本的问题，例如存储器就是一种典型的查找系统。总线输入数据地址（address），存储器可以返回对应地址上的数据（data）。在网络领域，输入的数据地址其实就是匹配域的值（key），返回的数据就是待处理的操作数（action）。这种过程也可以抽象为解决“key-value”的对应问题。操作数就对应着对数据包的具体执行动作，数据包在之后的流水线内可以被执行机构按操作数值进行处理。

2）各类包头域查找匹配方法

第一，基于 RAM 的精确匹配查找。上文提到由于软件查找算法性能较差，高性能路由设备内一般会使用基于硬件的快速存储器，包括 RAM/CAM/TCAM 等。RAM 是最简单的一类流表查找方法。如图2-11所示，在初始化配置流表项时，包头域的值作为地址，操作数作为内容，存储到 RAM 内。查找时先从包头提取出匹配域的值（key），

⑦ 指单个 CPU 核心，如果多核并行性能可进一步提升，但一般只能达到亚线性增长。

然后读出以 key 为地址位对应的数据即可得到操作数。一般 RAM 查找的时间复杂度只有 $O(1)$。但如果待查找的匹配域过宽，则会消耗掉一个很大 RAM 空间。例如，我们匹配 32 位的目的 IP 地址，则 RAM 表的地址总线宽度也是 32 位，如果用 1 字节（也就是可定义 256 种不同的操作）来定义操作数的话，那么总共需要的 RAM 空间是 $2^{32} \times 1Byte = 4GBytes$。由于实际的数据包中并不是每一个有可能的 IP 地址都会出现，对于不会出现的 IP 地址，我们无需对其进行设置。在一个网络中，目的 IP 地址也许不会超过 100 万个，但是 4GB 存储空间却消耗了可以存储 40 亿个 IP 地址的空间，利用率很低、比较不经济。如果希望查找位宽更大的 MAC（48bits）层 key，则需要 $2.6 \times 10^5 GB$ 内存容量，显然已经无法实现。但是对于查找一些小位宽的包头域（如，包头协议、TCP 端口号）[⑧]，则可以在内存容量消耗小于 100KB 下，实现最快速的单操作周期查找，经济适用性比较高。

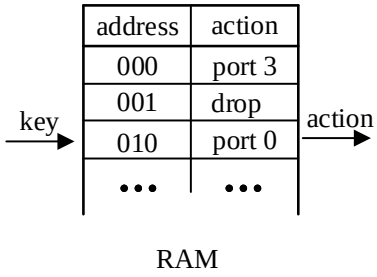


图 2-11 基于 RAM 的包头域查找过程

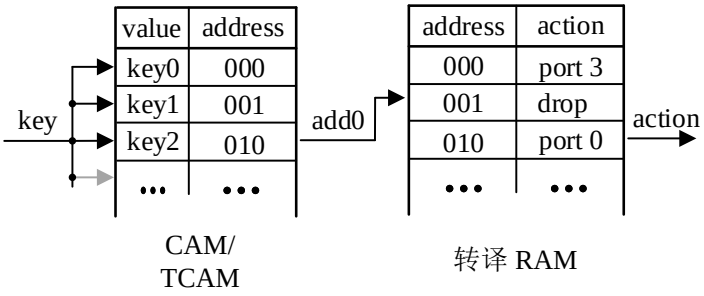


图 2-12 基于 CAM/TCAM 的包头域查找过程

第二，基于 CAM 的内容地址查找。前文提到，由于 RAM 查找法对于长匹配域无法实现资源优化，因而提出一种只跟内容数目相关的硬件查找方法（CAM）。如图2-12所示，在配置 CAM 时，将待查找 key 作为内容存储在 CAM 中，与 RAM 类似，每一个 key 都会对应到一个地址位置上。CAM 的输入内容是 key，当 CAM 接收到查找请求后，首先将 key 同步广播到每一个内容存储单元内，同时进行比较，如果与之前存储值相同，则会返回匹配成功，同时返回此单元内数值所对应的地址位置（addr0）。这一步操作的时间复杂度也是 $O(1)$。每一个 key 与地址位置等价，但地址位置并不能够代表操作数含义，因而在 CAM 后面会跟随一个“地址—操作数”转译 RAM 表。在此 RAM 表中，我们需要提前在 RAM 的 addr0 地址中存储一个操作数（action0）。所以当 CAM 查到 key 的地址 addr0 后，再由 RAM 查到 addr0 的内容 action0，两步查找的总时间复杂度还是 $O(1)$。由于 CAM 架构只需保存用户所需数目的 key，因而 CAM 可以将存储器空间资源占用率压缩到线性。CAM 在是广播查找时电路并行度很高，所以大量的布线资源比较耗费芯片逻辑空间。一次查找会引发全部内容比较，因而芯片耗电量也会增加。所以 CAM 查找表一般容量在数万条流表。

第三，基于 TCAM 的三态内容地址查找。与 CAM 类似，都是讲 key 存到芯片存储单元内。TCAM 可以支持任意位的掩码查找，也就是说可以在匹配时对某些 bit 为设置“不关心”状态，所有不关心 bits 都可以认为是匹配成功，TCAM 匹配的时间复杂度也

[⑧] 包头协议（8bits）对应内存容量空间 256Bytes，端口号（16bits）对应内存容量空间 64KBytes。

是 $O(1)$。这种架构的优势是可以支持匹配某一 IP 范围的全部匹配。假设在表项中设置了 Nbits 的无关位，由于无关位可以是任意值，所以总共满足可匹配的 key 数量是 2^N 个。因而 TCAM 理论上可以覆盖比 CAM 更多的 key 数目。TCAM 在实现最长前缀匹配，流量汇合等功能时具有无法替代的价值。但 TCAM 与 CAM 相比，在每个存储单元内增加了更多的逻辑数量，因而 TCAM 的造价和能源消耗也比相同表项数目的 CAM 更高。

3）SDN 流表面临问题

在软件定义网络时代，为了适配各种新的包头域，交换机内所需流表的宽度快速增长。相比较于传统 L2/L3 网络，软件定义网络所定义的包头域宽度是其数十倍。在流表容量不变的前提下，包头域宽度的增加就意味着深度减小。另外如上文所述，基于硬件的快速查表方案都存在硬件资源消耗大的问题。流表资源数目不足可直接引发诸多网络问题，例如，1）数据平面内无服务，2）因快速更替流表项内容导致安全通道内报文数量激增，3）增大流表容量使设备价格上涨经济效益变差。值得注意的是，第二点问题会耗费大量控制器计算资源，降低安全通道通信带宽，从而进一步产生影响全网络安全的问题。

如何缓解这类问题成为当前研究的重点内容。CompactTCAM[98] 提出一种宽、窄包头域的等效替换方式，压缩了已占用流表容量[99]，从而减小流表宽度的需求。但由于更改了其他公共包头域的功能，但这个机制适用于内网，无法直接用于大容量需求的骨干网广域和数据中心网络。工作 uFlow[100] 发现网络中小流可以经由控制器直接转发到目的地，从而不占用网络内流表存储。但其实大流的溢出才是最危险的，而且控制平面与数据平面混合会导致控制平面遭受 DDoS 攻击。并且现有工作都是从减小开销的角度去解决问题，但是本文后面证明，流表溢出在现有网络里是必然发生的，目前工作很难缓解当流表真的发生溢出后带来的网络安全危害。如何能够利用 SDN 全局视野以及目前新兴的可编程数据平面是本文研究的重点。

2.5 本章小结

随着网络数据传输需求日益增强、SDN 网络架构和可编程数据平面的提出以及数据中心虚拟化规模逐步扩大，本文发现制约网络性能的关键因素分别在网络的不同层面上：1）主机侧网络。CPU 已经成为主机网络通信速率的瓶颈。2）交换网络。目前的可编程数据平面依然有着灵活性与性能之间的矛盾。3）网络资源。作为高性能网络内设备核心资源的流表因造价高、容量小导致网络内极易产生流表溢出等现象。

学术界和产业界为解决上述问题均提出了各种新设备架构和新思想。经过分析网络发展的不同阶段历史规律，本文力图依靠解决如何利用科学严谨的思想设计新架构、如何根据现有技术做取舍、如何集中发挥目前技术某方面优势，来满足业界新的需求。在确保高性能的同时、兼顾网络稳定性提高安全保障。

3 研究可编程设备加速主机侧网络方法

3.1 本章引论

3.2 问题背景

随着数据中心服务器网络接口容量快速增长，处理与网络数据相关的服务已越来越耗费主机 CPU 的计算资源。主机虚拟化、流量工程、网络监管等功能在现代化网络管理中已经占据重要的位置。尽管目前发展出一系列 RDMA、DPDK 等基于网卡的网络数据包快速搬运架构，但它们只针对于块儿数据的传递进行了优化，对于需要“每包处理”的任务依然没有良好的对策。

本文将复杂网络计算问题分为两类任务类型，一种类型是由于数据包到达频繁而触发的大量计算，也许这类型计算并不复杂（处理器使用几条指令即可完成计算），但由于需要对每个数据包都进行处理从而导致计算量庞大、CPU 无法胜任，例如查找、转发、分类等。第二种类型是由于计算过程复杂（处理器需要耗费多条指令才可完成计算），导致 CPU 无法提升针对每个数据包处理的速度，进而造成网络分发数据包个数降低、性能需求无法胜任，例如防火墙、安全分析等。数据中心运营商面对着不断增长的功能需求和性能需求，同时也面对着需要降低运营成本提升能源利用率和绿色环保。

网络随路计算（in-network computing）是一种解决网络程序性能差的有效途径。网络随路计算是指在网络数据传输链路中增加特定功能的硬件设备，使得数据包在传送到主机内部之前，就完成了网络任务中的相关计算需求。当计算从主机内，下方到了网络内后，这便释放了软件处理瓶颈，可以节约数据中心内宝贵的 CPU 处理资源。但主机内的计算任务并不能够全都被硬件卸载，因而我们需要选取可编程数据平面来完成这件事。前文提到过三类可编程网卡他们分别是：基于可编程 ASIC 的智能网卡，基于 NPU 的智能网卡，以及基于 FPGA 的智能网卡。首先本文分析基于 ASIC 的智能网卡是非“图灵完全”的可编程硬件，尽管其处理性能优异，但无法支持灵活配置，因而本文不考虑。其次基于 NPU 的智能网卡，NPU 由众核处理器架构组成，可提升处理并行度，但这是基于批处理的计算模型。针对目前流式计算和有状态计算，因 NPU 每核处理性能低，从而导致 NPU 方案整体运行性能差。

本文从上述两类问题中各选取了一个应用场景，来说明使用基于 FPGA 的智能网卡具备强大的“每包处理”能力，以及强大的“复杂计算”能力。最重要的是 FPGA 可以支持网络内流式计算模型，本文在后面提出了一种在 FPGA 针对流式计算需求的 DC 抽象方法，可以将有前后状态依赖的“复杂计算”任务卸载到基于 FPGA 的可编程硬件。

3.3 系统架构

3.3.1 软件向硬件卸载分析

CPU 通过循环取指令等操作，完成通用的可计算任务。计算的数据通常有前后依赖关系，CPU 对于此类计算效率较低：由于中间计算状态在计算完成之后必须放回数据存储区，而下次重新拿回状态数据又需要再次搬运，数据在计算核心与存储池之间多次往返对计算最终结果是无意义操作。而即使使用众核处理器也无法优化此过程，虽然众核处理器可并行计算多个任务，但同一时刻每核心与其他核心处理内容并无关联。对于前后有依赖关系的处理，并行并不能加速其中一组数据的处理进程。在网络领域亦是如此，网络数据包到达密度很大，留给每个包的处理时间很有限，然而通常一个 CPU 无法在如此短时间内真正处理完一个包的触发计算。

1）软件处理延迟大。

目前高速网络处理器可以设计为核间流水线模式，每个核只处理固定的一步计算。当一个数据包触发的计算包括多个前后依赖计算时，人们使用多个核串行处理这组计算。即每个核心领取一个固定的快速处理任务，这样每个核都可以以最快的速度处理完当前步骤，然后交给后续核心继续处理。这样数据包的处理吞吐其实就可以达到某一个核心的最大速度，而处理延迟则是这些核心之间传递完整一次的时间。不难发现，数据在众核之间搬运会遇到访存时间过长、访存请求冲突等现象。虽然使用 CPU 可以获取很大的灵活性，但这种方式进一步增大了每个数据包处理的时延。如今在云端加速 AI 计算的场景下，高频次小包、数据快速到达的需求越来越高，这使得以 CPU 处理网络数据包造成很大的云计算性能瓶颈。后续我们通过本文所举的例子可以明显提现这一点。

2）软件处理时间精度低。

同时，由于数据在核心、存储池之间搬运会造成不定时的请求冲突、甚至计算等待，这将打乱顺序串行处理节拍。由于这条处理链中的处理速度，收到这条处理链最慢处的制约，因而处理链的性能表现总以最慢的瓶颈向外表现。软件的处理收到操作系统的指挥，操作系统一般会划分时间片区来分配给每一个待处理的任务。虚拟化的操作系统内有很多任务进程，操作系统会划分很多时间片区。这也进一步降低了 CPU 的专用任务处理性能，还带来处理时间精度不足的现象。后续我们通过本文所举的例子可以明显提现这一点。

3）软件卸载。

3.3.2 软件算法的硬件抽象方法

硬件编码思想

3.4 流量工程—网络流量捕获与回放

3.4.1 设计

1Gbps 硬件设计思路

3.4.2 优化

向 100G 出发

3.5 统计—网络测量实时压缩

3.5.1 设计

1）压缩算法
2）压缩算法硬件实现
3）效果与问题

3.5.2 优化

1）符合 DC 抽象的算法优化方法
2）无偏估计证明

3.6 软硬一体化的系统实验平台

3.6.1 软件

3.6.2 硬件

3.7 性能评估

时间精度
吞吐
估算
cacti 比较
在主机端，网络与计算的需求，

在第二章中提到 NPU 的可编程性最强且可以使用软件控制，由于他单核性能问题，尽管需要使程序员更换硬件思维，但业界普遍已经采取 FPGA 的方式来加速网络计算。

本章的主体是网络功能使用可编程硬件加速。

主机端网络开销最大的还属流量工程和网络内计算，因为每包处理，目前百 G NIC 包速率已经达到 150Mpps, 对于每包操作的计算任务，CPU 负载很大

1）分析软件到硬件卸载的可行性，背景

软件可扩展分析，

4 研究可编程设备加速网络硬件交换层方法

5 SDN 硬件流表可扩展性研究

6 参考文献格式

参考文献格式应符合国家标准 GB/T-7714-2005《文后参考文献著录规则》。中国国家标准化管理委员会于 2015 年 5 月 15 日发布了新的标准 GB/T 7714-2015《信息与文献参考文献著录规则》。因为二者的差别非常小，所以采用了新的标准。标准的 BiBTeX 格式网上资源非常多，本模板使用了李泽平开发的版本，该版本提供了多种参考文献的排序规则。学校博士论文规范指定了两种排序方法：一是按照文献的引用顺序进行排序，二是按照作者姓氏加出版年份进行排序。本模板采用第一种排序规则，第二种排序规则的使用方法请参考文献 [101]。

参考文献格式应符合国家标准 GB/T-7714-2005《文后参考文献著录规则》。中国国家标准化管理委员会于 2015 年 5 月 15 日发布了新的标准 GB/T 7714-2015《信息与文献参考文献著录规则》。因为二者的差别非常小，所以采用了新的标准。标准的 BiBTeX 格式网上资源非常多，本模板使用了李泽平开发的版本，该版本提供了多种参考文献的排序规则。学校博士论文规范指定了两种排序方法：一是按照文献的引用顺序进行排序，二是按照作者姓氏加出版年份进行排序。本模板采用第一种排序规则，第二种排序规则的使用方法请参考文献 [101]。

虽然本模板不讲解 LaTeX 的详细使用方法，但是为了方便大家使用本模板撰写论文，本章对论文写作中经常用到的 **图、表、公式**等内容的排版方法做一个简单介绍。

6.1 图

6.1.1 单幅图

图 6-1 是用 TeXLive 自带的宏包 Tikz 绘制而成，Visio 画不出这么好看的图。

6.1.2 多幅图

如果一幅图中包含多幅子图，每一幅子图都要有图注，并且子图用 (a)、(b)、(c) 等方式编号，如图 6-2 所示。

6.2 表

表格要求采用三线表，与文字齐宽，顶线与底线线粗是 $1\frac{1}{2}$ 磅，中线线粗是 1 磅，如表 6-1 所示[①]。

[①] 注意：图表中的变量与单位通过斜线 / 隔开。

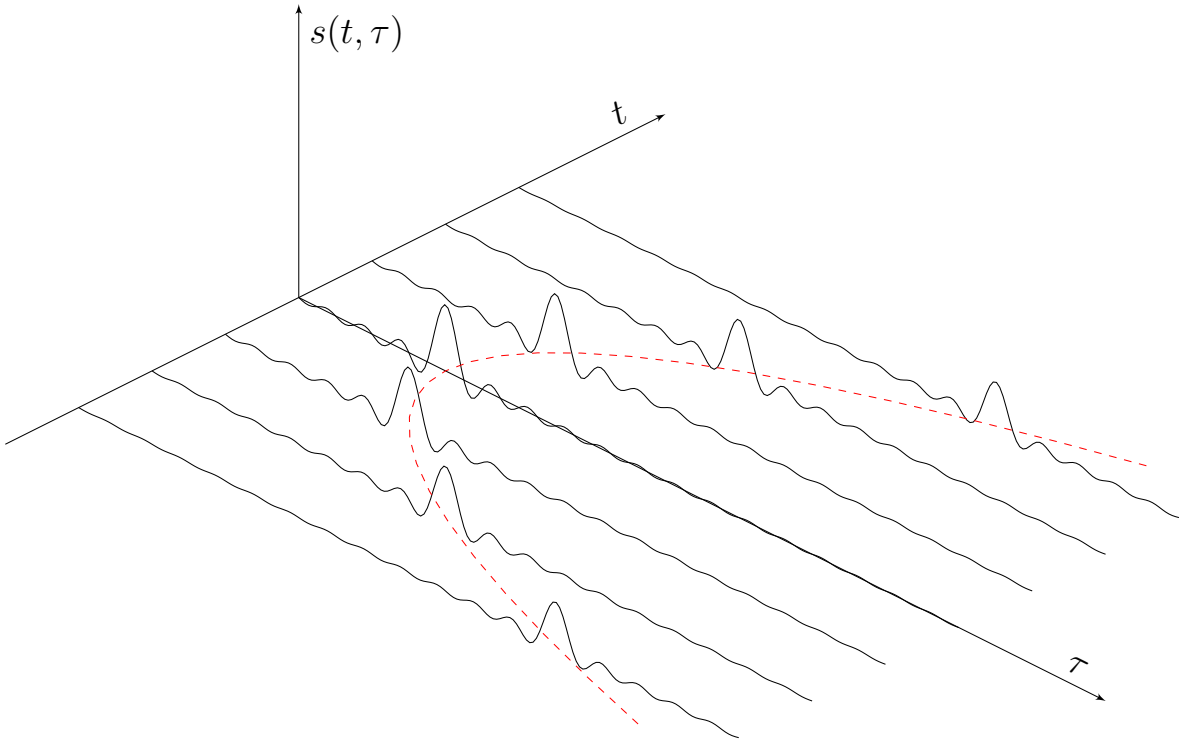
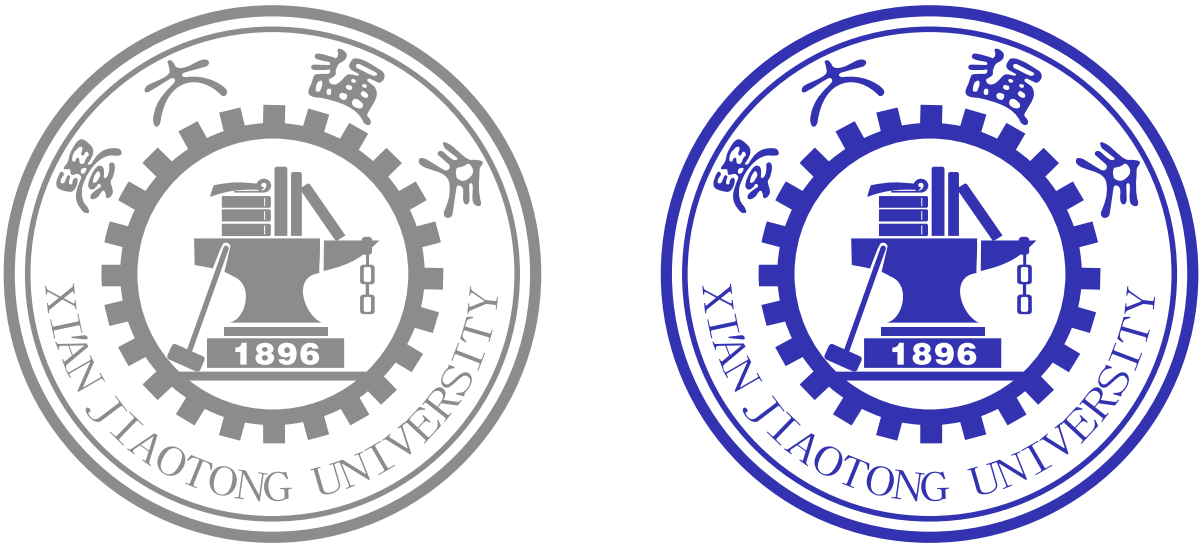


图 6-1 雷达回波信号 (注意：图注是五号字)。



(a) 灰色的交大校徽　　(b) 蓝色的交大校徽

图 6-2 交大校徽

6.3 公式

6.3.1 单个公式

LaTeX 最强大的地方在于对数学公式的编辑，不仅美观，而且高效。单个公式的编号如式 (6-1) 所示，该式是正态分布的概率密度函数[?]，

表 6-1 表题也是五号字

Interference	DOA / degree	Bandwidth / MHz	INR / dB
1	-30	20	60
2	20	10	50
3	40	5	40

$$f_Z(z) = \frac{1}{\pi\sigma^2}\exp\left(-\frac{|z-\mu|^2}{\sigma^2}\right) \tag{6-1}$$

式中：μ 是 Gauss 随机变量 Z 的均值；σ^2 是 Z 的方差。

6.3.2 多个公式

多个公式作为一个整体可以进行二级编号，如式 (6-2) 所示，该式是连续时间 Fourier 变换的正反变换公式[?]，

$$X(f) = \int_{-\infty}^{\infty} x(t)e^{-j2\pi ft}\mathrm{d}t \tag{6-2a}$$

$$x(t) = \int_{-\infty}^{\infty} X(f)e^{j2\pi ft}\mathrm{d}f \tag{6-2b}$$

式中：$x(t)$ 是信号的时域波形；$X(f)$ 是 $x(t)$ 的 Fourier 变换。

如果公式中包含推导步骤，可以只对最终的公式进行编号，例如：

$$\begin{aligned}\mathbf{w}_{\mathrm{smi}} &= \alpha\left[\frac{1}{\sigma_n^2}\mathbf{v}(\theta_0) - \frac{1}{\sigma_n^2}\mathbf{v}(\theta_0) + \sum_{i=1}^{N}\frac{\mathbf{u}_i^H\mathbf{v}(\theta_0)}{\lambda_i}\mathbf{u}_i\right] \\ &= \frac{\alpha}{\sigma_n^2}\left[\mathbf{v}(\theta_0) - \sum_{i=1}^{N}\mathbf{u}_i^H\mathbf{v}(\theta_0)\mathbf{u}_i + \sum_{i=1}^{N}\frac{\sigma_n^2\mathbf{u}_i^H\mathbf{v}(\theta_0)}{\lambda_i}\mathbf{u}_i\right] \\ &= \frac{\alpha}{\sigma_n^2}\left[\mathbf{v}(\theta_0) - \sum_{i=1}^{N}\frac{\lambda_i-\sigma_n^2}{\lambda_i}\mathbf{u}_i^H\mathbf{v}(\theta_0)\mathbf{u}_i\right]\end{aligned} \tag{6-3}$$

致　谢

致谢中主要感谢导师和对论文工作有直接贡献和帮助的人士和单位。致谢言语应谦虚诚恳，实事求是，字数不超过 1000 汉字。

用于盲审的论文，此页内容全部隐去。

参考文献

[1] 华为公司年报[EB/OL]. 2019. https://www-file.huawei.com/-/media/corporate/pdf/annual-report/annual_report_2019_cn.pdf.

[2] 思科公司互联网发展跟踪白皮书（2018-2023）[EB/OL]. 2019. https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html.

[3] 国际数据公司（IDC）. 上半年中国公有云市场[EB/OL]. 2019. https://www.idc.com/url.do?url=/getdoc/pdf_download.do?containerId=prCHC45634819&position=15&transactionId=39032154&term=&page=5&perPage=100.

[4] IREASEARCH. 中国公有云服务市场跟踪[EB/OL]. 2020. http://news.iresearch.cn/yx/2020/02/315730.shtml.

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

[6] BOSSHART P, GIBB G, KIM H S, et al. Forwarding metamorphosis: Fast programmable match-action processing in hardware for sdn[J]. ACM SIGCOMM Computer Communication Review, 2013, 43(4): 99-110.

[7] HONDA M, HUICI F, LETTIERI G, et al. mswitch: a highly-scalable, modular software switch[C]// Proceedings of the 1st ACM SIGCOMM Symposium on Software Defined Networking Research. [S.l.: s.n.], 2015: 1-13.

[8] MCKEOWN N, ANDERSON T, BALAKRISHNAN H, et al. Openflow: enabling innovation in campus networks[J]. ACM SIGCOMM Computer Communication Review, 2008, 38(2): 69-74.

[9] CASADO M, MCKEOWN N, SHENKER S. From ethane to sdn and beyond[J]. ACM SIGCOMM Computer Communication Review, 2019, 49(5): 92-95.

[10] SHAHBAZ M, CHOI S, PFAFF B, et al. Pisces: A programmable, protocol-independent software switch[C]//Proceedings of the 2016 ACM SIGCOMM Conference. [S.l.: s.n.], 2016: 525-538.

[11] HUAWEI. 1800V 虚拟交换机[EB/OL]. 2018. https://carrier.huawei.com/~/media/CNBG/Downloads/Product/Fixed%20Network/b2b/0920/1800-en.pdf.

[12] SANDVINE. Hyperscale data plane for next generation telco networks[EB/OL]. 2020. https://www.sandvine.com/hubfs/Sandvine_Redesign_2019/Downloads/2020/Datasheets/Network%20Optimization/Sandvine_DS_ActiveLogic.pdf.

[13] CENTEC. Hybrid v580 sdn switch[EB/OL]. 2019. http://www.centecnetworks.com/cn/DownView.asp?ID=2272&SortID=153.

[14] 高山渊, 蔡德忠, 赵晓雪, 等. 企业数字化基石-阿里巴巴云计算基础设施实践[M]. 北京市海淀区: 电子工业出版社, 2020.

[15] BAREFOOT. Second generation of world's fastest p4 programmable ethernet switch asics[EB/OL]. 2020. https://www.barefootnetworks.com/products/brief-tofino-2/.

[16] LU G, GUO C, LI Y, et al. Serverswitch: a programmable and high performance platform for data center networks.[C]//Nsdi: volume 11. [S.l.: s.n.], 2011: 2-2.

[17] FIRESTONE D, PUTNAM A, MUNDKUR S, et al. Azure accelerated networking: Smartnics in the public cloud[C]//15th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 18). [S.l.: s.n.], 2018: 51-66.

[18] 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.
[19] XILINX CO. L. [EB/OL]. 2020. https://www.xilinx.com/about/company-overview.html.
[20] XILINX. Smartnics for diverse workloads[EB/OL]. 2020. https://www.xilinx.com/applications/data-center/network-acceleration.html#smartnics.
[21] INTEL. Fpga programmable acceleration card n3000 for networking[EB/OL]. 2020. https://plan.seek.intel.com/psg_WW_psgcom3_LPCS_EN_2019_PACN3000ProductBrief.
[22] INTEL. 5G 前传边缘网络 FPGA（IP）方案[EB/OL]. 2020. https://plan.seek.intel.com/5GFrontHaulGatedFormCN_LP?erpm_id=8235613&erpm_id=8235613&elq_cid=6511651.
[23] XILINX. Stand alone nvme-of acceleration solution[EB/OL]. 2020. https://www.xilinx.com/publications/solution-briefs/partner/nvme-of_solutionbrief.pdf.
[24] INTEL. SDN/NFV 低延迟 GRE 处理加速器[EB/OL]. 2020. https://www.intel.cn/content/dam/altera-www/global/zh_CN/pdfs/literature/wp/low-latency-gre-processing-accelerator-evaluation-cn.pdf.
[25] INTEL. 电信解决方案 FPGA PAC N3000 助力在云环境中实现大容量 DDoS 防护[EB/OL]. 2020. https://www.intel.cn/content/dam/www/programmable/cn/zh/pdfs/literature/solution-sheets/sb-high-capacity-ddos-protection-in-cloud-environments-cn.pdf.
[26] INTEL. 英特尔 FPGA 可编程加速卡 N3000 的 IPSec 加速解决方案[EB/OL]. 2020. https://www.intel.cn/content/dam/altera-www/global/zh_CN/pdfs/literature/solution-sheets/sb-accelerating-ipsec-arrive-technology-intel-fgpa-pac3000-cn.pdf.
[27] XILINX. Vivado high-level synthesis accelerates ip creation by enabling c/c++ and system c specifications[EB/OL]. 2020. https://www.xilinx.com/products/design-tools/vivado/integration/esl-design.html.
[28] XILINX. Vivado hls documentation[EB/OL]. 2020. https://www.xilinx.com/products/design-tools/vivado/integration/esl-design.html#documentation.
[29] INTEL. Data plane development kit[EB/OL]. 2020. https://www.dpdk.org/.
[30] VMWARE. Single root i/o virtualization(sr-iov)[EB/OL]. 2019. https://docs.vmware.com/en/VMware-vSphere/7.0/com.vmware.vsphere.networking.doc/GUID-CC021803-30EA-444D-BCBE-618E0D836B9F.html.
[31] MELLANOX. Remote direct memory access(rdma)[EB/OL]. 2019. https://community.mellanox.com/s/global-search/rdma.
[32] MELLANOX. Rdma over converged ethernet(roce)[EB/OL]. 2020. https://docs.mellanox.com/pages/viewpage.action?pageId=19811943.
[33] MICROSOFT. Information about the tcp chimney offload, receive side scaling, and network direct memory access features[EB/OL]. 2008. https://support.microsoft.com/en-us/help/951037/information-about-the-tcp-chimney-offload-receive-side-scaling-and-net.
[34] XILINX. Alveo u250 data center accelerator card[EB/OL]. 2020. https://www.xilinx.com/products/boards-and-kits/alveo/u250.html.
[35] NETFPGA. A line-rate, flexible, and open platform for research, and classroom experimentation. [EB/OL]. 2020. https://netfpga.org/site/#/about/.
[36] 中国工业信息化部. 2012 年使用 4M 宽带的用户将过半[EB/OL]. 2011. http://www.gov.cn/jrzg/2012-04/01/content_2104826.htm.
[37] MCKEOWN N. A fast switched backplane for a gigabit switched router[J]. Business Communications Review, 1997, 27(12): 1-30.

[38] KATEVENIS M, PASSAS G, SIMOS D, et al. Variable packet size buffered crossbar (cicq) switches [C]//2004 IEEE International Conference on Communications (IEEE Cat. No. 04CH37577): volume 2. [S.l.]: IEEE, 2004: 1090-1096.

[39] AYBAY G. Method and apparatus for forwarding variable-length packets between channel-specific packet processors and a crossbar of a multiport switch[M]. [S.l.]: Google Patents, 2000.

[40] NACHIONDO T, FLICH J, DUATO J. Buffer management strategies to reduce hol blocking[J]. IEEE transactions on parallel and distributed systems, 2009, 21(6): 739-753.

[41] CISCO. Cisco 12000 series routers[EB/OL]. 2006. https://www.cisco.com/c/en/us/products/routers/12000-series-routers/index.html.

[42] YOSHIGOE K, CHRISTENSEN K J. A parallel-polled virtual output queued switch with a buffered crossbar[C]//2001 IEEE Workshop on High Performance Switching and Routing (IEEE Cat. No. 01TH8552). [S.l.]: IEEE, 2001: 271-275.

[43] HEITNER M L, SONG J J, VIANNA R. Folded clos architecture switching[M]. [S.l.]: Google Patents, 2004.

[44] BROADCOM. Tomahawk 4 industry’s highest bandwidth ethernet switch chip at 25.6tbps[EB/OL]. 2019. https://www.globenewswire.com/news-release/2019/12/09/1958047/0/en/Broadcom-Ships-Tomahawk-4-Industry-s-Highest-Bandwidth-Ethernet-Switch-Chip-at-25-6-Terabits-per-Second.html.

[45] CASADO M, FREEDMAN M J, PETTIT J, et al. Ethane: Taking control of the enterprise[J]. ACM SIGCOMM computer communication review, 2007, 37(4): 1-12.

[46] AL-FARES M, RADHAKRISHNAN S, RAGHAVAN B, et al. Hedera: dynamic flow scheduling for data center networks.[C]//Nsdi: volume 10. [S.l.: s.n.], 2010: 89-92.

[47] HELLER B, SEETHARAMAN S, MAHADEVAN P, et al. Elastictree: Saving energy in data center networks.[C]//Nsdi: volume 10. [S.l.: s.n.], 2010: 249-264.

[48] ONF/TS-022. Optical transport protocol extensions[EB/OL]. 2015. https://3vf60mmveq1g8vzn48q2o71a-wpengine.netdna-ssl.com/wp-content/uploads/2014/10/Optical_Transport_Protocol_Extensions_V1.0.pdf.

[49] JAIN S, KUMAR A, MANDAL S, et al. B4: Experience with a globally-deployed software defined wan[J]. ACM SIGCOMM Computer Communication Review, 2013, 43(4): 3-14.

[50] ARYAKA. Managed sd-wan for digital transformation[EB/OL]. 2017. https://www.aryaka.com/aryaka-sd-wan-solutions-for-manufacturing/.

[51] ONF/TS-029. Mpls-tp openflow protocol extensions for sptn[EB/OL]. 2017. https://3vf60mmveq1g8vzn48q2o71a-wpengine.netdna-ssl.com/wp-content/uploads/2017/07/MPLS-TP-OpenFlow-Protocol-Extensions-for-SPTN-1-0.pdf.

[52] DE CARLI L, PAN Y, KUMAR A, et al. Plug: flexible lookup modules for rapid deployment of new protocols in high-speed routers[C]//Proceedings of the ACM SIGCOMM 2009 conference on Data communication. [S.l.: s.n.], 2009: 207-218.

[53] ANWER M B, MOTIWALA M, TARIQ M B, et al. Switchblade: a platform for rapid deployment of network protocols on programmable hardware[C]//Proceedings of the ACM SIGCOMM 2010 conference. [S.l.: s.n.], 2010: 183-194.

[54] INTEL. Intel ethernet switch fm6000 series[EB/OL]. 2013. https://www.intel.com/content/dam/www/public/us/en/documents/product-briefs/ethernet-switch-fm6000-series-brief.pdf.

[55] BAREFOOT. The world’s fastest p4 programmable ethernet switch asics[EB/OL]. 2017. https://barefootnetworks.com/products/brief-tofino/.

[56] NAOUS J, ERICKSON D, COVINGTON G A, et al. Implementing an openflow switch on the netfpga platform[C]//Proceedings of the 4th ACM/IEEE Symposium on Architectures for Networking and Communications Systems. [S.l.: s.n.], 2008: 1-9.

[57] YABE T. Openflow implementation on netfpga-10g design document[M]. [S.l.]: Stanford University, 2011.

[58] HAN J H, MUNDKUR P, ROTSOS C, et al. Blueswitch: Enabling provably consistent configuration of network switches[C]//2015 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS). [S.l.]: IEEE, 2015: 17-27.

[59] LI B, 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. [S.l.: s.n.], 2016: 1-14.

[60] WANG H, SOULÉ R, DANG H T, et al. P4fpga: A rapid prototyping framework for p4[C]//Proceedings of the Symposium on SDN Research. [S.l.: s.n.], 2017: 122-135.

[61] XILINX. Packet processor smartcore[EB/OL]. 2017. https://www.xilinx.com/support/documentation-navigation/development-tools/software-development/sdnet.html.

[62] FIRESTONE D, PUTNAM A, MUNDKUR S, et al. Azure accelerated networking: Smartnics in the public cloud[C]//15th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 18). [S.l.: s.n.], 2018: 51-66.

[63] INTEL. Ixp4xx product line of network processors[EB/OL]. 2010. https://www.intel.com/content/dam/www/public/us/en/documents/specification-updates/ixp4xx-product-line-network-processors-spec-update.pdf.

[64] OPENSWITCH. Cavium-xpliant® family of programmable ethernet switches[EB/OL]. 2010. https://www.openswitch.net/cavium/.

[65] NETRONOME. Agilio cx 2x40gbe intelligent server adapter[EB/OL]. 2016. http://colfaxdirect.com/store/pc/catalog/Agilio-CX-2x40GbE.pdf.

[66] PAGIAMTZIS K, SHEIKHOLESLAMI A. Content-addressable memory (cam) circuits and architectures: A tutorial and survey[J]. IEEE journal of solid-state circuits, 2006, 41(3): 712-727.

[67] FELDMAN A, MUTHUKRISHNAN S. Tradeoffs for packet classification[C]//Proceedings IEEE INFOCOM 2000. Conference on computer communications. Nineteenth annual joint conference of the IEEE computer and communications societies (Cat. No. 00CH37064): volume 3. [S.l.]: IEEE, 2000: 1193-1202.

[68] KOGAN K, NIKOLENKO S, ROTTENSTREICH O, et al. Sax-pac (scalable and expressive packet classification)[C]//Proceedings of the 2014 ACM conference on SIGCOMM. [S.l.: s.n.], 2014: 15-26.

[69] SRINIVASAN V, SURI S, VARGHESE G. Packet classification using tuple space search[C]//Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication. [S.l.: s.n.], 1999: 135-146.

[70] GREENHALGH A, HUICI F, HOERDT M, et al. Flow processing and the rise of commodity network hardware[J]. ACM SIGCOMM Computer Communication Review, 2009, 39(2): 20-26.

[71] FLOWFORWARDING. Link is not closed[EB/OL]. 2013. https://flowforwarding.github.io/LINC-Switch/.

[72] FLOODLIGHT. Open source project to support openflow on a range of physical and now virtual switch platforms[EB/OL]. 2013. https://github.com/floodlight/indigo.

[73] IN BRAZIL E I C. Basic openflow software switch[EB/OL]. 2013. https://cpqd.github.io/ofsoftswitch13/.

[74] SNABB. Snabb switch:a simple and fast packet networking toolkit[EB/OL]. 2015. https://github.com/snabbco/snabb.

[75] PFAFF B, PETTIT J, KOPONEN T, et al. The design and implementation of open vswitch[C]//12th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 15). [S.l.: s.n.], 2015: 117-130.

[76] MOLNÁR L, PONGRÁCZ G, ENYEDI G, et al. Dataplane specialization for high-performance openflow software switching[C]//Proceedings of the 2016 ACM SIGCOMM Conference. [S.l.: s.n.], 2016: 539-552.

[77] PANDA A, HAN S, JANG K, et al. Netbricks: Taking the v out of {NFV}[C]//12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16). [S.l.: s.n.], 2016: 203-216.

[78] CONSORTIUM P L, et al. Behavioral model (bmv2)[J]. URL: https://github. com/p4lang/behavioral-model [cited 2020-01-21], 2018.

[79] AZURE M. Open source network operating system[EB/OL]. 2016. https://azure.github.io/SONiC/.

[80] DALTON M, SCHULTZ D, ADRIAENS J, et al. Andromeda: Performance, isolation, and velocity at scale in cloud network virtualization[C]//15th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 18). [S.l.: s.n.], 2018: 373-387.

[81] SHINDE P, KAUFMANN A, ROSCOE T, et al. We need to talk about nics[C]//Presented as part of the 14th Workshop on Hot Topics in Operating Systems. [S.l.: s.n.], 2013.

[82] COSTA P, DONNELLY A, ROWSTRON A, et al. Camdoop: Exploiting in-network aggregation for big data applications[C]//Presented as part of the 9th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 12). [S.l.: s.n.], 2012: 29-42.

[83] SAPIO A, ABDELAZIZ I, ALDILAIJAN A, et al. In-network computation is a dumb idea whose time has come[C]//Proceedings of the 16th ACM Workshop on Hot Topics in Networks. [S.l.: s.n.], 2017: 150-156.

[84] MAI L, RUPPRECHT L, ALIM A, et al. Netagg: Using middleboxes for application-specific on-path aggregation in data centres[C]//Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies. [S.l.: s.n.], 2014: 249-262.

[85] GRAHAM R L, BUREDDY D, LUI P, et al. Scalable hierarchical aggregation protocol (sharp): a hardware architecture for efficient data reduction[C]//2016 First International Workshop on Communication Optimizations in HPC (COMHPC). [S.l.]: IEEE, 2016: 1-10.

[86] LIU M, LUO L, NELSON J, et al. Incbricks: Toward in-network computation with an in-network cache[C]//Proceedings of the Twenty-Second International Conference on Architectural Support for Programming Languages and Operating Systems. [S.l.: s.n.], 2017: 795-809.

[87] SANVITO D, SIRACUSANO G, BIFULCO R. Can the network be the ai accelerator?[C]// Proceedings of the 2018 Morning Workshop on In-Network Computing. [S.l.: s.n.], 2018: 20-25.

[88] SIRACUSANO G, BIFULCO R. In-network neural networks[EB/OL]. 2018. https://arxiv.org/pdf/1801.05731.pdf.

[89] JOUPPI N P, YOUNG C. In-datacenter performance analysis of a tensor processing unit[EB/OL]. 2017. https://arxiv.org/ftp/arxiv/papers/1704/1704.04760.pdf.

[90] MIAO R, ZENG H, KIM C, et al. Silkroad: Making stateful layer-4 load balancing fast and cheap using switching asics[C]//Proceedings of the Conference of the ACM Special Interest Group on Data Communication. [S.l.: s.n.], 2017: 15-28.

[91] LAPOLLI Â C, MARQUES J A, GASPARY L P. Offloading real-time ddos attack detection to programmable data planes[C]//2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM). [S.l.]: IEEE, 2019: 19-27.

[92] YANG T, JIANG J, LIU P, et al. Elastic sketch: Adaptive and fast network-wide measurements[C]// Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication. [S.l.: s.n.], 2018: 561-575.

[93] KIM C, SIVARAMAN A, KATTA N, et al. In-band network telemetry via programmable dataplanes [C]//ACM SIGCOMM. [S.l.: s.n.], 2015.

[94] JIN X, LI X, ZHANG H, et al. Netcache: Balancing key-value stores with fast in-network caching [C]//Proceedings of the 26th Symposium on Operating Systems Principles. [S.l.: s.n.], 2017: 121-136.

[95] SCIENCE N C. Rmt and p4 notes[EB/OL]. 2018. https://cs.nyu.edu/~anirudh/CSCI-GA.2620-001 /lectures/lec8_rmt_p4.txt.

[96] TOONK A. Linux kernel and measuring network throughput.[EB/OL]. 2020. https://medium.com/devops-dudes/linux-kernel-and-measuring-network-throughput-547c3b68c4d2.

[97] BERNAT V. Performance progression of ipv4 route lookup on linux[EB/OL]. 2017. https://vincent.bernat.ch/en/blog/2017-performance-progression-ipv4-route-lookup-linux.

[98] KANNAN K, BANERJEE S. Compact tcam: Flow entry compaction in tcam for power aware sdn [C]//International conference on distributed computing and networking. [S.l.]: Springer, 2013: 439-444.

[99] 周亚东, 陈凯悦, 冷俊园, 等. 软件定义网络流表溢出脆弱性分析及防御方法[J]. 西安交通大学学报, 2017(10): 53-58.

[100] 郑鹏, 胡成臣, 李昊. 基于流量特征的 OpenFlow 南向接口开销优化技术[J]. 计算机研究与发展, 2018, 55(2): 346-357.

[101] LEE Z. GB/T 7714-2015 参考文献 BiBTeX 样式[EB/OL]. 2016. https://github.com/zepinglee/gbt7714-bibtex-style.

附录 A 公式定理证明

附录编号依次编为附录 A，附录 B。附录标题各按一级标题编排。附录中的图、表、公式另行编排序号，编号前加“附录 A-”字样。这部分内容非强制性要求，如果论文中没有附录，可以省略。

排版数学定理等环境时最好给环境添加结束符，以明确定理等内容的起止标志，方便阅读。官方模板未对这些内容进行规范，本模板中定义的结束符采用 ◊，例子的结束符采用 ⧫，定理的结束符采用 □，证明的结束符采用 ■。

定义 A.1 (**向量空间**)： 设 X 是一个非空集合，$\mathbb{F}$ 是一个数域 (实数域 $\mathbb{R}$ 或者复数域 $\mathbb{C}$)。如果在 X 上定义了加法和数乘两种运算，并且满足以下 8 条性质：

1. 加法交换律，$\forall\, x, y \in X$，$x + y = y + x \in X$；
2. 加法结合律，$\forall\, x, y, z \in X$，$(x + y) + z = x + (y + z)$；
3. 加法的零元，$\exists\, 0 \in X$，使得 $\forall\, x \in X$，$0 + x = x$；
4. 加法的负元，$\forall\, x \in X$，$\exists\ -x \in X$，使得 $x + (-x) = x - x = 0$。
5. 数乘结合律，$\forall\, \alpha, \beta \in \mathbb{F}$，$\forall\, x \in X$，$(\alpha\beta)x = \alpha(\beta x) \in X$；
6. 数乘分配律，$\forall\, \alpha \in \mathbb{F}$，$\forall\, x, y \in X$，$\alpha(x + y) = \alpha x + \alpha y$；
7. 数乘分配律，$\forall\, \alpha, \beta \in \mathbb{F}$，$\forall\, x \in X$，$(\alpha + \beta)x = \alpha x + \beta x$；
8. 数乘的幺元，$\exists\, 1 \in \mathbb{F}$，使得 $\forall\, x \in X$，$1x = x$，

那么称 X 是数域 $\mathbb{F}$ 上的一个**向量空间** (linear space)。

例 A.1 (**矩阵空间**)： 所有 $m \times n$ 的矩阵在普通矩阵加法和矩阵数乘运算下构成一个向量空间 $\mathbb{C}^{m\times n}$。如果定义内积如下：

$$\langle A, B\rangle = \mathrm{tr}(B^H Q A) = \sum_{i=1}^{n} b_i^H Q a_i \tag{A-1}$$

其中 a_i 和 b_i 分别是 A 和 B 的第 i 列，而 Q 是 Hermite 正定矩阵，那么 $\mathbb{C}^{m\times n}$ 构成一个 Hilbert 空间。 ⧫

定理 A.1 (Riesz **表示定理**)： 设 H 是 Hilbert 空间，H^* 是 H 的对偶空间，那么对 $\forall\, f \in H^*$，存在唯一的 $x_f \in H$，使得

$$f(x) = \langle x, x_f\rangle, \qquad \forall\, x \in H \tag{A-2}$$

并且满足 $\|f\| = \|x_f\|$。 □

证明： 先证存在性，再证唯一性，最后正 $\|f\| = \|x_f\|$。 ■

附录 B　算法与代码

对于数学、计算机和电子信息专业，算法和代码也是经常用到的排版技巧。

B.1 算法

算法描述使用 algorithm2e 宏包，效果如算法 B-1 所示。

Input: $\mathbf{x}(k), \quad \mu, \quad \mathbf{w}(0)$
Output: $y(k), \quad \varepsilon(k)$
for $k = 0, 1, \cdots$ **do**
　$y(k) = \mathbf{w}^H(k)\mathbf{x}(k)$　// output signal
　$\varepsilon(k) = d(k) - y(k)$　// error signal
　$\mathbf{w}(k+1) = \mathbf{w}(k) + \mu\varepsilon^*(k)\mathbf{x}(k)$　// weight vector update

算法 B-1　LMS 算法详细描述

B.2 代码

源代码使用 listings 宏包，LMS 算法的 Verilog 模块端口声明如代码 B-1 所示。

代码 B-1　空时 LMS 算法 Verilog 模块端口声明

```
module stap_lms
#(
parameter       M               = 4,    // number of antennas
                L               = 5,    // length of FIR filter
                W_IN            = 18,   // wordlength of input data
                W_OUT           = 18,   // wordlength of output data
                W_COEF          = 20    // wordlength of weights
)(
output  signed  [W_OUT-1:0]     y_i,    // in-phase component of STAP output
output  signed  [W_OUT-1:0]     y_q,    // quadrature component of STAP output
output                          vout,   // data valid flag of output (high)
input           [M*W_IN-1:0]    u_i,    // in-phase component of M antennas
input           [M*W_IN-1:0]    u_q,    // quadrature component of M antennas
input                           vin,    // data valid flag for input (high)
input                           clk,    // clock signal
input                           rst     // reset signal (high)
);
```

攻读学位期间取得的研究成果

研究成果包括以下内容：

1. 已发表或已录用的学术论文、已出版的专著/译著、已获授权的专利按参考文献格式列出。
2. 科研获奖，列出格式为：获奖人 (排名情况). 项目名称. 奖项名称及等级, 发奖机构, 获奖时间.
3. 与学位论文相关的其它成果参照参考文献格式列出。
4. 全部研究成果连续编号编排。

用于盲审的论文，只列出已发表学术论文的题目和刊物名称，可以备注自己为第几作者，及期刊影响因子。

学位论文独创性声明 (1)

本人声明：所呈交的学位论文系在导师指导下本人独立完成的研究成果。文中依法引用他人的成果，均已做出明确标注或得到许可。论文内容未包含法律意义上已属于他人的任何形式的研究成果，也不包含本人已用于其他学位申请的论文或成果。

本人如违反上述声明，愿意承担以下责任和后果：

1. 交回学校授予的学位证书；
2. 学校可在相关媒体上对作者本人的行为进行通报；
3. 本人按照学校规定的方式，对因不当取得学位给学校造成的名誉损害，进行公开道歉；
4. 本人负责因论文成果不实产生的法律纠纷。

论文作者 (签名)：　　日期：　　年　　月　　日

学位论文独创性声明 (2)

本人声明：研究生_____ 所提交的本篇学位论文已经本人审阅，确系在本人指导下由该生独立完成的研究成果。

本人如违反上述声明，愿意承担以下责任和后果：

1. 学校可在相关媒体上对本人的失察行为进行通报；
2. 本人按照学校规定的方式，对因失察给学校造成的名誉损害，进行公开道歉；
3. 本人接受学校按照有关规定做出的任何处理。

指导教师 (签名)：　　日期：　　年　　月　　日

学位论文知识产权权属声明

我们声明，我们提交的学位论文及相关的职务作品，知识产权归属学校。学校享有以任何方式发表、复制、公开阅览、借阅以及申请专利等权利。学位论文作者离校后，或学位论文导师因故离校后，发表或使用学位论文或与该论文直接相关的学术论文或成果时，署名单位仍然为西安交通大学。

论文作者 (签名)：　　日期：　　年　　月　　日

指导教师 (签名)：　　日期：　　年　　月　　日