可编程虚拟路由器关键技术与原型系统

罗腊咏[1),2)]，贺鹏[1),2)]，关洪涛[1)]，李振宇[1)]，谢高岗[1)]

[1)](中国科学院计算技术研究所网络技术研究中心 北京 100190)

[2)](中国科学院大学 北京 100049)

摘 要 未来网络创新试验床和数据中心网络迫切需要具有虚拟化和可编程能力的路由器设备的支持，可编程虚拟路由器是构建未来网络试验床和数据中心网络的核心设备，逐渐成为研究热点. 然而，可编程虚拟路由器在设计与实现中面临一系列挑战，其关键技术和原型系统研究对于可编程虚拟路由器研制具有十分重要的意义. 本文从未来网络试验床和数据中心网络的需求出发，分析了可编程虚拟路由器的特性要求，归纳了可编程虚拟路由器在虚拟化、可编程性和高性能数据包转发等方面存在的技术挑战，并分类讨论了相关关键技术研究进展. 论文最后评价和比较了国内外设计实现的可编程虚拟路由器原型系统，并讨论了可编程虚拟路由器中有待进一步解决的研究问题.

关键词 路由器；虚拟化；可编程性；性能

Key Technologies and Prototype Systems of Programmable Virtual Routers

LUO La-Yong[1),2)], HE Peng[1),2)], GUAN Hong-Tao[1)], LI Zhen-Yu[1)], XIE Gao-Gang[1)]

[1)](Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China)

[2)](University of Chinese Academy of Sciences, Peijing 100049, China)

Abstract Future Internet testbeds and data center networks need network devices with virtualization and programmability. As the key building blocks for future Internet testbeds and data center networks, programmable virtual routers have attracted much attention in recent years. However, we are facing great challenges when designing and implementing the programmable virtual routers, and thus it is critical to study the key technologies and prototype systems for building programmable virtual routers. This paper analyzes the requirements of programmable virtual routers for building future Internet testbeds and enabling innovation on data center networks, and summarizes the technical challenges, including virtualization, programmability, and forwarding performance. Then, this paper comprehensively introduces the existing research work that aimed to address the above technical challenges. At last, this paper presents the prototypes of programmable virtual routers, and discusses the remaining open issues.

Key words router; virtualization; programmability; performance

1 引言

现有的 TCP/IP 网络体系结构在可扩展性、移动性、安全可控性等方面面临诸多挑战[1]，未来网络研究旨在设计全新的网络体系结构和新的网络协议以应对上述挑战. 未来网络研究需要大规模试验床的支撑，而可编程虚拟路由器是构建未来网络试验床的关键设备. 可编程虚拟路由器是指支持可编程和虚拟化两个特性的新型路由器，可编程性允

收稿日期：2013年1月13日；最终修改稿收到日期：2013年5月4日. 本课题得到国家自然科学基金(61133015,61202411)；国家“八六三”高技术研究发展计划项目基金(2013AA013501)；中国科学院战略性先导科技专项项目基金(XDA06010303)；中国科学院重大科研装备项目(YZ201229)资助. 罗腊咏，男，1986年生，博士研究生，E-mail: luolayong@ict.ac.cn，主要研究领域为可编程虚拟路由器、硬件加速网络数据包处理. 贺鹏，男，1987年生，博士研究生，E-mail: hepeng@ict.ac.cn，主要研究领域为软件路由器、高性能数据包处理算法. 关洪涛，男，1980年生，博士，E-mail: guanhongtao@ict.ac.cn，副研究员，主要研究领域为网络虚拟化、路由器虚拟化. 李振宇，男，1980年生，博士，E-mail: zyli@ict.ac.cn，副研究员，主要研究领域为未来互联网体系结构、P2P计算. 谢高岗，男，1974年生，博士，E-mail: xie@ict.ac.cn，研究员，博士生导师，主要研究领域为未来互联网体系结构、可编程虚拟路由器、网络测量、分析与模型化.

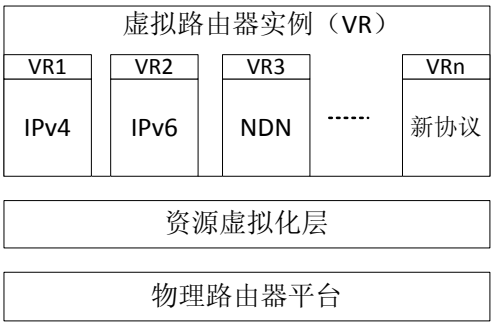

图 1 可编程虚拟路由器逻辑结构

许用户自定义数据包处理方法，而虚拟化支持多个路由器实例并行运行在同一个物理路由器平台之上. 为满足试验床的需求，可编程虚拟路由器需要向用户提供灵活的可编程能力，允许用户自定义数据包格式与处理过程，从而支持新协议的实验和部署. 在此基础之上，路由器平台还需要支持虚拟化，从而使多个用户实验可以并行隔离运行.

在数据中心网络中，多样化的应用需求（如高可靠、低延迟、高吞吐等）要求网络设备支持差异化的数据包转发方法（多路径路由、最短路径路由等）[2]. 可编程虚拟路由器支持灵活可编程，可以根据不同应用的需求，对数据包转发方式（如查找关键字、查找方法等）进行灵活修改. 此外，大型数据中心是海量服务器的互联，通过虚拟机和物理机的形式提供各种不同的服务. 由于不同服务之间存在隐私控制、流量隔离等需求，因此需要数据中心网络提供良好的虚拟化机制，使提供不同服务的业务子网相互隔离和独立，能动态共享数据中心网络中的服务器资源.

为了满足未来网络试验床和数据中心网络需求，可编程虚拟路由器需要具备以下特性.

（1）虚拟化：在一台路由器物理设备上同时运行多个虚拟路由器实例. 每个虚拟路由器实例可以运行不同的协议，路由器实例之间在功能、性能和故障方面相互隔离，互不干扰，从而支持现有网络业务与未来网络实验和部署的并行运行.

（2）可编程：路由器体系结构一般分为数据平面和控制平面两部分. 其中，数据平面主要负责数据包查找和转发操作，控制平面运行路由协议以实现路由信息交换和转发表生成. 在新型路由器中支持可编程特性，需要提供从数据平面到控制平面的多层次可编程接口，支持新转发算法和新路由协议，适应未来网络（IP 和非 IP 网络）创新的快速实验与部署.

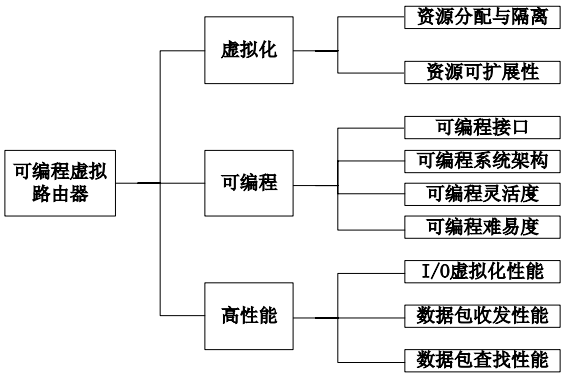

图 2 可编程虚拟路由器中的技术挑战

（3）高性能：基于有限的物理资源，虚拟路由器实例需要达到与商用路由器可比拟的吞吐量和时延等性能指标，使虚拟网络中数据包转发性能接近真实物理网络.

在以上特性中，可编程和虚拟化是松耦合关系. 可编程是为了支持自定义的数据包处理，从而支持新协议和新业务的实验与部署；而虚拟化的目的是在此基础之上支持多个不同路由器实例共存，从而支持多个不同用户实验和网络业务的并行隔离运行. 不过，可编程和虚拟化会增加系统开销，给路由器性能带来一定的影响（例如，I/O 虚拟化可能导致 I/O 性能下降），因此需要在支持虚拟化和可编程特性的同时，支持高性能数据包处理.

传统的商业路由器具有很好的数据包处理性能，但是其不支持虚拟化技术，或只支持少数同构的 IP 虚拟路由器实例，难以提供不同网络实验之间的隔离，以及实验用户与正常网络用户之间的隔离；此外，商业路由器的半封闭特性，导致其提供的编程和配置接口有限，难以满足新型网络体系结构和新协议实验和部署的需求. OpenFlow[5]在数据平面没有提供虚拟化支持，且基于多元组的规则匹配方式，难以满足非 IP 数据包对可编程性的需求. 尽管可重构路由器和可编程路由器能够提供可编程特性，支持自定义的数据包处理，然而，由于不支持虚拟化特性，无法使多个用户共享同一个物理路由器资源，即难以支持多个用户实验的并行隔离运行，以及现有网络业务与未来网络实验与部署的共存. 因此，同时支持虚拟化、可编程和高性能特性的新型路由器能够更好地满足未来网络试验床和数据中心网络的需求.

2 技术挑战

可编程虚拟路由器的逻辑架构如图 1所示. 在一个物理路由器平台之上，实现一个资源虚拟化层，对硬件资源进行虚拟化和管理. 在虚拟化层之上，同时运行多个虚拟路由器实例，每个虚拟路由器实例可以运行不同的协议. 在设计与实现上，可编程虚拟路由器在虚拟化、可编程性和数据包转发性能等方面，存在着诸多技术挑战，如图 2 所示.

（1）虚拟化问题. 在可编程虚拟路由器中，虚拟化引入两个方面的挑战：资源分配与隔离问题以及可扩展性问题. 在一个虚拟路由器平台中，多个虚拟路由器实例共享网络带宽、CPU 和存储器等物理资源. 资源分配的基本要求是满足各个虚拟路由器实例的应用需求. 在此前提下，需要尽可能公平的分配物理资源，避免若干虚拟路由器实例过度消耗资源导致资源浪费等问题. 此外，不同虚拟路由器实例在共享资源的前提下，还需要相互隔离，避免单个路由器实例的故障影响其它路由器实例的正常工作. 然而，资源分配与隔离会导致严重的性能问题，在合理分配资源与保证虚拟路由器实例之间隔离性的前提下，如何尽可能降低虚拟化对性能的影响，也是资源分配与隔离机制的重要挑战. 在可编程虚拟路由器中，多个虚拟路由器实例同时运行，每个路由器实例拥有一个转发引擎. 随着应用需求的不断增加和网络规模的扩大，虚拟路由器实例个数不断增加，从而导致转发引擎数量不断增加. 一方面，转发引擎中的包头协议解析和数据包修改等计算部件，对计算资源（如 CPU 资源、FPGA 中的寄存器和逻辑资源）的需求，随着虚拟路由器实例个数的增加而迅速增长；另一方面，转发引擎中的转发表（Forwarding Information Base, FIB）等存储部件，对存储资源（如 TCAM、SRAM 和 DRAM 等存储器）的需求，也随着虚拟路由器实例个数增加而线性增长. 如何在计算和存储资源有限的情况下，支持尽可能多的转发引擎，从而支持尽可能多的虚拟路由器实例，是可编程虚拟路由器中面临的可扩展性挑战.

（2）可编程性问题. 该问题主要体现在四个方面：可编程接口、可编程系统架构、可编程灵活度和可编程难易度. 可编程接口要定义良好的开发接口，使用户专注于业务功能开发，而不用关心底层的具体实现方式；可编程系统架构要抽象数据包处理的基本单元，并定义功能单元之间的统一接口，通过对已有功能单元的选择组合，实现用户自定义的数据包处理流程. 如何设计可编程架构，并保证系统实现性能不受影响，是可编程系统架构面临的挑战；此外，在可编程系统架构中，增强单个功能单元（如包头解析单元）的可编程灵活度，可以进一步提高系统的编程灵活性，满足用户自定义数据包处理需求，尽量避免用户重新开发新的功能单元. 然而，功能单元的可编程灵活度与实现复杂度之间存在矛盾，是功能单元可编程灵活度面临的主要挑战；最后，当需要用户重新开发新功能单元时，如何提供高效的编程方法，以降低用户编程难度，缩短用户开发时间，构成了可编程难易度的挑战.

（3）数据包转发性能问题. 在虚拟路由器中，数据包转发性能主要受到 I/O 虚拟化性能，驱动和内核级的数据包收发机制的效率，以及数据包查找算法性能的影响. 多个虚拟路由器实例的虚拟网络接口共享物理网络接口，需要 I/O 虚拟化技术的支持. 然而，I/O 虚拟化技术由于需要负责虚拟接口和物理接口之间的数据包分发，增加了系统 I/O 开销，降低了数据包转发性能. 此外，路由器实例中驱动和内核级的数据包收发机制的效率，也直接影响着数据包采集和发送性能. 在传统路由器中，IP 查找和包分类等查找匹配操作，是实现数据包快速转发的瓶颈. 在虚拟路由器环境下，上述瓶颈依然存在，而且由于可能承载非 IP 协议，数据包查找匹配问题较传统的 IP 查找和五元组包分类问题更为复杂，面临更为严重的挑战.

为了解决可编程虚拟路由器中的上述关键技术挑战，研究人员从虚拟化带来的资源分配与隔离问题以及可扩展性问题、可编程性问题和转发性能问题等方面展开研究.

3 关键技术研究进展

3.1 虚拟化

为了实现路由器虚拟化，需要重点研究解决资源分配与隔离、计算和存储资源可扩展性等问题.

3.1.1 资源分配与隔离

目前，可编程虚拟路由器中的资源分配与隔离通常采用传统的服务器虚拟化技术实现，例如，vRouter[35]使用 OpenVZ 和 Xen 实现资源分配与隔离，SwitchBlade[21]中使用 OpenVZ 虚拟化技术，而 PEARL[37]中使用 LXC 虚拟化技术. 已有研究[3][4]

对服务器虚拟化技术进行了较为全面的综述，本文按照虚拟层次的不同，将虚拟化技术分为三类，并进行简要分析和总结.

（1）全虚拟化. 该技术在操作系统和底层硬件之间使用一个软件中间层 Hypervisor（虚拟机管理程序）管理底层硬件资源，使得硬件资源能够被上层多个客户操作系统共享访问. 当客户操作系统执行某项指令时，Hypervisor 将接管该段代码，并执行对应的操作. 由于在全虚拟化技术中，客户操作系统不能对底层硬件进行任何直接控制，而必须通过 Hypervisor 进行代理，使得 I/O 性能会受到一定影响. 全虚拟化技术可以支持不同的操作系统，但是不能模拟不同硬件平台. 全虚拟化技术的代表软件为 VMware，KVM 等. 近年来一些新的 X86 系列的 CPU 在指令级别上支持全虚拟化，使得全虚拟化环境下的操作系统能达到原生级别的运行速度.

（2）半虚拟化. 该技术同样采用软件中间层来隔离底层硬件和操作系统，不同的是，它试图修改操作系统代码，让客户操作系统本身知道自己运行在一个虚拟化平台上. 该技术能够显著提升虚拟化技术的 I/O 性能. 例如，当网卡收到一个数据包时，在采用全虚拟化的情况下，Hypervisor 必须要模拟硬件中断，然后将数据包传输到某个操作系统上. 而由于在半虚拟化技术中，操作系统本身就知道自己运行在一个虚拟化平台上，底层的 Hypervisor 可以使用一定方式，批量的将数据包传输给该操作系统，而不需要模拟硬件中断. 由于半虚拟化技术需要对操作系统代码进行修改，使得该项技术不能应用于非开源的商业操作系统，如 Windows. 半虚拟化技术代表软件为 Xen 等.

（3）操作系统级虚拟化. 操作系统级虚拟化技术和以上所有虚拟化都不相同，它更类似于 UNIX 的 Jail 系统调用.它利用操作系统所提供一些机制（如 Linux 中命名空间），将一组进程放入到“容器”（container）中，与主操作系统的进程隔离开来，并使用这些机制限制这一组进程的 CPU 占用率和内存使用情况，从而达到虚拟化技术中资源的隔离与分配的目的. 这类虚拟化技术只能运行在特定的操作系统上（几乎都是 Linux）. 所有的“虚拟机”（在该技术中，称为 Server）都共享一个内核. 该虚拟化技术的代表软件为 OpenVZ，Linux-VServer，LXC 等. 操作系统级别虚拟化所带来的开销非常小，但是主要缺点也很明显，使用该虚拟化技术，不能运行多个异构的操作系统.

3.1.2 资源可扩展性

可编程虚拟路由器中的可扩展性研究，主要考虑在资源受限的情况下，如何支持多种新型网络协议对数据包处理的不同需求，以及支持尽可能多的虚拟路由器实例同时运行（即支持尽可能多的网络协议与业务实验与部署的并行运行）. 为了提高可扩展性，研究人员提出了统一的数据平面抽象，以一个统一的数据平面抽象支持不同新型网络协议对数据包处理的特殊需求；在此基础之上，进一步研究多个数据平面共存带来的可扩展性问题，提出了数据平面模块共享、转发表合并、软硬件虚拟数据平面迁移等方法.

3.1.2.1 统一的数据平面抽象

OpenFlow[5]为TCP/IP协议数据包转发提供了一个统一的数据平面抽象. OpenFlow将数据平面抽象为规则抽取、查找匹配和动作执行三个部分. 首先抽取数据包头部的若干重要域（如十元组）作为查找匹配的关键字，然后对关键字中的域进行精确匹配或通配符匹配. 最后根据匹配结果对数据包进行转发、丢弃或修改等操作. OpenFlow的统一数据平面能够满足TCP/IP体系中各种不同协议的需求，然而，OpenFlow的数据平面抽象没有考虑对全新网络协议的支持，难以满足新型网络协议特殊的数据包处理需求. 为了解决这一问题，LabelCast[6]提出了一种普适的数据平面抽象，包括Label表和Cast表两部分. LabelCast将网络地址映射为定长标签，通过Label表实现无状态的快速标签查找转发，基于Cast表对服务器计算和存储资源进行描述，支持复杂的网络协议语义和带状态的深度报文处理服务. 因此，LabelCast不仅仅支持现有路由器中常见的基于规则的转发方式，还支持新型网络转发服务，如NDN① (Named Data Networking)中基于名字的转发. 即，LabelCast以一个统一的数据平面抽象，支持多种不同网络协议的数据包处理需求，提高了数据平面的可扩展性.

3.1.2.2 数据平面模块共享

Anwer[7]等人通过分析虚拟路由器数据平面的基本功能模块特性，将数据包处理模块划分为可共享模块和不可共享模块. 其中，可共享模块是指无状态模块，不需要维护跟虚拟路由器实例相关的任何状态信息，因此可以在不同虚拟路由器数据平面

① Named Data Networking (NDN). http://www.named-data.net/

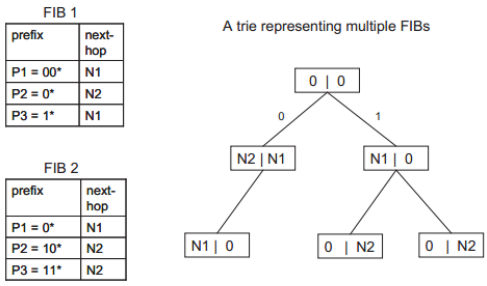

图 3　trie overlap 方法[10]

之间共享，如数据包头部解析、校验和计算、TTL 减一等单元. 不可共享模块是指带状态的模块，需要维护跟特定虚拟路由器实例相关的信息，如转发表模块. 根据上述分析，Anwer 等人基于可编程硬件 FPGA 设计了虚拟路由器数据平面，通过在各个数据平面之间共享模块，有效提高硬件资源利用率. 实验结果表明，在一块 NetFPGA[8]数据包处理卡上，可以实现 8 个虚拟路由器数据平面. 与独立实现 8 个虚拟路由器数据平面相比，节省约 75%~79%的逻辑资源.

上述方法通过共享不同虚拟路由器数据平面之间的计算资源，在一定程度上提高了虚拟路由器数据平面的可扩展性. 然而，由于转发表资源并没有在虚拟路由器实例之间共享，总的转发表大小随着虚拟路由器实例个数的增加而线性增长. 当虚拟路由器实例个数为 8 时，占用了 FPGA 片内 87%的存储资源. 因此，有限的物理存储资源，进一步限制其支持的虚拟路由器实例个数难以继续增加.

3.1.2.3　转发表合并

虽然不同虚拟路由器实例的转发表各不相同，无法直接共享和复用. 然而，通过挖掘不同虚拟路由器实例转发表之间的相似性，可以实现多个转发表的合并，从而有效降低多个转发表对存储资源的需求，进一步提高可编程虚拟路由器的可扩展性.

转发表的存储器类型不同，其数据结构和合并算法有很大区别. 根据转发表存储使用的物理存储器类型，可以将转发表合并方法分为基于 SRAM 和基于 TCAM 的合并方法.

在基于 SRAM（或 DRAM）的转发表查找方法中，常用的一种数据结构为 trie 树[9]. Fu 等人[10]首次提出了将多个虚拟路由器实例的转发表合并压缩到一个 trie 树的方法，即 trie overlap 方法. 在 trie overlap 方法中，通过挖掘不同转发表中前缀之间的相似性（其对应的 trie 树也是相似的），将多个转发表合并成一个 trie 树进行存储. 图 3 所示为使用 trie

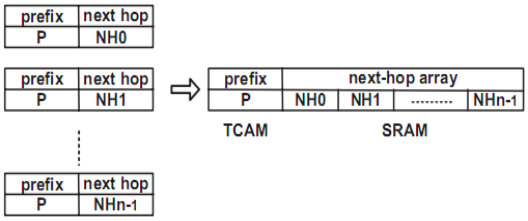

图 4　基于 TCAM 的转发表合并数据结构[15]

overlap 方法，将图中两个转发表 FIB 1 和 FIB 2 合并成一个 trie 树. 与多个 trie 树单独隔离存储相比，该方法有效降低了合并后 trie 树的节点个数，从而节省存储空间. 然而，在合并后的 trie 树中，每个节点中存储的下一跳指针个数跟虚拟路由器实例个数相同（如图 3 所示，每个 trie 树节点中要存储两个下一跳指针），因此，当虚拟路由器实例个数不断增加时，节点所占存储空间太大，这会导致两个问题：（1）访问每个节点需要多次访存操作，增加了每次 IP 查找所需的总访存次数，从而降低了查找速度；（2）过大的节点会导致合并后的 trie 树存储空间需求较大，难以在空间较小的高速存储器中进行存储. 为了解决节点太大这个问题，trie overlap 方法使用叶推技术[11]，将中间节点存储的所有下一跳指针推至叶子节点，从而中间节点不需要存储下一跳信息. 使用叶推技术后，trie overlap 方法中 trie 树节点大小显著降低. 然而，叶推技术会导致增量更新困难. 为了避免这一问题，Luo 等人在合并后的 trie 树节点中引入前缀位图[12]，使节点和下一跳信息分离. 每增加一个虚拟路由器实例，trie 树节点大小只增加 1 比特，有效提高了节点大小的可扩展性；在 trie 树合并的过程中，避免了叶推技术的使用，从而在增加可扩展性的同时，支持快速的增量更新.

当不同 FIB 的前缀相似度很高时，trie overlap 方法的压缩效果很好. 然而，虚拟路由器中多个转发表的前缀相似度不一定很高，此时仅仅使用 trie overlap 方法，可能无法取得明显的压缩效果. 为了将前缀不相似的多个 FIB 对应的 trie 树转换为相似，Song 等人[13]提出了一种 trie braiding 机制，通过在 trie 树节点中增加一个分支比特，允许 trie 树每个节点的左、右孩子互换位置，从而将不相似的 trie 树调整到相似，以方便使用 trie overlap 方法进行合并压缩. Ganegedara 等人[14]发现，在 VPN 路由器中，其 FIB 前缀中由于存在不同的公共前缀，即使使用 trie braiding 机制也很难将其对应的 trie 树转换为相

似的 trie 树，因此，提出了一种 multiroot 方法，允许从 trie 树的某个中间节点开始合并，从而最大程度上利用子树的相似性.

基于 TCAM 实现 FIB 存储的基本思路是将前缀逐条存储在 TCAM，将前缀对应的下一跳信息存储在附属的 SRAM. 如果每个虚拟路由器实例的转发表单独存储，TCAM 存储空间需求随虚拟路由器实例个数（FIB 个数）的增加而线性增长，有限的 TCAM 存储空间中难以支持大量的虚拟路由器实例的 FIB.

为了解决这一问题，Luo 等人[15]通过挖掘不同 FIB 前缀之间的相似性，在 TCAM 中存储共享前缀，从而降低 TCAM 存储空间需求. 多个转发表在 TCAM 中共享前缀的基本数据结构如图 4 所示. 如果属于不同 FIB 的某条前缀相同（下一跳不必相同），可以将其合并成一条前缀，该前缀对应的所有下一跳信息顺序存储在 SRAM 中. 例如，分属于两个 FIB 的条目<P，NH1>和<P，NH2>通过共享前缀，可合并成一个条目<P，[NH1，NH2]>.

基于上述数据结构，属于不同虚拟路由器实例的转发表可以合并在一起，从而显著降低 TCAM 存储空间的需求. 然而，上述合并方法可能导致不正确的 IP 查找结果. 为了避免转发表合并带来的不正确查找的问题，Luo 等人提出了两种基于 TCAM 的 FIB 合并方法：FIB completion 和 FIB splitting. FIB completion 将所有的转发表前缀合并在一个 TCAM 中，并使用正确的下一跳结果填充所有无效的下一跳. 该方法具有最好的可扩展性，然而其最坏情况下的更新开销较大；FIB splitting 将转发表中不相交的叶子前缀合并在一个 TCAM 中，将其余前缀以隔离存储的方式存储在另一个 TCAM 中. 该方法也能保持较好的可扩展性，同时保证较小的更新开销上界. 与传统的隔离存储方式相比，上述两种方法能够有效降低多转发表对存储空间的需求. 存储 14 个核心路由器的转发表时，TCAM 存储空间需求分别降低了 92%和 82%.

3.1.2.4 软硬件虚拟数据平面迁移

高效的处理模块共享和转发表合并，可以分别提高计算资源和存储资源的利用率，从而提高硬件虚拟数据平面的可扩展性. 然而，高速硬件资源有限是目前存在的客观事实，仅仅依靠提高硬件资源利用率，难以从根本上解决可扩展性问题. 要想支持更多的虚拟路由器实例个数，必须借助于可扩展性很好的软件虚拟数据平面.

Unnikrishnan 等人[16]提出以软件虚拟路由器和硬件虚拟路由器相结合的方式进一步提高可扩展性. 少数的、高吞吐的虚拟路由器数据平面在 NetFPGA 中实现，其它大量的、低吞吐的虚拟路由器数据平面在服务器 OpenVZ 中实现. 硬件数据平面和软件数据平面支持相同的用户定义的接口，以保证数据平面的位置（在 FPGA 硬件中还是服务器软件中）对用户来讲是透明的. 虚拟路由器数据平面可以在软件和硬件中进行动态迁移，以满足动态变化的应用需求.

虚拟路由器数据平面从硬件往软件迁移的过程十分简单. 首先，在服务器中新创建一个 OpenVZ 虚拟机实例，并在其中运行 Click[19]软件路由器，配置 Click 使得数据包转发处理过程与待迁移的硬件数据平面的处理流程完全相同；其次，将待迁移的硬件数据平面中的转发表，完整地拷贝到刚刚运行的软件路由器中；最后，改变数据包分发模块的配置，将发送至该硬件数据平面的数据包重定向到软件路由器中.

数据平面从软件往硬件迁移有一些限制. 如果软件虚拟路由器实例和硬件虚拟路由器实例的转发机制完全相同，拷贝转发表可以实现软件虚拟路由器实例往硬件的迁移. 如果软件虚拟路由器实例中的转发机制，在硬件数据平面中并不支持，简单的转发表拷贝无法满足迁移的需求，需要对硬件数据平面的逻辑功能进行修改，才能满足这种迁移需求. 因此，Unnikrishnan 等人[16]提出使用在线动态可重构的方法解决这一问题. 当硬件数据平面需要进行显著修改才能满足迁移要求时，通过以下步骤实现动态迁移. 首先，将所有的硬件虚拟数据平面迁移到软件中，并将发往这些硬件数据平面的流量重定向到相应的软件路由器中；其次，对整个 FPGA 进行重新编程，在硬件数据平面中实现新的数据包转发机制；最后，将待迁移的软件路由器实例中的转发表拷贝到相应的硬件数据平面中，并配置数据包分发模块，使流量重新定向至硬件虚拟路由器中.

软硬件虚拟路由器相结合的方式，可以很好的解决虚拟路由器实例的可扩展性问题. 然而，上述方法中，软件路由器往硬件迁移时可能存在较大开销. 当一个软件路由器向硬件迁移时，如果硬件中所有的虚拟数据平面均无法实现该软件路由器所需的转发机制，所有的硬件虚拟数据平面都必须迁移到软件中，然后将整个 FPGA 进行重新编程，最后将对应的虚拟路由器实例迁移回硬件. 实验结果

表 1 虚拟化问题研究进展

方法	设计目标	主要思想	仍存在的问题
资源分配与隔离	为虚拟路由器实例分配物理资源，并保证隔离性	使用服务器虚拟化技术实现 CPU、存储和网络带宽等资源的分配与隔离	隔离性和转发性能之间存在矛盾，通常隔离性越强，转发性能越差
统一的数据平面抽象	为多种网络协议的数据包处理提供统一的数据平面	对数据平面中数据包处理流程、计算和存储资源的描述进行抽象	需要进一步研究资源描述和服务扩展机制，以提高数据平面的普适性
数据平面模块共享	降低多个数据平面共存时的计算资源消耗	挖掘数据平面间的共同处理单元，实现处理模块共享	硬件数据平面可扩展性仍然受到存储资源的限制
转发表合并	降低多个数据平面共存时的存储资源消耗	挖掘转发表间的相似性，实现不同转发表的合并	硬件资源利用率的提高，难以从根本上解决硬件可扩展性不足的问题
软硬件虚拟数据平面迁移	使用软件数据平面的可扩展性，弥补硬件数据平面扩展性的不足	软硬件数据平面结合，通过软硬件数据平面动态迁移，提高系统可扩展性	降低迁移开销，尤其是异构数据平面的软硬件迁移开销，值得进一步研究

表明，当硬件中只有一个虚拟数据平面时，上述迁移过程需要超过 10 秒的时间. 在这段时间里，网络数据包不能在 FPGA 硬件中进行转发.

为了降低迁移时间，Yin 等人[17]提出将每个硬件虚拟数据平面的转发逻辑和转发表划分为一个局部可重构区域. 当硬件数据平面的转发机制需要修改时，不需要对整个 FPGA 进行重新编程，只需利用 Xilinx 公司 FPGA 提供的局部动态可重构特性，对其中某一个局部可重构区域进行动态编程，从而显著降低了迁移时间，同时也不需要在迁移过程中停止整个 FPGA 硬件数据平面的数据包转发操作. 实验结果表明，与整个 FPGA 完全重新编程的迁移方法相比，该方法能将迁移速度提高近 20 倍.

表 1 总结了虚拟化问题研究进展. 路由器虚拟化带来资源分配与隔离、资源可扩展性等问题. 其中，资源分配与隔离机制目前通常采用成熟的服务器虚拟化技术实现，在隔离性与转发性能之间存在矛盾. 资源可扩展性问题主要来源于计算和存储资源的稀缺性与虚拟路由器实例个数不断增长的应用需求之间的矛盾. 研究人员提出统一的数据平面抽象以支持多种不同网络协议的数据包处理，扩展数据平面的普适性. 在多个数据平面共存时，通过共享和复用多个虚拟路由器的数据包处理模块，合并多个虚拟路由器的转发表等方法，有效提高了硬件资源的利用率，从而改善了硬件虚拟数据平面的可扩展性. 然而，即使硬件资源利用率很高，其支持的硬件虚拟数据平面个数仍然比较少. 为了进一步提高可扩展性，研究人员提出了使用软件虚拟路由器与硬件虚拟路由器相结合的方式，在硬件中实现少量高性能的路由器实例，在软件中实现大量低吞吐、高灵活性的路由器实例. 同时，支持软硬件路由器实例的动态迁移，并使用局部动态可重构技术降低迁移开销，从而满足动态变化的应用需求.

3.2 可编程性

软件系统和可编程硬件系统（如 FPGA）本身具备可编程性，可以满足自定义数据包处理的需求. 然而，在虚拟路由器中研究可编程性的目的，是为了提出更简单的编程机制，从而降低用户编程的难度；提出更快速的编程机制，从而缩短原型系统设计、实验与部署的时间. 为了实现这一目标，研究人员设计了开放的可编程接口，模块化的流水线架构和灵活的可编程模块，实现了易于使用的网络数据包处理编译器.

3.2.1 开放的可编程接口

OpenFlow 协议[18]定义了开放的可编程接口，允许用户通过软件自定义数据包转发操作，而不用关心底层交换机的具体实现方式. OpenFlow 的可编程接口由一组预先定义的消息（控制器-交换机消息、异步消息和对称消息）以及这些消息的数据结构组成. 通过这些消息实现 OpenFlow 交换机和控制器之间的指令和状态信息交互. 该可编程接口对现有 TCP/IP 网络协议的查找转发操作支持较好，然而，没有提供对复杂的带状态的深度数据包处理的支持. LabelCast[6]中定义了基本的服务原语（缓冲区原语、线程原语和注册原语），作为用户自定义服务开发的编程接口. 扩展的 LabelCast 资源容器提供了用户编程的环境，提供了服务运行所需的各种资源. 用户通过使用缓冲区原语和线程原语分别实现对存储和计算资源的管理，使用注册原语（即

开放的服务注册接口）对自定义服务进行动态加载. 由于 LabelCast 定义的服务支持协议语义和状态相关的深度数据包处理，该编程接口可以方便扩展支持新型网络协议.

3.2.2 模块化的流水线架构

模块化的流水线架构，主要是为了解决可编程系统的架构问题. 模块化的可编程思想是，将路由器中数据包处理的基本单元模块化，通过对这些基本模块进行选择和组合，实现自定义的数据包处理流水线. 基于上述模块化的思想，Morris 等人[19]提出了模块化的路由器数据平面 Click，支持用户自定义的数据包查找和转发处理；Handley 等人[20]提出了模块化的路由器控制平面 XORP，支持用户自定义的路由计算和路由管理等功能. 因此，使用 Click 和 XORP 相结合，可以实现模块化的路由器. 当前主流的 Linux 操作系统通过一定调节和配置也能够实现与 Click 相同的路由器数据平面功能.

然而，Click数据平面并没有考虑对虚拟化的支持. 此外，纯软件实现方式导致其数据包查找和转发的性能较差. 为了解决这些问题，Anwer等人[21]基于NetFPGA硬件实现了模块化的虚拟路由器SwitchBlade. 在数据平面，SwtichBlade采用模块化的流水线架构，基于FPGA实现多个相互隔离的虚拟数据平面，同时兼顾了良好的可编程性和很高的转发性能；在控制平面，基于OpenVZ提供虚拟化环境，在每个虚拟机中运行Quagga②作为路由软件.

在上述模块化的路由器结构中，可编程性的强弱取决于预先实现的基本模块（基本处理单元）功能是否完备，能否灵活支持模块间的自由选择组合. 然而，一个系统中预先实现的基本模块有限. 进一步提高可编程性有两种基本的思路：一种思路是实现灵活的可编程模块，增强单个基本模块的灵活性；另一种思路是降低基本模块的开发难度，以方便用户快速实现自定义模块.

3.2.3 灵活的可编程模块

在模块化的流水线架构基础之上，增强每个基本模块的灵活度可以进一步提高可编程性. 例如，在包头协议解析模块，如果只实现基本的 IPv4 和 IPv6 协议解析和目的 IP 地址抽取功能，其灵活性非常有限，只能适用于 IP 查找操作；如果该模块能够支持自定义偏移、自定义长度的多域关键字抽取，则可以适应 MAC 地址查找、IP 查找、五元组匹配等多种查找操作的包头提取.

对于传统的 TCP/IP 协议，OpenFlow[5]提出了非常灵活的可编程模块. OpenFlow 将数据包的处理抽象为包头解析、关键字匹配和动作执行三个模块. 在包头解析模块，提取数据包的十元组作为查找关键字. 在关键字匹配模块，支持基于十元组的通配符匹配和精确匹配. 在动作执行模块，定义和实现了路由器中常用的数据包执行动作. 上述灵活的可编程模块，可以很好地满足目前 TCP/IP 路由器中常见的各种数据包处理需求.

然而，OpenFlow 基于预先定义的十元组的匹配方法仍然有其编程的局限性. 首先，关键字的提取受到预先定义的十元组的限制. 如果用户的新协议使用数据包头的其它域（不在十元组定义的范围之类），OpenFlow 将不能很好的对这种协议进行支持. 为了弥补这一缺陷，OpenFlow 定义的关键字越来越长，从最初的 10 元组扩展到 36 元组；其次，OpenFlow 中元组的定义是基于标准的 TCP/IP 协议，其可编程范围局限在 TCP/IP 体系范围之内. 对于非 IP 查找，如 NDN 查找等，OpenFlow 预定义的多元组无法满足需求.

CAFE[22]和 SwitchBlade[21]在其包头解析模块中，设计了任意比特抽取器，其最终目的就是支持数据包头部任意比特域的自由组合，从而支持用户自定义的查找关键字提取方式，支持新的协议类型. 这种包头解析模块的灵活性越强，其可编程性越好，对新协议的支持就越好. 然而，其复杂程度也越高. 复杂的抽取过程给包头解析带来了性能方面的挑战. 为了实现高性能的、灵活的包头解析模块，Attig 等人[23]基于 FPGA 设计了 400 Gbps (Gigabit per second，千兆比特每秒)包头解析流水线，将每一个头部的解析映射到流水线中的一个流水级，在流水级中通过用户配置的微码，实现灵活的自定义多域抽取.

3.2.4 数据包处理编译器

如果通过基本模块的自由选择组合，以及可编程模块的灵活配置，仍然无法满足用户的新协议对数据包处理的需求，就需要用户自己编写相应的数据包处理单元，然后以插件的方式，插入到模块化的流水线架构中，或者替换已有模块.

对于软件数据包处理模块来讲，统一的、友好的编程接口可以方便用户的编程过程；然而，对于硬件可编程模块，即使具备定义良好的模块接口，用户自己编程也会遇到较大的困难，因为涉及到硬

② The quagga routing suite. http://www.nongnu.org/quagga/

表 2 可编程性问题研究进展

方法	设计目标	主要思想	仍存在的问题
开放的可编程接口	提出开放的、统一的编程接口，支持用户自定义的数据包处理	抽象数据包处理的基本操作（或原语），用户通过对基本操作的调用进行编程	可以在基本操作（或原语）的基础上，实现常用的数据包处理算法，并将其封装成高层编程接口，以提高用户编程效率
模块化的流水线架构	提出可编程的数据包处理系统架构	将包处理单元模块化，通过对模块自由组合和替换，实现自定义的包处理流水线	当已有模块的组合无法满足特定的数据包处理需求时，需要用户自己编写新模块
灵活的可编程模块	增强包处理架构中单个模块的可编程灵活度	丰富单个模块的功能，通过对模块的简单配置满足更多的处理需求	丰富模块的功能导致模块实现的复杂程度提高，即灵活性与高性能之间存在矛盾
数据包处理编译器	降低模块或插件的编程难度	当重新编程无法避免时，设计高级语言编译器，避免用户直接编写底层硬件代码	编译器自动产生的硬件模块，效率和性能不一定最优，即编程的友好性和高性能之间也存在矛盾

件编程语言 VHDL/Verilog 的学习使用，以及对硬件底层细节的了解和认识.

为了简化用户的编程难度，尤其是对硬件模块的编程难度，研究人员设计了网络数据包处理编译器，可以让用户使用自己熟悉的高级语言，如 C/C++或脚本语言，编写数据包处理功能，然后，通过编译器将其转换为对应的硬件模块实现. Unnikrishnan 等人[24]针对基于 FPGA 的模块化虚拟路由器数据平面，设计 Click 风格的数据包处理编译器 ReClick，以降低硬件模块编程的难度. 如果用户熟悉 Click 的编程方式，将会十分容易地使用 ReClick 进行硬件数据包处理. 在 Attig 等人[23]设计的包头解析流水线中，也同时设计了数据包头部解析语言（Packet Parsing Language，PPL），以类 C++的风格定义了每个头部的格式和处理规则，方便用户对新协议包头解析的自定义处理.

表 2 总结了虚拟路由器中可编程性问题的研究进展. 研究人员设计了开放的可编程接口，以简化用户编程难度，使用户不用关心底层的具体实现方式；提出模块化的数据包处理架构，设计实现了常见数据包处理流程中的大部分模块，并提供统一的、友好的接口. 用户对已有模块进行复用和组合，以满足大部分常见的数据包处理需求，有特殊需求的用户也只需将精力集中于新协议特定的数据包处理功能上，然后以插件的方式，将自定义的模块插入到包处理流水线架构中；在模块化的架构中，提高每个重要模块，尤其是包头解析模块的灵活性，整体架构对新协议的适应性就越强，就越能降低用户开发新系统的工作量，避免大量的重新开发工作；在用户开发实现无法避免的情况下，通过研究设计网络数据包处理编译器，提供给用户熟悉的高层编程语言接口，降低用户编程的难度和门槛，从而实现用户自定义协议的快速编程实现.

3.3 转发性能

在可编程虚拟路由器中，I/O 虚拟化带来的开销，数据包收发机制的效率，以及数据包查找匹配操作的性能，成为限制虚拟路由器转发性能的重要因素. 为了提高转发性能，工业界和学术界提出大量的虚拟 I/O 加速技术，优化驱动和内核级的数据包收发机制，并加速查找匹配操作的性能.

3.3.1 I/O虚拟化加速技术

在可编程虚拟路由器中，每个虚拟路由器实例都有自己的虚拟网络接口，不同路由器实例之间的虚拟接口逻辑上相互隔离，物理上共享实际的网络接口，如图 5 所示. 为了实现虚拟网络接口的抽象，虚拟机管理程序需要负责完成物理网络接口与各个虚拟网络接口之间的数据包分发，参与各个虚拟

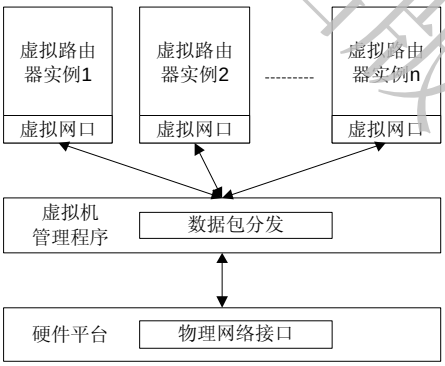

图 5　虚拟环境下的网络接口

路由器实例的每一次 I/O 操作. 因此，虚拟机管理程序中的数据包分发工作成为 I/O 虚拟化带来的瓶颈.

Russell 等人[25]给 Linux 操作系统中的各种虚拟化软件设计了一套标准化的半虚拟 I/O 接口 Virtio. 半虚拟化 I/O 能够让客户操作系统意识到自身运行在虚拟化环境中，并在这个基础上采用一定的优化措施，提升了客户操作系统的 I/O 性能. 目前，KVM 虚拟化技术支持 Virtio.

尽管半虚拟化I/O技术能够提升客户操作系统的I/O性能，但仍然存在上下文开销过高等问题. 解决问题的关键在于取消虚拟机管理程序的代理，让客户操作系统能够直接访问到物理设备. 具体说来，直接访问物理设备可以划分为两个问题：1）对客户操作系统的设备缓冲区直接DMA（Direct Memory Access）访问. 2）中断重映射. Intel VT-d技术③（Intel Virtualization Technology for Directed I/O）解决了以上两个问题，提供了数据由底层硬件到客户操作系统的直接通路. SR-IOV④（Single-Root I/O Virtualization）提供了将单个PCI设备虚拟成多个设备的方法. SR-IOV为每个虚拟机提供独立内存空间、中断和DMA流. SR-IOV架构的设计允许一个I/O设备支持多个虚拟功能， 从而提供了一种不需要软件模拟就可以共享I/O设备和I/O端口的物理功能的方法. 此外， Xen 开发人员开发的PCI Passthrough⑤技术支持直接访问PCI设备，使得客户操作系统能越过虚拟机管理程序访问设备，从而获得接近原生Linux环境下的I/O性能.

3.3.2 数据包收发机制优化

I/O 虚拟化加速技术降低了虚拟化引入的额外 I/O 开销. 在此基础上，数据包收发机制的优化，能够进一步提高数据包的转发性能.

传统的 I/O 中断方式具有响应速度快、延迟低等优点，然而，当数据包速率过高时，频繁的中断处理会显著增加软件负担. RouteBricks[26]使用轮询方式代替传统硬件的中断方式，极大提高了通用硬件的 64 字节小包转发性能；采用批处理的方式，每次总线事务（bus transaction）都传递多个数据包的地址，充分利用了 PCIE 的总线带宽；利用网卡的多队列特性，将不同队列与不同核进行绑定，极大的提升了数据处理的并行性. 使用以上三种手段，可以使得单台服务器的 64 字节小包转发性能达到 9.7 Gbps. PacketShader[27]对 Linux 的内核协议栈进行了优化，将内核中 sk_buff 结构体由原来的 208 字节减少为 8 字节，采用批处理的方式，极大的提升了数据包处理性能. 实验表明，仅采用这种优化手段，可以将数据包吞吐量提升到 10.5 Gbps，性能提升 13.5 倍. PacketShader 同时也利用了多核多队列的特性，并且考虑到其采用的架构为 NUMA(非一致性内存访问)，针对数据包的分发均衡性和提升数据局部性做了精心优化，使其整体转发性能达到 40 Gbps.

3.3.3 查找算法优化

数据包查找操作一直都是软件虚拟路由器设计中的一个主要的性能瓶颈. 文献[9][28]对传统的 IP 查找算法进行了全面综述，本文中就不再对传统的 IP 查找算法进行描述. 传统的 IP 查找算法也能够应用于可编程虚拟路由器中，用于实现快速的 IP 查找. 然而，可编程虚拟路由器不仅仅需要支持快速的 IP 查找，还需要对非 IP 协议进行支持，如支持 NDN 协议中的基于内容名字的查找等.

NDN 中的内容名字具有类似 HTTP URL 的层次结构. 与传统的 IP 地址相比，NDN 数据名字长度不定长，且无上界，导致线速最长名字前缀匹配成为 NDN 查找中一个重要的挑战. 为了解决这一问题，Wang 等人[29]提出了高效的名字子串编码方法，并应用改进的状态转移数组加速最长名字前缀匹配. Yuan 等人[30]评估了 NDN 的原型系统实现 CCNx，分析了其转发平面的性能瓶颈，并找到影响 NDN 转发性能的三个主要问题：支持快速更新的精确字符串匹配问题，基于变长和无上界名字的最长前缀匹配问题，以及大规模流表维护问题. 针对这些问题，提出了五大设计原则，用于指导性能可扩展的 NDN 转发引擎设计.

表 3 对可编程虚拟路由器中转发性能问题的研究进行了总结. 其中，I/O 虚拟化的性能开销是虚拟化技术的引入带来的特殊问题，已经得到学术界和工业界的广泛关注，部分虚拟 I/O 性能加速技术，如 Intel VT-d 技术、SR-IOV 技术等，已广泛应用于服务器和网卡产品中；传统的数据包收发机制以及 IP 查找操作，在虚拟化环境下依然成为性能瓶颈，一直以来都受到学术界的持续关注；非 IP 环境，尤

③ Intel VT-d. http://software.intel.com/en-us/articles/intel-virtualization-technology-for-directed-io-vt-d-enhancing-intel-platforms-for-efficient-virtualization-of-io-devices

④ SR-IOV. http://www.pcisig.com/specifications/iov/single_root/

⑤ PCI Passthrough. http://wiki.xen.org/wiki/Xen_PCI_Passthrough

表 3 转发性能问题研究进展

方法	设计目标	典型方案	仍存在的问题
I/O 虚拟化加速	降低虚拟化 I/O 接口带来的额外开销	Virtio, Intel VT-d, SR-IOV, PCI Passthrough 等	如何基于 SR-IOV 技术加速可编程虚拟路由器的虚拟 I/O 性能值得进一步研究
数据包收发机制优化	提高数据包接收和发送的性能	RouteBricks，PacketShader 等	除了软件收发机制优化，还需进一步优化硬件 DMA 传输效率，获得更多的收发性能的提升
数据包查找算法优化	提高数据包查找匹配的速度	各种典型的 IP 查找算法和面向 NDN 协议的非 IP 查找算法	NDN 协议中基于不定长名字的最长名字子串匹配问题，其查找性能有较大的优化空间

其是 NDN 体系结构下的路由查找问题，随着新型体系结构研究的深入，也逐渐得到人们的重视，成为一个新的重要研究方向.

4 原型系统研究进展

可编程虚拟路由器原型系统的设计可追溯到 PlanetLab. Peterson 等人[31]于 2003 年规划了 PlanetLab 的设计蓝图，并首次提出了“网络分片”（slice-ability）的设计思想. 随后，Bavier 等人[32]详细介绍了 PlanetLab 的设计原则和实现细节，其中一个重要的设计原则就是“不同的服务使用不同的网络分片”. PlanetLab 的“分片”思想能够为不同的服务提供异构的运行网络，能够给新的网络架构和协议的设计提供一个良好的实验平台. Anderson 等人[33]首次提出了利用 PlanetLab 打破现有互联网发展僵局，提出以“虚拟化”作为网络“细腰”来支持多种不同网络架构并行发展的思想. Turner 等人[34]提出，作为网络互联的基本设备，路由器需要支持虚拟化的功能，实现“分片”设计的思想. PlanetLab 结点是一台运行虚拟化软件的服务器，为用户提供独立的虚拟机进行编程，保证了不同实验之间良好的隔离，也提供了较好的可编程性. 然而，纯软件实现导致其转发性能不如商业路由器. Egi 等人[35]提出了 vRouter，在 x86 多核处理器平台上采用虚拟机 Xen 和 OpenVZ，实现数据平面软件 Click，通过对数据包收发机制进行优化，使一台通用服务器的转发性能达到了 7 Mpps (Million packets per second, 百万数据包每秒). 受访存延迟的限制，其转发性能难以进一步提升. Anwer 等人[21]设计了一套基于 FPGA 的可编程虚拟路由器框架 SwitchBlade，利用硬件流水线架构解耦包处理中的各个模块，方便用户自定义数据包处理模块，并“插入”到流水线中，以提高数据平面的可编程性，加速新协议的实验和部署. SwitchBlade 基于 NetFPGA 硬件实现，其转发性能远高于 Linux 内核的原始数据包转发性能. 然而，受 NetFPGA 资源的限制，该原型系统可扩展性较差，只能够同时支持 4 个不同的硬件虚拟数据平面. Lu 等人[36]采用 Switch 芯片和普通 Server 相结合，设计了针对数据中心网络应用的高性能可编程路由器 ServerSwitch. 通过 Switch 芯片本身提供的可编程性，满足数据中心网络中新业务的可编程需求；通过普通 Server 提供的丰富计算和存储资源，在网络节点中提供复杂数据包处理（in-network packet processing）能力.

国内关于可编程虚拟路由器的研究主要包括，中国科学院计算技术研究所的PEARL[37]，国防科技大学提出的 NetMagic ⑥，清华大学提出的 OpenRouter[38]和TUNIE[39]. PEARL是基于 FPGA加速板卡和通用服务器的虚拟路由器，采用多核处理器和专用数据包处理卡相结合的方式来构建可编程虚拟路由器. 专用数据包处理卡内嵌TCAM和SRAM存储器，可实现线速路由查找和转发等功能，而高性能多核处理器用于路由管理，利用操作系统级虚拟机LXC实现控制平面虚拟化. 因此，PEARL不仅支持虚拟化和灵活可编程，还具有良好的转发性能. 受 4 个千兆物理网络接口带宽的限制，PEARL原型系统实现了 4 Gbps的转发性能，提高物理接口数量和带宽可以显著提高其转发性能. 国防科技大学提出的NetMagic是支持新服务和协议部署的FPGA加速板卡，提供良好的用户接口方便实验人员进行快速新服务和新协议的部署开发. NetMagic将数据包查找转发操作和协议处理解耦，并规范化两者之间的接口，达到新协议快速设计的目的. NetMagic硬件主要关注可重构的模块化设计以方便用户编程，未提及对硬件数据平面虚拟化的

⑥ NetMagic. http://www.netmagic.org/

表 4 可编程虚拟路由器原型系统研究进展

原型系统	实现平台	资源分配与隔离	可扩展性	可编程性	性能
PlanetLab	Server	虚拟内核	基于服务器的丰富资源实现较好的可扩展性	软件灵活可编程	虚拟化开销导致转发性能下降
vRouter	Server	Xen，OpenVZ	基于服务器的丰富资源实现较好的可扩展性	软件灵活可编程	7 Mpps（受访存延迟的限制）
SwitchBlade	Server, NetFPGA	虚拟硬件数据平面，OpenVZ	受硬件资源限制，虚拟数据平面数量可扩展性较差	模块化流水线架构，降低 FPGA 编程难度	1.4 Mpps（受发包仪发包速率限制）
ServerSwitch	Server, Switch	未提及虚拟化支持	使用 CPU 作为协处理器，扩展复杂的数据包处理功能	Switch 芯片可编程性有限，只支持标准协议	4 Gbps（受 4 个 GE 网络接口带宽限制）
PEARL	Server, FPGA, TCAM	虚拟硬件数据平面，LXC	使用软件数据平面弥补硬件数据平面数量不足的问题	硬件数据平面可编程性有限，软件数据平面灵活可编程	4 Gbps（受 4 个 GE 网络接口带宽限制）
NetMagic	FPGA, External Controller	未提及数据平面虚拟化支持	使用外部控制器（服务器）扩展支持带状态的复杂数据包处理	使用可重构的模块化设计提高 FPGA 数据平面的可编程性	FPGA 数据平面可以实现线速转发
OpenRouter	Commercial router	未提及数据平面虚拟化支持	扩展 OpenFlow 以支持更多协议的数据包处理需求	受 OpenFlow 的限制，无法支持非 IP 协议	基于商业路由器实现，性能未知
TUNIE	Server, NetFPGA, OpenFlow Switch	虚拟数据平面，OpenVZ，Xen	使用 Click 软件数据平面可以弥补 FPGA 虚拟数据平面数量的不足	Click 软件数据平面灵活可编程	FPGA 虚拟数据平面可以实现线速转发

支持. OpenRouter是清华大学提出的针对OpenFlow的扩展方案. OpenRouter主要从三个方面对OpenFlow网络框架进行扩展：控制层面，OpenFlow协议和OpenFlow流表. 在控制层面上，OpenRouter保留了分布式路由协议，让底层的路由设备继续运行OSPF协议；在OpenFlow协议方面，提出了三个TLV扩展：网络状态TLV, 网络配置TLV和流表TLV扩展；在OpenFlow流表上，提出最大化利用交换机的转发表、ACL控制列表、NetFlow的功能，实现了增强的OpenFlow交换机. 然而，受OpenFlow协议本身的限制，OpenRouter仍然难以支持全新网络协议（非TCP/IP协议）的特殊数据包处理需求. 清华大学实现了支持虚拟化的网络创新实验平台TUNIE. 系统采用高性能多核处理器灵活包处理与可编程硬件高性能转发相融合的方式，具有资源灵活可切片、系统深度可编程、配置灵活多样化和系统可扩展等特征与优势.

表 4 综合比较了上述可编程虚拟路由器原型系统研究进展. 基于通用服务器平台实现的可编程虚拟路由器原型（如 PlanetLab 和 vRouter）具有很好的隔离性、可扩展性和可编程性，然而其转发性能较差；基于可编程硬件 FPGA 实现的可编程虚拟路由器（如 SwitchBlade、PEARL、NetMagic、TUNIE）具有很好的转发性能. 然而，受 FPGA 硬件资源以及编程灵活度的限制，其可扩展性和可编程性有待进一步提高；基于 ASIC 芯片（如 Switch）设计的可编程虚拟路由器（如 ServerSwitch 和 OpenRouter）具有比 FPGA 平台更高的转发性能，然而，其可编程灵活性和功能的可扩展性非常有限. 总体来讲，单一硬件平台难以同时满足可编程虚拟路由器中虚拟化（资源隔离性和可扩展性）、可编程性和高性能三大特性需求，异构的硬件平台逐渐得到广泛关注（如 Server 和 FPGA 相结合、Server 和 Switch 相结合）. 如何在异构平台上充分发挥各种硬件平台的优势以弥补对方的不足，设计灵活高性能的可编程虚拟路由器原型系统，仍然值得进一步研究.

5 未来研究方向

虽然目前学术界针对可编程虚拟路由器的研究已取得一定的成果，但是，可编程虚拟路由器功能需求复杂，技术难点和挑战较大，仍然有以下问

题值得进一步研究：

（1）转发表特征研究. 高效的查找转发算法依赖于对转发表特性的充分挖掘，已有研究通过挖掘转发表相似性等特征解决虚拟路由器的存储空间可扩展性问题，通过挖掘路由更新规律提出快速增量更新算法. 然而，随着应用场景的改变，未来虚拟路由器的转发表特征（如相似性、路由更新模式等）可能发生变化. 因此，需要进一步研究虚拟路由器转发表特征及其变化规律，为设计高效的转发表查找算法奠定基础.

（2）高性能数据包处理算法. 可编程虚拟路由器涉及的不仅仅是传统路由器中的查找转发操作，还包括更复杂的网络中数据包处理功能，如 NDN 还要求网络设备提供内容缓存相关的存储和计算功能. 然而，目前关于虚拟路由器数据包处理的研究，主要集中在查找转发算法. 因此，需要结合新型网络体系结构和新协议的需求，研究可编程虚拟路由器中的高性能网络数据包处理算法，在保证高速查找转发的前提下，满足新协议和体系结构对数据包计算和存储处理的特殊需求.

（3）异构平台的资源虚拟化问题. 不同硬件平台具有不同优势，如 FPGA 适合于简单、高性能计算，TCAM 支持高速查找，而多核处理器具有灵活性强，支持复杂计算处理等特点. 在可编程虚拟路由器中，为了更好地满足不同网络应用的性能需求，硬件平台趋于异构化. 然而，对于用户来讲，简单一致的编程模型更加友好. 因此，需要在异构的硬件平台之上，研究资源虚拟化方法，向用户提供一致的编程模型，以简化用户编程的难度.

（4）数据包处理原子操作. 允许用户直接编程可以支持用户实现新的数据包处理功能. 但是，如果能够将数据包处理的流程抽象为若干原子操作的自由组合，将有效提高用户编程的效率. 例如，Click[19]中定义了 IP 数据包处理的一系列基本操作，通过对基本操作模块的自由组合可以实现常见的 IP 数据包处理功能. 但是，Click 不支持虚拟化，也不支持非 IP 数据包处理. 研究可编程虚拟路由器中数据包处理的原子操作，并保证原子操作集合的完备性，将为可编程特性提供一种全新的实现方式，显著提高用户编程的效率.

（5）可编程虚拟路由器的安全问题. 可编程虚拟路由器的灵活可编程特性可以方便用户自定义数据包处理功能，但是，也会因此带来一系列安全隐患. 例如，恶意代码的植入或者蓄意的攻击可能导致整个可编程虚拟路由器平台无法正常工作，甚至影响网络的正常运行；用户的路由规则和安全策略有可能被同一个路由器平台上的其它用户窃取. 因此，需要研究用户编程的安全检查机制，防止恶意攻击或者误操作导致的路由器系统崩溃；也需要研究不同用户之间的数据保护与隔离机制，防止未授权用户非法获取其它用户信息.

（6）可编程虚拟路由器体系结构. 可编程虚拟路由器需要支持虚拟化、可编程和高性能三个特性. 灵活可编程与高性能之间存在矛盾：纯软件的数据包处理易于提供灵活可编程性，但是数据包处理性能较差；硬件数据包处理可以保证高性能，但是可编程灵活性差. 虚拟化与高性能之间也存在矛盾：虚拟化的层次越高（如全虚拟化技术），隔离性越强，但性能越差. 因此，可编程和虚拟化的结合会给数据包处理性能带来严重挑战. 在可编程虚拟路由器体系结构设计时，需要根据不同应用场景的需求，研究数据平面和控制平面的实现平台划分、可编程和虚拟化特性的支持层次划分等问题，从而在虚拟化、可编程和高性能三个特性之间进行折衷.

6 总结

未来网络体系结构的研究、新型协议和业务的实验与部署，迫切需要支持可编程和虚拟化的试验床，可编程虚拟路由器是搭建未来网络试验床的核心设备. 本文分析了可编程虚拟路由器的应用背景和特性需求，讨论了可编程虚拟路由器关键技术和挑战，并详细介绍了其关键技术和原型系统研究进展. 最后，对未来研究方向进行了展望. 从路由器本身的发展来看，可编程虚拟路由器内部结构和关键技术的研究将会持续下去，最终的设计目标是实现与现有商用路由器性能相媲美的支持成百上千个虚拟路由器实例的细粒度的可编程虚拟路由器. 从可编程虚拟路由器的应用场景来看，它将从实验平台应用扩展到数据中心网络乃至企业或者服务提供商内部网络. 在这些应用场景中，将会涌现出更多值得研究的问题，比如说虚拟网络拓扑映射，以及网络设备的集中控制和编程等.

参 考 文 献

[1] Xie G., Zhang Y., Li Z., Sun Y., Xie Y., Li Z., Liu Y. A Survey on Future Internet Architecture. Chinese Journal of Computers, 2012,

35(6): 1109-1119 (in Chinese)
(谢高岗，张玉军，李振宇，孙毅，谢应科，李忠诚，刘韵洁. 未来互联网体系结构研究综述. 计算机学报，2012，35(6): 1109-1119)

[2] Chen K., Hu C., Zhang X., Zheng K., Chen Y., and Vasilakos A. V. Survey on Routing in Data Centers: Insights and Future Directions. IEEE Network, 2011, 25(4): 6-10.

[3] Nanda S. and Chiueh T. A Survey on Virtualization Technologies. RPE Report, February 2005.

[4] Pearce M., Zeadally S., Hunt, R. Virtualization: Issues, security threats, and solutions. ACM Computing Surveys. 2013, 45(2):17:1-17:39.

[5] McKeown N., Anderson T., Balakrishnan H., Parulkar G., Peterson L., and Turner J. OpenFlow: enabling innovation in campus networks. ACM SIGCOMM Computer Communication Review, 2008, 38(2): 69–74.

[6] Lv G., Sun Z., Li T., Mao J., Yang A. LabelCast: A General Abstraction for the Forwarding Plane of SDN. Chinese Journal of Computers, 2012, 35(10): 2037-2047 (in Chinese)
(吕高锋，孙志刚，李韬，毛健彪，杨安. LabelCast: 一种普适的 SDN 转发平面抽象. 计算机学报, 2012, 35(10): 2037-2047)

[7] Anwer M. B. and Feamster N. Building a fast, virtualized data plane with programmable hardware//Proceedings of the 1st ACM SIGCOMM workshop on Virtualized Infrastructure Systems and Architectures. 2009.

[8] Naous J., Gibb G., Bolouki S., and McKeown N. NetFPGA: reusable router architecture for experimental research//Proceedings of the ACM workshop on Programmable Routers for Extensible Services of Tomorrow. 2008.

[9] Reitblatt M., Yi C., and Afanasyev A. Survey and taxonomy of IP address lookup algorithms. IEEE Network, 2001, 15(2): 8–23.

[10] Fu J. and Rexford J. Efficient IP-address lookup with a shared forwarding table for multiple virtual routers//Proceedings of the ACM CoNEXT. 2008.

[11] Srinivasan V. and Varghese G. Fast address lookups using controlled prefix expansion. ACM Transactions on Computer Systems, 1999, 17(1): 1–40.

[12] Luo L., Xie G., Salamatian K., Uhlig S., Mathy L., and Xie Y. A Trie Merging Approach with Incremental Updates for Virtual Routers//Proceedings of the IEEE INFOCOM. 2013: 1246-1254.

[13] Song H., Kodialam M., Hao F., and Lakshman T. V. Building Scalable Virtual Routers with Trie Braiding//Proceedings of the IEEE INFOCOM. 2010: 1442-1450.

[14] Ganegedara T., Jiang W., and Prasanna V. Multiroot: Towards Memory-Efficient Router Virtualization//Proceedings of the IEEE International Conference on Communications. 2011.

[15] Luo L., Xie G., Uhlig S., Mathy L., Salamatian K., and Xie Y. Towards TCAM-based scalable virtual routers//Proceedings of the ACM CoNEXT. 2012: 73–84.

[16] Unnikrishnan D., Vadlamani R., Liao Y., Dwaraki A., Crenne J., Gao L., and Tessier R. Scalable network virtualization using FPGAs//Proceedings of the 18th annual ACM/SIGDA international symposium on Field programmable gate arrays. 2010: 219–228.

[17] Yin D., Unnikrishnan D., Liao Y., Gao L., and Tessier R. Customizing virtual networks with partial FPGA reconfiguration//Proceedings of the second ACM SIGCOMM workshop on virtualized infrastructure systems and architectures. 2010: 57–64.

[18] OpenFlow Switch Specification, version 1.3-rc6. April 11, 2012.

[19] Morris R., Kohler E., Jannotti J., and Kaashoek M. F. The Click modular router//Proceedings of the 17th ACM Symposium on Operating Systems Principles. 1999: 217–231.

[20] Handley M., Hodson O., and Kohler E. XORP: an open platform for network research. ACM SIGCOMM Computer Communication Review. 2003, 33(1): 53–57.

[21] Anwer M. B., Motiwala M., Tariq M. B., and Feamster N. SwitchBlade: a platform for rapid deployment of network protocols on programmable hardware. ACM SIGCOMM Computer Communication Review, 2010, l 40(4): 183–194.

[22] Lu G., Shi Y., Guo C., and Zhang Y. CAFE: a configurable packet forwarding engine for data center networks//Proceedings of the 2nd ACM SIGCOMM workshop on programmable routers for extensible services of tomorrow. 2009: 25–30.

[23] Attig M. and Brebner G. 400 Gb/s Programmable Packet Parsing on a Single FPGA//Proceedings of the ACM/IEEE ANCS. 2011: 12–23.

[24] Unnikrishnan D., Li J., Gao L., and Tessier R. ReClick - A Modular Dataplane Design Framework for FPGA-Based Network Virtualization//Proceedings of the ACM/IEEE ANCS. 2011:145–155.

[25] Russel R. Virtio: towards a de-facto standard for virtual I/O devices. SIGOPS Operating Systems Review, 2008, 42(5): 95–103.

[26] Dobrescu M., Egi N., Argyraki K., Chun B. G., Fall K., Iannaccone G., Knies A., Manesh M., and Ratnasamy S. RouteBricks: exploiting parallelism to scale software routers//Proceedings of the 22nd ACM Symposium on Operating Systems Principles. 2009: 15–28.

[27] Han S., Jang K., Park K., and Moon S. PacketShader: a GPU-accelerated software router. ACM SIGCOMM Computer Communication Review, 2010, 40(4): 195–206.

[28] Xu K., Xu M., Wu J., Wu J. Survey on Routing Lookup Algorithms. Jounal of Software, 2002, 13(1):42-50(in Chinese)
(徐恪，徐明伟，吴建平，吴剑. 路由查找算法研究综述. 软件学报, 2002, 13(1): 42-50)

[29] Wang Y., He K., Dai H., Meng W., Jiang J., Liu B., and Chen Y. Scalable Name Lookup in NDN Using Effective Name Component Encoding//Proceedings of the ICDCS. 2012: 688–697.

[30] Yuan H., Song T., and Crowley P. Scalable NDN Forwarding: Concepts, Issues and Principles//Proceedings of the 21st International Conference on Computer Communications and Networks. 2012.

[31] Peterson L., Anderson T., Culler D., and Roscoe T. A blueprint for introducing disruptive technology into the Internet. ACM SIGCOMM Computer Communication Review, 2003, 33(1): 59–64.

[32] Bavier A., Bowman M., Chun B., Culler D., Karlin S., Muir S., Peterson L., Roscoe T., Spalink T., and Wawrzoniak M. Operating system support for planetary-scale network services//Proceedings of the USENIX NSDI. 2004: 253–266.

[33] Anderson T. E., Peterson L. L., Shenker S., and Turner J. S. Overcoming the Internet Impasse through Virtualization. IEEE Computer, 2005, 38(4): 34–41.

[34] Turner J. S. and Taylor D. E. Diversifying the Internet//Proceedings of the IEEE GLOBECOM. 2005: 755–760.

[35] Egi N., Greenhalgh A., Handley M., Hoerdt M., Huici F., and Mathy L. Towards high performance virtual routers on commodity hardware//Proceedings of the ACM CoNEXT. 2008.

[36] Lu G., Guo C., Li Y., Zhou Z., Yuan T., Wu H., Xiong Y., Gao R., and Zhang Y. ServerSwitch: A programmable and high performance platform for data center networks//Proceedings of the USENIX NSDI. 2011.

[37] Xie G., He P., Guan H., Li Z., Xie Y., Luo L., Zhang J., Wang Y., Salamatian K. PEARL: A Programmable Virtual Router Platform. IEEE Communications Magazine, 2011, 49(7):71–77.

[38] Feng T., Bi J., Hu H. OpenRouter: OpenFlow Extension and Implementation Based on a Commercial Router//Proceedings of the IEEE ICNP. 2011:141-142.

[39] Li Y., Su L., Jin D., Zeng L. TUNIE: A Virtualized Platform for Network Experiment on Programmable Infrastructure//Proceedings of the IEEE ICNP. 2011:125-126.

LUO La-Yong, born in 1986, Ph.D. student. E-mail: luolayong@ict.ac.cn. His research interests include programmable virtual routers, and hardware accelerated network packet processing techniques.

HE Peng, born in 1987, Ph.D. student. E-mail: hepeng@ict.ac.cn. His research interests include software routers, and high performance packet processing algorithms.

GUAN Hong-Tao, born in 1980, Ph.D., associate professor. E-mail: guanhongtao@ict.ac.cn. His research interests include virtualization techniques of computer network and routers.

LI Zhen-Yu, born in 1980, Ph.D., associate professor. E-mail: zyli@ict.ac.cn. His research interests include future Internet architecture and P2P computing.

XIE Gao-Gang, born in 1974, Ph.D., professor, Ph.D. supervisor. E-mail: xie@ict.ac.cn. His research interests include future Internet architecture, programmable virtual routers and network measurement and modeling.

Background

The experiment and deployment of future Internet architectures and new protocols require network innovation testbeds. Programmable virtual router is the key building block for future Internet testbeds. Developing such a router is faced with three main challenges: virtualization, programmability, and performance.

This paper analyzes the challenges, and surveys the relevant research work that aimed to address these challenges. At last, this paper points out the remaining open issues on programmable virtual routers.

This work was supported in part by National Natural Science Foundation of China (NSFC) with Grants 61133015 and 61202411, by National High-tech R&D Program of China with Grant 2013AA013501, by Strategic Priority Research Program of CAS with Grant XDA06010303, and by Instrument Developing Project of CAS with Grant YZ201229.

ICT, CAS started the research on programmable virtual router and future Internet architecture in 2009, and has achieved some promising results, including the programmable virtual router platform (PEARL), and the service-oriented future Internet architecture (SOFIA). More detailed can be found at http://fi.ict.ac.cn.