软件定义网络中基于匹配动作表的 IP 隧道

张克尧[1),2),3)] 毕 军[1),2),3)] 王旸旸[1),3)]

[1)](清华大学网络科学与网络空间研究院 北京 100084)

[2)](清华大学计算机科学与技术系 北京 100084)

[3)](北京信息科学与技术国家研究中心 北京 100084)

摘 要 当前基于 IP 层的隧道技术在网络虚拟化、构建覆盖网络、连接异构网络等方面有着广泛的应用,但是这些传统 IP 隧道在管理配置方面存在不易维护、管理复杂、效率低等问题. 软件定义网络是一种新型网络管控体系结构,它将网络的控制和管理逻辑从网络设备中抽离出来,并提供了开放统一的编程接口,从而大大提升了网络的管理效率. 但作为软件定义网络的重要标准,OpenFlow 原生并不支持 IP 隧道的建立,因此在 SDN 网络中建立隧道依然依赖于传统的配置方式. 该文采用 SDN 中数据平面的匹配动作表编程模型,提出了一种新的 IP 隧道机制——MAT 隧道. MAT 隧道可以通过下发流表规则对隧道报文直接进行封装和解封,不再通过配置隧道端口的方式. 该文基于开源软件交换机 Open vSwitch 和开源控制器 Floodlight 完成了 MAT 隧道原型的实现,并利用 DPDK 对于其性能做了一定优化. 该文还根据真实拓扑搭建了仿真环境,对 MAT 隧道与 Open vSwitch 原有的隧道进行了对比评估,结果显示 MAT 隧道可以将隧道的平均时延降低 10%左右,而采用 DPDK 加速后可以进一步降低 20%左右. 而通过隧道进行路径切换的测试表明,MAT 隧道将隧道切换过程中的最大抖动降低 3 个数量级,同时将对吞吐量的影响降低 50%.

关键词 软件定义网络;OpenFlow;IP 隧道技术;匹配动作表;Open vSwitch

中图法分类号 TP393 **DOI 号** 10.11897/SP.J.1016.2019.00282

A Mechanism of IP Tunneling via Match-Action Table in Software Defined Networking

ZHANG Ke-Yao[1),2),3)] BI Jun[1),2),3)] WANG Yang-Yang[1),3)]

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

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

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

Abstract IP tunneling is a technology for packet encapsulation, which encapsulates the original packets in the payload of IP packets. It has been widely used in the field of network virtualization, overlay network, heterogeneous network and so on. Software Defined Networking (SDN) is a new network management architecture, which extracts the control and management logic from the device, thus promoting the innovation of the network. SDN provides open and unified APIs, which greatly enhances the network management efficiency. The establishment and management of tunnels is an important requirement of many applications in SDN. However, as a significant southbound interface, OpenFlow only supports tag-based tunneling (e.g., MPLS), but does not primitively support the establishment of IP tunnels. As a result, OpenFlow has many restrictions on network application, function and scalability in terms of tunneling. To solve the problem,

收稿日期:2017-11-06;在线出版日期:2018-05-30. 本课题得到国家"十三五"重点研发计划"网络空间安全"专项项目(2017YFB0801701)、国家自然科学基金项目(61472213)资助. **张克尧**,男,1993 年生,硕士研究生,主要研究方向为软件定义网络域间互联机制、可编程数据平面. E-mail: keyaozhang@126.com. **毕 军**(通信作者),男,1972 年生,博士,长江学者特聘教授,国家"863 计划"首席科学家,博士生导师,主要研究领域为新型网络体系结构(软件定义网络、网络功能虚拟化和网络空间安全体系结构). E-mail: junbi@tsinghua.edu.cn. **王旸旸**,男,1979 年生,博士,博士后,主要研究方向为互联网体系结构、软件定义网络、网络测量等.

data plane which supports OpenFlow usually adopts the approach of traditional configurations, which provides various of vendor-dependent configure commands and programmable APIs, rather than a unified standard interface. But these commands or APIs are different on different targets. Therefore, IP tunneling is not actually simplified in SDN, suffering from maintenance difficulty, management complexity, and low flexibility. Inspired by the Match-Action Table programming models in OpenFlow, we argue that expressing tunneling logic with the MAT model could improve the programmability and flexibility. We propose a mechanism of IP tunneling based on Match-Action Table in SDN, called MAT tunnel. The MAT tunnel can encapsulate and decapsulate directly by real-time installing flow rules instead of manually configuring tunnel ports. We extend the Match and Action Fields in OpenFlow so that the controllers can install flow entries about MAT tunnel on the switches. We also provide RESTful API on controllers for network applications and administrators, which makes it easier to create or remove the MAT tunnel. In addition, we introduce an ARP proxy on the controller to deal with the problem of layer 3 connectivity between MAT tunnel endpoints and traditional gateways. This paper implements the MAT tunnel prototype based on Open vSwitch and Floodlight controller, including VxLAN and GRE tunnels. In our implementation, the first packet of a new flow will be sent to user space, and the following packets of the flow will just be handled in kernel, not going through user space, which can improve the performance. And then, we further enhance the data plane performance of the MAT tunnel using DPDK. This paper also constructs a simulation network environment based on a real ISP topology from the topology zoo dataset. Comparing traditional tunnels, we find that the MAT tunnel can reduce the average delay by 10 percent, which can be further reduced by about 20% with DPDK. In addition, to evaluate the efficiency of MAT tunnels, we conduct tests in which we switch flow traffic between two different paths by MAT tunnels. This tunnel path switching tests show that the MAT tunnel can significantly decrease the maximum jitter by 3 orders of magnitude and reduce the throughput loss by 50%. These results indicate that the MAT tunnel can effectively reduce the cost of creation and revocation of IP tunnels.

Keywords　software defined networking; OpenFlow; IP tunneling; match-action table; Open vSwitch

1　引　言

IP 隧道技术是一种数据包封装技术，它是将原始数据包封装在另一个 IP 报文的载荷中进行传输. 如图 1 所示，IP 报文的载荷既可以在网络层之上(over IP)，也可以在传输层(over TCP/UDP)之上，甚至可以是应用层之上.

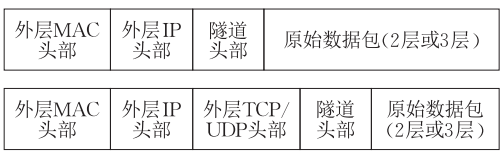


图 1　IP 隧道报文

IP 隧道技术在现有网络环境中有广泛的应用. 例如，IP 隧道技术在网络虚拟化环境中，起到隔离用户和资源的作用[1]；在互联网上通过隧道技术建立覆盖网络[2-3]，提高端到端路径性能；通过隧道跨越底层物理网络，连接与底层网络异构的新型网络(比如 ICN[4]、IPv6 网络等). 然而传统 IP 隧道在管理配置方面存在管理复杂、效率低等问题.

软件定义网络(Software Defined Networking，SDN)是一种新型网络管控体系结构，它将网络的控制和管理逻辑从设备中抽离出来，并向用户提供开放的可编程接口从而加速网络的创新[5]. 由于 SDN 的数据平面提供了开放统一的编程接口，大大提升了网络的管理效率. 近年来，SDN 在校园网、企业网、数据中心等都得到了广泛的应用.

在 SDN 中，隧道的建立和管理是重要的应用需求. 比如 SDN 方式的数据中心网络虚拟化. 而另一方面，现有的 SDN 大多部署在网络边缘，在现有的

IP 互联网中以“孤岛”的形式存在，因此如果想把这些 SDN 部署域在数据平面互联起来，往往需要依赖 Overlay 技术，并通过 IP 隧道技术实现. 但作为 SDN 的重要标准，OpenFlow[6] 只支持基于标签的隧道(例如 MPLS)，而并不支持 IP 隧道的建立. 这使得 OpenFlow 在网络应用和功能的支持上，以及基于 OpenFlow 技术的 SDN 网络本身的扩展上都受到了很多限制. 例如在网络虚拟化场景，或者 SDN 网络跨域互联的场景下，单纯依赖 OpenFlow 的当前的技术无法满足灵活建立 IP 隧道的需求.

为了支持 IP 隧道的功能，OpenFlow 交换机在具体的实现过程中，依然会延用由厂商主导定义的配置命令或编程接口，而没有一套统一的标准. 例如，在被广泛使用的软件交换机 Open vSwitch (OVS)[7] 中，提供 OVSDB[8] 的接口用于管理隧道相关的功能. 而在更多的 OpenFlow 硬件交换机中，包括 HP、Dell 等，大多只能通过 CLI 的接口进行手动配置. 因此 OpenFlow 并没有真正意义上提高 IP 隧道的管控效率，其主要存在的问题包括：第一，配置复杂. 不同交换机会存在配置接口和需求的差异，配置效率低下而且容易出错；第二，难以维护. 控制器并不能获得这些 IP 隧道的状态配置信息；第三，缺乏灵活性. 在 SDN 网络中隧道动态性高的场景下，隧道的建立和拆除需要在流表之外的接口实现，网络运行者需要同时控制多套编程接口(例如流表、不同交换机的不同远程配置接口)，因此不够灵活、管控效率低.

受到 OpenFlow 中匹配动作表(Match-Action Table，MAT)编程模型的启发，我们认为同样可以将隧道的逻辑使用 MAT 模型表达出来，从而提高 IP 隧道的可编程性和灵活性. 由此我们提出了一种 SDN 中基于匹配动作表的 IP 隧道机制——MAT 隧道. 通过对 OpenFlow 进行扩展，MAT 隧道可以利用统一的编程接口对数据平面隧道相关的功能进行管控，也可以实现 SDN 数据平面跨域高效互联.

本文的主要贡献包括：

(1) 提出了一种基于匹配动作表的 IP 隧道机制，它允许控制器通过安装流表的方式指定封装和解封的动作以及隧道相关的参数，而不是通过配置隧道端口的方式实现. 在报文匹配相关流表之后，交换机根据指定的参数执行隧道相关的动作.

(2) 在开源软件交换机 Open vSwitch 和 SDN 控制器 Floodlight 中完成了 MAT 隧道的原型实现，包括 VxLAN[9] 和 GRE[10] 两种隧道类型. 此外我们还利用 DPDK 对 MAT 隧道原型实现做了进一步的性能优化.

(3) 根据真实拓扑搭建了仿真网络环境，在时延和吞吐方面对比评估了 MAT 隧道和 OVS 标准配置方式的隧道. 实验结果表明 MAT 隧道可以降低转发时延，提高转发效率. 而采用 DPDK 进行性能优化后，可以进一步降低 MAT 隧道的转发时延. 另一方面，MAT 隧道能大幅降低隧道切换过程中的抖动以及对吞吐量的影响，这表明 MAT 模式可以有效降低隧道创建和拆除过程的开销.

本文第 2 节介绍 IP 隧道及其与 SDN 相关的研究工作；第 3 节介绍 MAT 隧道的系统设计；第 4 节介绍 MAT 隧道在 OVS 中原型实现，以及采用 DPDK 对其做进一步的性能优化；第 5 节针对 MAT 隧道的性能做了实验评估；第 6 节对本文的工作进行总结和展望.

2 相关工作

本节我们将介绍 IP 隧道以及其在 SDN 网络中应用的相关研究工作.

基于 IP 的隧道技术在各种网络环境中有广泛的应用. 首先，IP 隧道技术被广泛用于云服务提供商和企业的数据中心网络虚拟化中[11]. 数据中心通常承载了多租户的多种应用服务，内部组网环境和需求复杂. 通过隧道封装，数据中心内为不同租户或者不同应用构建虚拟覆盖网络，实现不同租户不同应用的组网互连、网络(空间和资源)隔离、虚拟主机自由编址和移动. 实现网络虚拟化的隧道技术有工业界广泛支持的 VxLAN 技术[9]. VxLAN 隧道把二层 Ethernet 帧封装在 IP-UDP 报文载荷里. 微软开发了 NVGRE 技术[12]，它利用 GRE 头部携带 24 位的虚拟子网标识符信息(VSID). 互联网国际标准组织 IETF 专门成立 NVO3(Network Virtualization Overlays)工作组，针对以 IP 为底层网络设施的数据中心网络的复杂场景需求，研究数据中心网络内构建虚拟网络的协议或者扩展协议标准，为虚拟网络多租户和工作负载移动提供 2 层或者 3 层服务. 比如正在进行的一般化网络虚拟化封装协议 Geneve[13]，以及对 VxLAN 的通用性扩展 VxLAN-GPE[14]，支持多种协议封装. 但是 NVO3 工作组不考虑在数据中心环境中基于扩展 BGP 和 LISP 方

案的网络虚拟化.

IP 隧道技术不仅应用在数据中心企业网内，在 IP 基础设施的互联网中也有许多应用. Touch 等人在 IETF 草案[15]中总结了在互联网体系结构中的 IP 隧道技术. 该草案总结了已有支持 IP 隧道的协议和面对的问题挑战，讨论了 IP 隧道协议设计中需要考虑的关键问题，比如 IP 报文分段对 ID 字段的要求和隧道会遇到的最小的 MTU 问题. Peter 等人在文献[16]中提出了一种互联网上跨自治域级别的转发路径隧道服务. 该服务由互联网运行商 ISP 提供. 用户通过特定的信号在 ISP 的自治域网络之间部署建立隧道，把多个域间路径段连接起来，从而实现采取不同于 BGP 选择的最优路由路径的端到端路径来到达指定的网络目的. 在该隧道设计中，在隧道转发过程中采用逐跳(hop-by-hop)方式的安全认证. 通过该隧道，可以实现域间流量绕过传统 BGP 路由路径，避免域间链路拥塞或者故障等问题，也可以减少 BGP 路由劫持对流量转发路径带来的影响. LISP 协议[17]提供了一种新型互联网路由体系结构，受到了工业界和学术界的关注. 它把互联网核心网和边缘网地址空间分离，通过 LISP 映射-封装隧道方式互联边缘网络，以减少核心网路由表，并且可以带来流量工程、多路径路由等灵活性优势.

IP 隧道技术在软件定义网络中也有广泛的应用. SD-WAN 是将 SDN 技术应用到广域网的场景中的一种服务，它可以用于连接广阔地理范围的企业网络，包括企业的分支机构及数据中心. 在现有的 SD-WAN 实现中，包括思科、Viptela、Versa、华为、中兴、凌锐蓝信、大河云联等公司提供的解决方案，常常通过 IP 隧道在当前的互联网上构建 Overlay 网络，从而使用多个 WAN 链路来提高应用性能，简化 WAN 架构，减少对 MPLS 的依赖. Google 构建的 B4 网络[18]是 SD-WAN 的一个典型案例. B4 网络是一个私有广域网，它连接分布在全球各地的 Google 数据中心站点. 为了更好地利用和管理站点之间的网络带宽利用率和延迟，站点之间采用 IP-in-IP 隧道封装解封跨越 IP 互联网，构建站点之间的互联网络. B4 交换机有 3 种类型：(1)封装交换机用来发起隧道封装和流量分流；(2)传送交换机用于根据外层包头转发报文；(3)解封交换机实现隧道的终止和解封，之后使用常规路由转发报文. 通过 SDN 中央控制器集中控制的流量调度，B4 网络实现了站点之间流量均衡和链路带宽最优利用. Rodriguez-Natal 等人[19]提出采用 LISP 协议作为软件定义网络的一种南向接口协议，以集中控制方式管理 LISP 隧道映射表，以及引入重封装隧道路由器，把 LISP 隧道连接起来构建虚拟网络，实现网络隔离、流量工程. 作者认为：(1) LISP 的映射系统保存了网络控制状态，为 LISP 系统提供了良好的可扩展性，SDN 可以利用映射系统建立可扩展的网络状态数据库，用于 SDN 数据平面和控制平面设备对网络状态的查询；(2) LISP 提供了灵活的名字空间和映射封装的 LISP 隧道，有利于在域间部署建立 Overlay 网络. Heinonen 等人[20]针对基于 SDN 的蜂窝核心网络，提出了新型的动态隧道交换技术，它引入了一个虚拟化演进分组核心，并通过动态 GPRS 隧道协议在使用通用硬件的云环境和专用硬件的快速路径之间切换活动会话的移动锚点.

上述已有工作包含了多种隧道技术和应用，但都仍然以配置方式构建隧道. 在面对大规模、异构和动态场景时仍然存在问题与挑战. 微软的 Firestone[21]在公有云的环境中，提出了一种虚拟交换机平台 VFP，它根据微软的实际运营经验，针对可编程虚拟交换机提出了若干个设计目标. 其中就提到希望基于 MAT 模型表达尽可能多的网络功能，包括隧道的封装解封装过程，将尽可能多的控制逻辑上交给控制器，而只把核心的数据平面留在虚拟交换机中. VFP 认为通过可定义的封装和解封装规则，可以更加灵活地定义一个虚拟网络. 但 VFP 并没有针对如何用 MAT 模型表达隧道功能进行细化地设计和评价.

在本文中，我们提出基于 OpenFlow 协议的 MAT 模型的隧道，并且在 Open vSwitch 上实现了原型，对该隧道机制的转发性能和路径切换性能进行了详细地评价. 本文提出的 MAT 隧道机制，屏蔽了 SDN 数据平面设备的异构配置方式. 该隧道机制可以应用于 SDN 网络中，使得 SDN 控制器能够以可编程的方式灵活地在数据平面实现隧道的管理调度.

3　系统设计

3.1　总体设计

本节我们将介绍 MAT 隧道的总体设计. 如图 2 所示，本系统的数据平面包含经过扩展的支持 MAT 隧道的 SDN 交换机. 在这些交换机中，数据包可以通过匹配流表实现隧道相关的特性与功能，并通过流表的动作实现隧道报文的封装与解封.

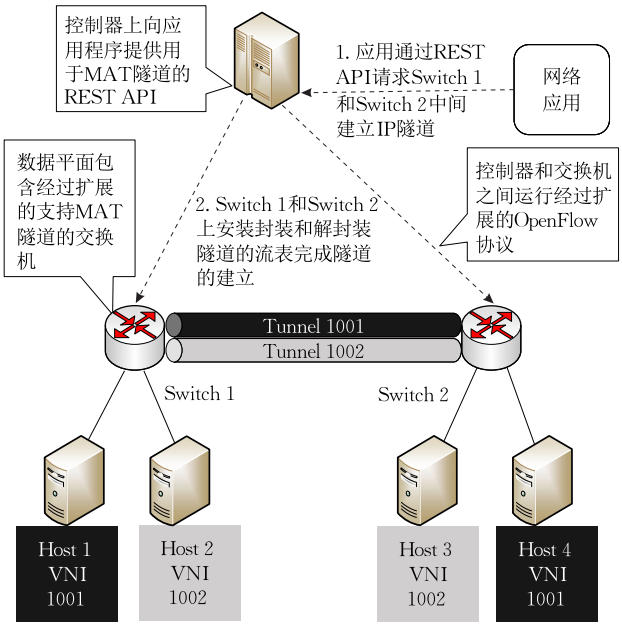


图 2 MAT 隧道总体设计

控制平面的控制器与数据平面的 SDN 交换机之间运行经过扩展的的 OpenFlow 协议. 我们扩展了 OpenFlow 协议中的 Match 和 Action 字段,这样在 Flow_Mod 消息中就能指定新的 Match 和 Action 类型,从而保证控制器能够向 SDN 交换机下发隧道相关的流表.

考虑到在 SDN 网络中,网络管理员常常会使用 REST 应用管理网络,本系统也在控制器上抽象出了用于隧道功能的 REST API. 通过 REST API,用户可以更加方便地使用 Python 或者 Shell 等脚本语言实现隧道相关流表的下发与删除,从而完成隧道的建立与拆除.

以图 2 的情景为例,租户 A 使用 Host 1 和 Host 4 建立虚拟网络 1,租户 B 使用 Host 2 和 Host 3 建立虚拟网络 2. 我们为虚拟网络 1 和虚拟网络 2 分配的 VNI 分别为 1001 和 1002. 网络管理员通过 REST API 请求在 Switch 1 和 Switch 2 两个交换机之间建立两条 IP 隧道,这些 IP 隧道以扩展的 OpenFlow 流表的形式下发到交换机上,并由这些流表完成隧道的功能.

MAT 隧道旨在通过扩展 OpenFlow 作为统一的编程接口管理数据平面设备上的 IP 隧道功能. 但是在其具体设计中需要解决以下问题:

(1) 如何通过匹配动作表表达 IP 隧道的逻辑,数据平面的交换机如何根据这样的匹配动作表工作. 其中一个最核心的问题是,如何根据表项实现隧道的封装和解封装功能.

(2) 现有的用于网络虚拟化的 IP 隧道都能提供资源隔离,MAT 隧道如何支持隔离性.

(3) 在绝大多数的 OpenFlow 软硬件实现中,被加入到 OpenFlow 实例中的物理接口将会以“哑”的二层端口方式工作,他们只会根据流表进行匹配和执行相关动作,其自身没有 IP 地址,从这些端口收到的数据包也不会经过三层网络协议栈的处理,因此我们需要解决 MAT 隧道与传统互联网关的三层互通问题,主要就是处理 MAT 隧道相关的 ARP 消息.

下面我们分别详细说明 MAT 隧道的设计中如何解决这些关键性问题.

3.2 MAT 隧道对匹配动作表的扩展

本节将介绍如何对 OpenFlow 中的匹配动作表进行扩展从而表达隧道的功能逻辑.

OpenFlow 协议作为控制器平面与数据平面之间的通信接口,其工作方式基于 MAT 模型. 在该模型中,数据包头部域的一个子集会与一张或多张表进行掩码匹配,而每个匹配的表项中则指定了相应的动作,这些动作最终会被应用到该报文中,从而完成处理过程. 匹配动作表构成了 OpenFlow 中报文处理的管道(Pipeline),并有效地支持了 OpenFlow 灵活可编程的特性. 这样的匹配动作表标识了一组具有相同特征的数据包,因此在 OpenFlow 中也被称为流表.

但是现有 OpenFlow 协议中的匹配动作表却并不支持隧道报文的处理. 因此在当前的软件交换机实现中,在两个交换机之间建立 IP 隧道需要通过人工或者 OVSDB 等方式进行配置,生成一个虚拟的隧道端口,根据隧道端口类型的不同,实现不同隧道报文的封装和解封,同时隧道的控制信息和策略则保存在交换机本地.

为了充分利用 MAT 模型所带来的灵活性,论文扩展了 OpenFlow 协议中的流表用以支持 IP 隧道的功能. OpenFlow 消息中使用 TLV(Type, Length 和 Value)的形式对网络数据进行重组,能够支持对于流表结构的灵活扩展. 如图 3 所示,MAT 隧道对于流表中 Match TLV 和 Action TLV 进行了扩展. 在 Match TLV 中我们增加了新的 Match 类型去匹配隧道头部的字段,例如 VxLAN 隧道的 VNI. 而在 Action TLV 中增加了两类 Action 分别用于隧道报文的封装与解封,例如对于 VxLAN 隧道增加了 PUSH_VXLAN_TUNNEL

和 POP_VXLAN_TUNNEL 两个 Action. 其中用于封装的 Action 中还定义了封装隧道的参数，例如 src_ip、dst_ip 和 vni 等. 当完成 Match TLV 和 Action TLV 的扩展后，在 Flow_Mod 消息中就可以指定新增的 Match 和 Action 类型.

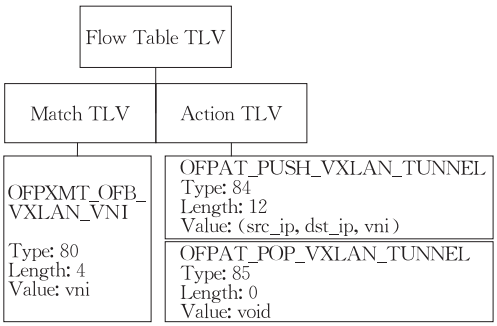


图 3　MAT 隧道对 OpenFlow 协议的扩展

在 MAT 隧道中，隧道的建立以流表下发的方式，而非配置的方式. 建立隧道时，MAT 模型下需要在两端交换机下发总共四条流表，每个交换机上包含一条隧道报文封装动作及一条隧道报文解封动作的流表，不需要生成额外的虚拟端口. 交换机流表中不同的动作类型，对应不同的隧道类型. 对应于传统方式的隧道，隧道的配置信息以动作的参数体现，交换机根据匹配动作表完成隧道端点（Tunnel Endpoint，TEP）的功能. 隧道的控制信息和策略通过控制器进行统一的管理，从而将隧道的控制与转发相分离.

MAT 隧道并没有增加流表的条目数. 在现有 OpenFlow 交换机的隧道实现中，同样需要在两端交换机下发四条流表. 每个交换机上的两条流表项负责表示所生成的虚拟隧道端口与其他物理端口收发两个方向上的转发逻辑. 而 MAT 隧道可以将隧道封装和解封装的动作与转发逻辑放在一个流表项中进行表达，因此 MAT 隧道本身没有额外引入表项，并不会影响系统的可扩展性.

3.3　MAT 隧道的隔离性支持

本节将介绍如何利用 MAT 隧道支持资源隔离.

在数据中心虚拟化网络中，隧道建立的场景常常需要满足提供租户隔离性的需求. 隧道的隔离性保证了不同租户的流量无法互通. 以 NVGRE 和 VxLAN 为代表的用于虚拟化的隧道协议都提供了隔离性支持，两个租户在三层可能使用相同的地址集合，但是他们之间并不能相互访问.

在 MAT 模型中，同样可以支持隧道的隔离性. 如图 4 所示，以 VxLAN 隧道的隔离性为例. 首先控制器根据管理的策略给每个主机分配 VNI，不同租户应具有不同 VNI. 当两台主机间请求建立隧道时，控制器首先检查两台主机是否属于同一个 VNI，如果不是，则直接拒绝该隧道的建立. 如果是，则控制器会在 VNI-MAC-VTEP IP 表中添加两条表项，分别是源主机 VNI-源主机 MAC-隧道入口交换机虚拟 IP、目的主机 VNI-目的主机 MAC-隧道出口交换机虚拟 IP. 而后控制器向隧道入口和出口交换机下发流表. 入口交换机上的流表匹配入端口、源 MAC、目的 MAC、目的 IP 地址，而动作则包括新增的隧道封装动作，例如 PUSH_VXLAN_TUNNEL，该流表需要传入源 IP 地址、目的 IP 地址和 VNI 等参数. 出口交换机上的流表匹配源端口、源地址、目的地址以及 VNI，动作则包括了新增的隧道解封装动作，例如 POP_VXLAN_TUNNEL.

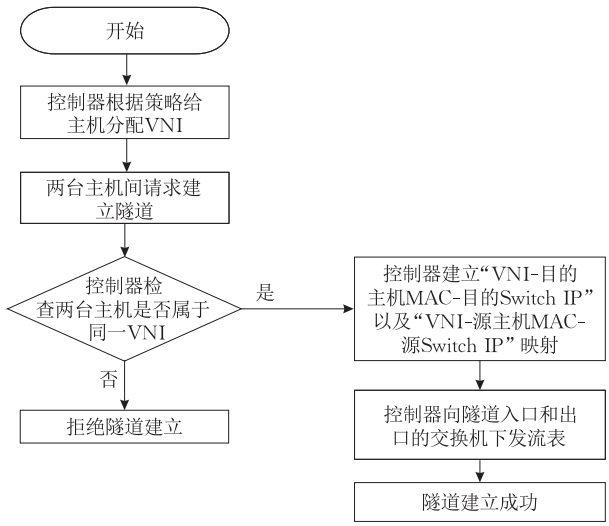


图 4　MAT 隧道的隔离性支持

如前面所述，一条双向的基于 MAT 模型的隧道需要下发 2×2 条流表规则，在表 1 中展示了图 2 的场景中，Host 1→Host 4 方向的两条流表. 由于在 Switch 1 和 Switch 2 上有两个虚拟网络，因此需要建立两条隧道. 对于不同的 VNI，控制器下发了不同的流表动作，由于控制器知道每个 VNI 对应交换机的哪些端口，因此只是将报文送到对应的端口，从而实现了不同虚拟网络之间的隔离. 在 MAT 隧道的工作模式下，由于两端交换机的数据端口都工作在二层，因此我们在这里引入了虚拟 IP 地址作为隧道报文外层封装的 IP 地址，在后面的内容中会作进一步的详细说明.

表 1 MAT 隧道中隧道端点交换机的流表规则

名称	Match	Action
隧道入口交换机	inport=from host1 src_mac=host 1 MAC dst_mac=host 4 MAC	push_vxlan_tunnel= (src_ip=switch 1 虚拟IP地址 dst_ip=switch 2 虚拟IP地址 vni=1001) output=switch 1 外网接口
隧道出口交换机	inport=switch 1 外网接口 src_ip=switch 1 虚拟IP地址 dst_ip=switch 2 虚拟IP地址 udp_port=4789(VxLAN) vni=1001	pop_vxlan_tunnel output=to Host 4(属于网络1的所有端口)

需要说明的是,以上给出的只是 MAT 隧道机制用于 VxLAN 的一个实例.在实际应用中,并不一定是通过 MAC 地址与 VNI 映射,例如也可能将 VLAN 映射到一个 VxLAN 上,这时候只需要对流表中的匹配项作一些修改.MAT 隧道的机制既可以用于主机间的隧道,也可以用于交换机之间的隧道.其下发的表项数量依赖于具体的应用场景.但如前所述,MAT 隧道机制本身并没有引入额外的流表条目,只是增加了匹配的项目和动作的列表,因此并没有对流表的查找和流表空间产生额外的压力.

3.4 MAT 隧道的工作流程

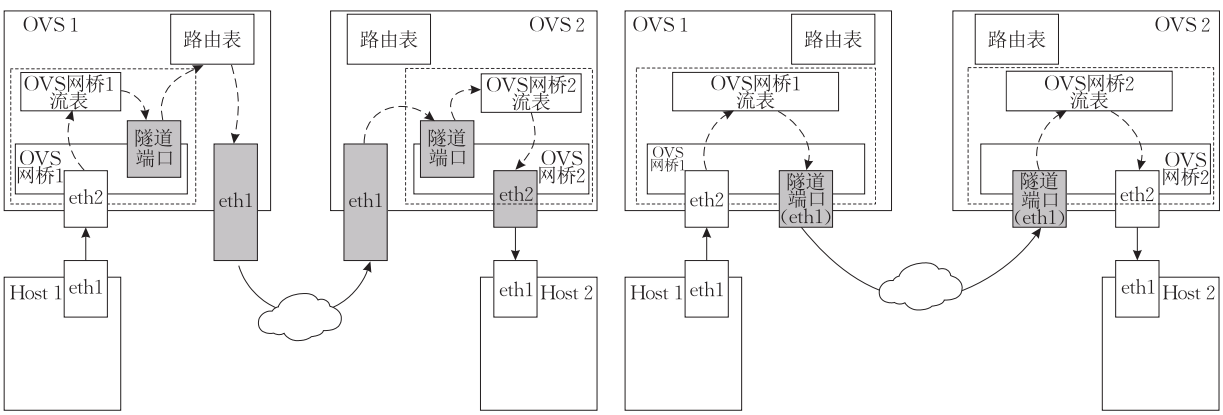
本节我们将详细介绍 MAT 隧道中,数据平面的交换机内部如何处理隧道报文.

在传统 OpenFlow 交换机 IP 隧道的功能实现中,一般需要创建一个隧道端口,并将其加入到 OpenFlow 实例中.这个隧道端口可以根据配置信息完成隧道的封装和解封功能.如同前面所提到的,OpenFlow 实例中的端口并不具有三层能力,因此封装完成的隧道报文一般会根据交换机上的路由表被转发到 OpenFlow 实例外的具有三层能力的物理端口.

我们以软件交换机 OVS 为例说明隧道报文的转发过程.如图 5 所示,假设 Host 1 与 Host 2 属于不同物理网络进行通信,需要在 OVS 1 和 OVS 2 之间建立隧道.图中的 OVS 网桥在这里就相当于交换机中的一个 OpenFlow 实例.

如图 5(a),我们使用 OVS 所实现的 IP 隧道,首先需要在 OVS 1 和 OVS 2 上进行配置,生成隧道端口,并将隧道端口加入到 OVS 网桥中.Host 1 发送报文时,原始报文通过 OVS 网桥匹配流表并被送至隧道端口,从而被封装成相对应的隧道报文.封装完成的数据包而后被主机网络协议栈处理,匹配路由表后从 OVS 网桥之外的物理端口 eth1 发出.隧道报文经过外部的 IP 网络最终被路由至 OVS 2 的物理端口 eth1,OVS 2 解封外部的 MAC 和 IP 头部,并根据隧道类型送给相应的隧道模块,隧道模块解封装隧道头之后送到 OVS 2 的网桥,而后报文匹配网桥上的流表后最终送到 Host 2.在报文传送过程中,发送端进行了两次端口转发、一次流表匹配、一次协议栈路由表匹配,接收端进行了两次端口转发、一次流表匹配.

而如果采用 MAT 隧道时,如图 5(b)所示,它不需要依赖额外的隧道端口.其物理接口 eth1 将被直接加入到 OVS 网桥中.需要注意的是,当这个物理接口被加入到 OVS 网桥后,它将只会工作在二层,而不再具有三层的 IP 地址.因此控制器需要在封装的动作中为隧道的两端分配可路由的虚拟 IP 地址,从而保证隧道两端的交换机在互联网上的可达性.同时控制器也需要处理此虚拟 IP 地址的 ARP 消息(详见下一小节).当 Host 1 向 Host 2 发送报文时,原始报文经过 OVS 网桥,通过匹配流表直接封装成对应的隧道报文并从 eth1 端口送出.隧



(a) OVS实现的隧道 (b) MAT隧道

图 5 隧道报文在 Open vSwitch 中的处理流程

道报文到达 OVS 2 之后匹配网桥上的流表，而后对隧道报文进行解封，并从 eth2 端口送出，最终送达 Host 2. 在报文转送过程中，发送端进行了一次端口转发，一次流表匹配，在接收端进行了一次端口转发、一次流表匹配.

从发送和接收两端交换机的处理流程看，MAT 隧道简化了报文转发的流程，可以减少内核协议栈的开销.

3.5　MAT 隧道与 IP 网关的三层互通

本节我们将介绍如何解决 MAT 隧道端点与传统网关设备的三层互通的问题.

如同上面所提到的，在 MAT 隧道中，当物理端口工作在 OpenFlow 实例下，以二层方式收发数据包，其本身没有 IP 地址. 控制器为隧道两端的交换机分配了可供路由的虚拟 IP 地址以及对应的虚拟 MAC 地址，并在控制器上维护一个 IP-MAC 绑定表. 当进行隧道的封装时，控制器会告诉 OpenFlow 实例隧道两端的 MAC 地址以及默认网关的 MAC 地址，这样它就会知道如何封装隧道报文的外层 MAC 头部.

如果隧道入口和出口交换机在同一个子网，那么这样的工作方式并不会存在问题. 但是如果一旦跨越子网，隧道端点的外部端口因为工作在二层无法处理 ARP 消息，因此与其相连的传统 IP 网络的网关就无法获取到这个虚拟 IP 地址对应的 MAC 地址，它会认为这个虚拟 IP 不可达. 因此我们就在控制器上设计了一个 ARP 代理来处理有关隧道端点的 ARP 消息.

如图 6 所示，如果隧道两端的 SDN 交换机跨越三层网络，则在隧道入口处，所封装的隧道报文外部目的 MAC 地址变为入口交换机所连接网关的 MAC 地址. 该隧道报文通过三层网络最终被路由至隧道出口交换机所连接的网关，此时网关会发送 ARP 请求，ARP 请求的目标 IP 地址就是隧道出口交换机的虚拟 IP 地址，该 ARP 请求会在隧道出口交换机以 Packet_In 的消息发送给控制器，控制器收到 ARP 请求后，构造 ARP 响应报文，填充目标硬件地址为隧道出口交换机的虚拟 MAC 地址. 该 ARP 响应报文以 Packet_Out 消息发送给隧道出口交换机，从原请求的入端口送出，并由网关接收. 网关完成隧道报文外部的二层帧封装并送到隧道出口交换机，从而完成隧道报文的解封装.

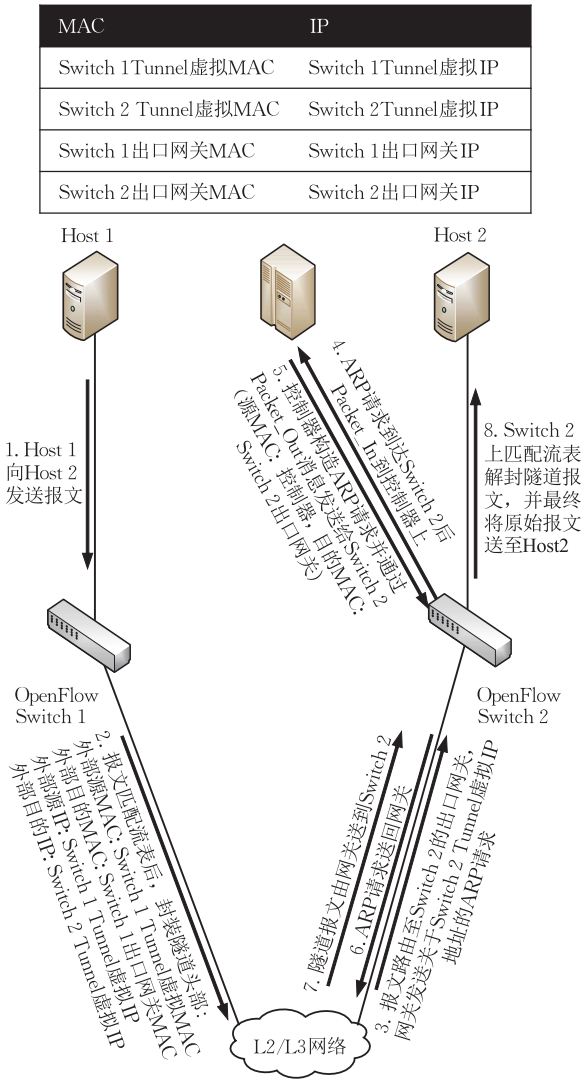


图 6　MAT 隧道与 IP 网关的三层互通

4　系统实现

MAT 隧道给出了一种 OpenFlow 设备支持 IP 隧道的扩展方案，这种机制本身既可以用于软件交换机，也可以用在硬件平台.

我们基于开源软件交换机 Open vSwitch 2.6.1 提供了 MAT 隧道的一种原型实现，包含了 GRE 和 VxLAN 两种 IP 隧道. 同时，我们还基于开源控制器 Floodlight 1.2 提供了 MAT 隧道相关的 REST API 支持. 该项目主页为 https://github.com/mat-tunnel.

我们下面主要介绍 OVS 中实现 MAT 隧道的细节. MAT 隧道的核心功能在于隧道报文的封装与解封. 在原型实现中，我们需要尽可能高效地实现这样的功能.

OVS 在设计时包含了用户空间和内核空间两

个部分.我们在处理隧道报文的时候也充分利用了OVS的这种特性,从而提高隧道报文的处理逻辑.

如图7所示,数据报文经过协议栈被送至OVS的内核空间.OVS首先会对报文进行解析,提取出报文的头部信息,然后根据提取出的头部信息首先查询内核态的Datapath流表.对于一个流的首包,在这个内核态的流表中是查不到与之匹配的表项的.这时数据包会以NETLINK消息的方式上交到用户空间.用户空间同样会对数据包进行解析和匹配.一般地,在用户空间的OpenFlow流表中找到匹配项之后,就会获取到流对应的动作,这时候它会对这个动作做进一步的解析,并在用户空间执行.经过扩展后,这里的动作就会包含隧道报文的封装和解封装,不同的隧道类型对应着不同的动作.例如对于VxLAN隧道的封装,其对应的动作由用户指定隧道两端的IP地址以及VNI等信息作为参数,隧道封装动作的执行过程会依次添加VxLAN头部、外层UDP头部、IP头部以及MAC头部.解封过程则恰好相反.这条路径被称为"慢速路径".但如果每个报文都需要从内核发送到用户空间执行,效率就会很低.因此当首包在用户空间处理完成之后,它会将用户空间匹配的OpenFlow流表转化为内核中使用的Datapath流表,并以NETLINK消息的方式发送给内核.当这之后的数据包到达交换机后,在内核态的Datapath流表中就会包含匹配的项目,并执行其对应的动作.这时候就不必再将报文发送到用户空间,而可以完全在内核态中处理完成.我们在内核空间中也实现了隧道封装解封的动作.这条处理路径被称之为"快速路径".

还有一种情况是当用户空间也没有匹配的流表项时,OVS会向所连接的控制器发送Packet_In消息,控制器则以Flow_Mod的消息向OVS下发流表规则,OVS需要将控制器下发的流表规则进行解析并存储到用户态的OpenFlow流表中.同样,我们在流表解析的过程中增加了对于隧道相关的匹配和动作项支持,从而保证了能够正确添加隧道相关的流表规则.

我们同样也使用DPDK技术进一步提高MAT隧道的性能.如图8所示,在使用DPDK后,PMD Driver会以轮询而非中断的方式从网卡获取数据包并将其直接放到用户空间中.这时数据包不会再经过内核中的Datapath,而是直接进入用户空间的Datapath.因此我们在这个DPDK的Datapath上也添加了对于MAT隧道的支持.

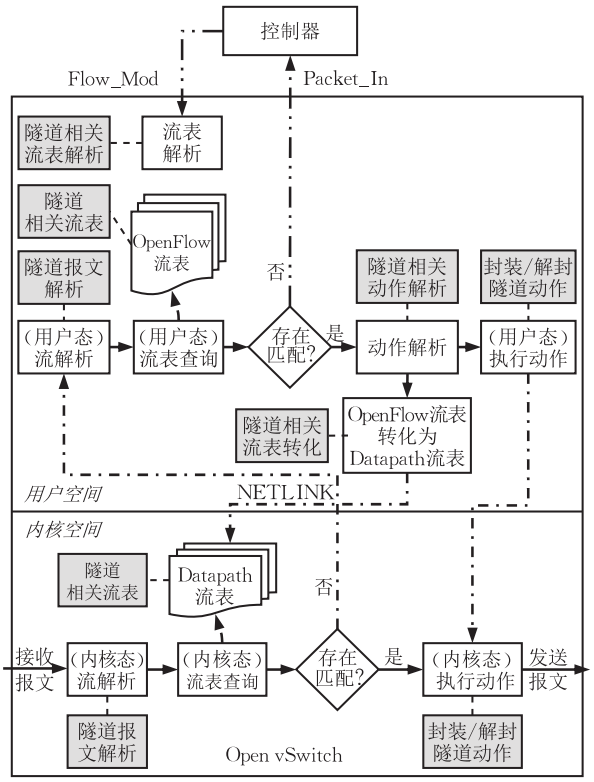


图7 隧道报文在Open vSwitch内的处理流程

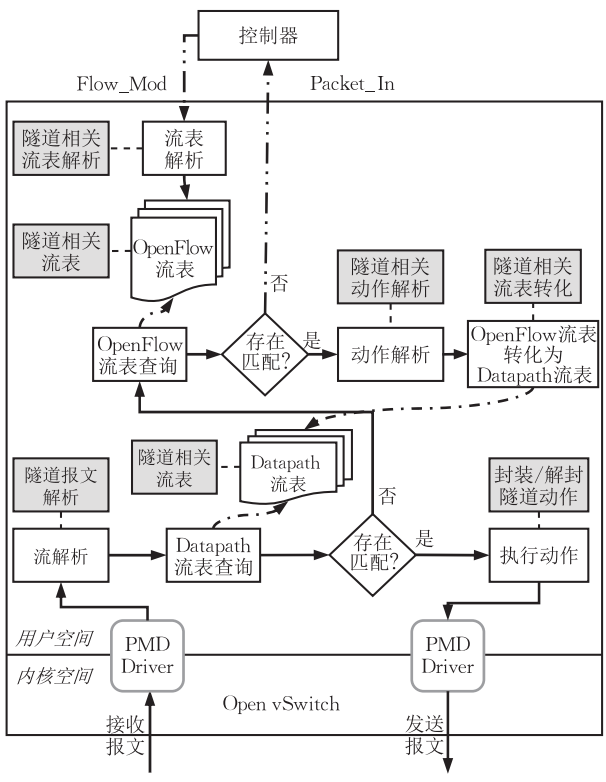


图8 使用DPDK加速MAT隧道

5　实验评估

在实验评估部分，首先我们使用 Mininet[①] 搭建了模拟测试环境，并先采用基于内核的 OVS（也即没有经过 DPDK 加速，下文中如无特殊说明，均为基于内核的 OVS）. Mininet 可以在单台机器上仿真较大规模的网络拓扑. 我们采用线性拓扑，使得两台模拟的主机之间分别经过 20、40、60、80 和 100 跳，采用 Ping 工具两个主机之间的时延，评估了 MAT 隧道封装和解封装动作在时延方面带来的开销，对比的基准是不进行隧道封装而直接做转发的情形. 其结果如图 9 所示. 可以发现在本实验环境下，经过 MAT 隧道封装的时延比无隧道大约高 0.1 ms 到 0.15 ms，随着跳数地增加，开销增加较为缓慢，说明 MAT 隧道具有较为良好的可扩展性.

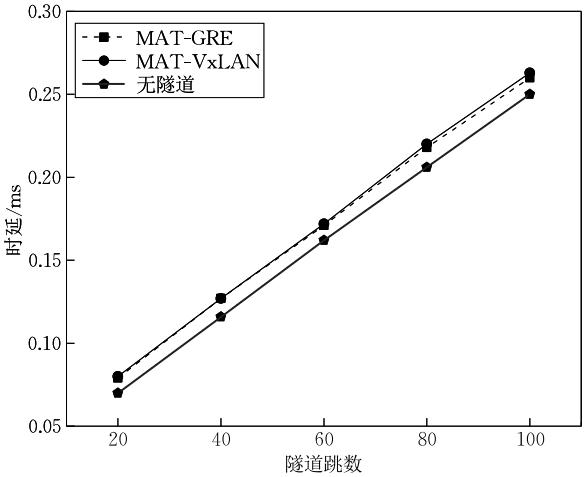


图 9　MAT 隧道封装和解封装的时延开销

Mininet 虽然能仿真大规模网络，但是它是在同一台机器上，真实性较差. 在接下来进一步实验中，我们采用了 Topology Zoo 数据集提供的中 Sprint 的真实拓扑[②]. 该拓扑由 11 个节点和 18 条链路构成. 如图 10 所示，我们使用 Vmware ESX 作为虚拟化平台搭建上述拓扑，将该拓扑中的每个节点映射为一台虚拟机，并将链路映射为虚拟机之间的连接关系. 而在每一台虚拟机中，安装带有 MAT 隧道功能的 OVS. 此外还需要两台虚拟机作为通信主机，并与 OVS 直接相连；一台虚拟机作为控制器，与各 OVS 通过网络连接，以实现对各个 OVS 的控制和管理.

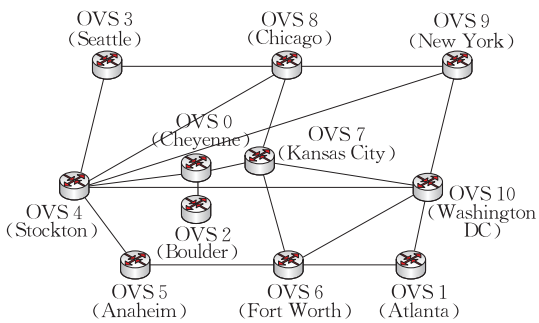


图 10　实验拓扑

我们首先将两台通信主机接入实验拓扑中，两台相连的 OVS 之间建立隧道以模拟节点之间 Overlay 的场景. 通过控制器向对应的 OVS 下发流表控制两台通信主机连续经过 1～5 跳隧道，并使用 Ping 工具测量每种情况下两台主机之间的通信延迟. 同时我们引入了两台 OVS 之间不建立隧道的情形作为对照. 图 11 展示了使用 MAT 模型的 GRE、VxLAN 隧道以及 OVS 标准实现的 GRE、VxLAN 隧道的平均时延（300 个包）对比. 更进一步，我们对于经过 5 跳隧道得到的时延数据做了统计，图 12 展示了统计结果.

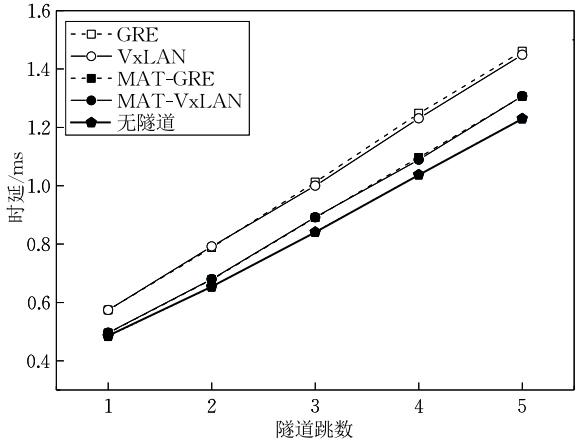


图 11　两台主机之间的平均时延

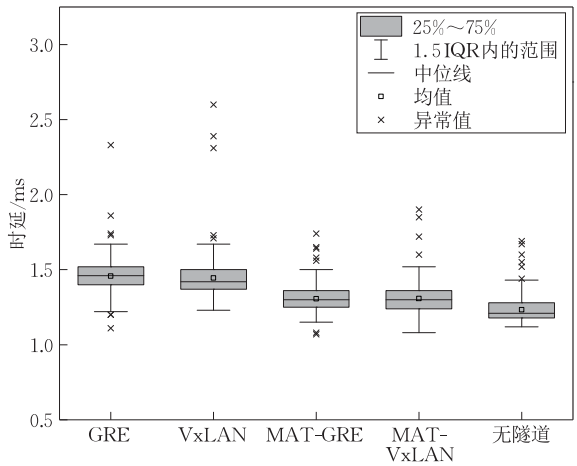


图 12　两台主机经过 5 跳隧道的平均时延分布

① Mininet，http://mininet.org/
② The Internet Topology Zoo. http://www.topology-zoo.org/dataset.html

隧道的时延测试表明，基于 MAT 模型的隧道由于简化了数据包在 OVS 内部的处理流程，相比于 OVS 实现的标准隧道，平均时延降低了 10%左右. 而时延分布表明，基于 MAT 模型的隧道在时延的分布上也更为集中.

我们还测量了在使用 DPDK 技术加速 MAT 隧道对其转发时延的影响. 如图 13 所示，可以看到在使用 DPDK 技术后，MAT 隧道可以进一步将其转发时延降低 20%左右.

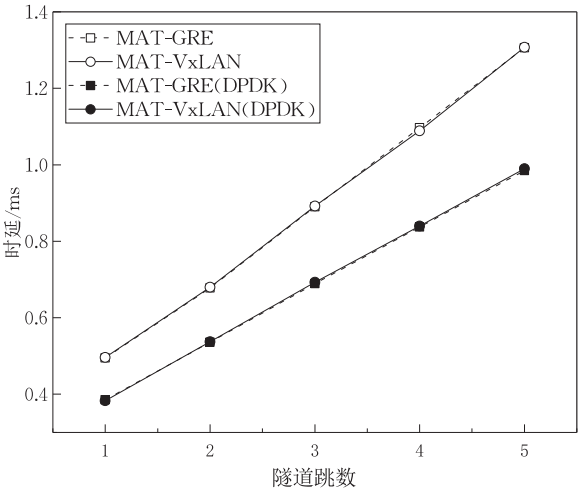


图 13 使用 DPDK 加速 MAT 隧道的平均时延

我们还进行了通过隧道进行路径切换的实验. 两台通信主机分别接在 OVS1 与 OVS3 上. 通过断开和恢复 OVS 之间的链路，引起路径切换. 而两个相连 OVS 之间需要建立隧道，因此路径的切换最终引起隧道的建立与拆除. 例如两台通信主机原有的通信路径为：HOST1-OVS1-OVS10-OVS4-OVS3-HOST2. 当断开 OVS10 与 OVS4 之间的链路后，路径需要切换为：HOST1-OVS1-OVS10-OVS9-OVS4-OVS3-HOST2. 在这个过程中，OVS10 与 OVS4 之间原有建立的隧道需要被拆除，而 OVS10 与 OVS9、OVS9 与 OVS4 之间的隧道需要被建立，而另一方面也需要通过控制器下发流表规则以新路径进行数据包转发. 对于 OVS 标准实现的隧道，通过 OVSDB 的方式下发隧道配置并额外下发用于转发的流表，而基于 MAT 模型的隧道则将隧道封装解封装的流表与用于转发的流表进行合并而一次性下发给相关的 OVS. 图 14 展示了实验过程中链路与通信主机之间路径的变化情况.

我们通过 iperf 工具（UDP 流，指定带宽为 800 Mbps）测试了两台通信主机在路径变化过程中吞吐量与抖动，测试结果取三次实验的平均值，如图 15、图 16 所示.

图 14 链路变化与通信主机之间的路径变化过程

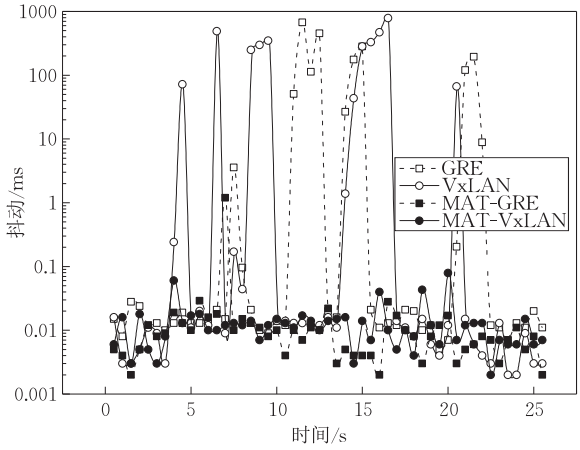


图 15 路径切换过程中两台主机间的通信抖动变化

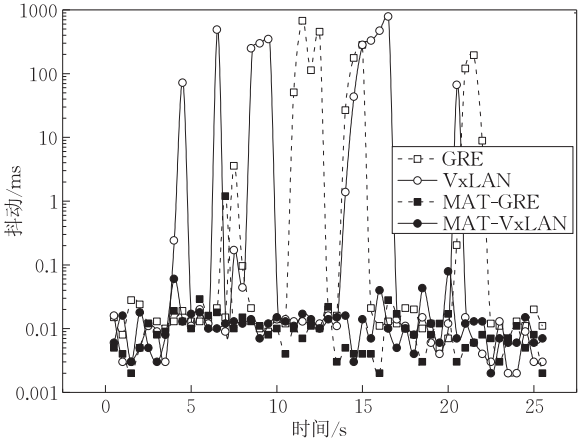


图 16 路径切换过程中两台主机间的吞吐量变化

路径切换测试表明，基于 MAT 的隧道模型相比于 OVS 实现的标准隧道，路径切换过程中的最大抖动最多可降低 3 个数量级，吞吐量损失也降低

了 2 倍. 这是因为 MAT 隧道只需要下发表项，不需要在 Linux 内核中生成用于隧道的网络设备，从而降低了建立和删除隧道过程中的开销. 因此在使用隧道进行流量调度的过程中，基于 MAT 模型的隧道具有高效灵活的特征.

我们测试了 MAT 隧道所在主机的 CPU 占用率情况，在两端通信主机使用 iperf 发包，发现通信主机的 CPU 会先于 OVS 所在主机达到满载状态，这种情况下，使用 MAT 隧道的主机与使用 OVS 所实现的隧道的主机 CPU 占用率没有明显差别.

我们还评估了 MAT 隧道所引入的额外开销. 在 MAT 隧道中，我们虽然简化了隧道报文的处理逻辑，但是这也让所有的数据端口都工作在二层，而不具有三层的能力. 为了实现三层互通，我们额外引入基于控制器的 ARP 代理，处理隧道端点相关的 ARP 消息. 因此当隧道两端的交换机跨越子网时，对于每台主机的的第一个报文，其需要打上两端交换机的虚拟 IP 地址，而且必须通过这个 ARP 代理响应网关的 ARP 请求从而完成三层互通. 因此我们将两个安装 OVS 的虚拟机放在不同的两个网段上，中间经过传统的 IP 网关，然后测试与两个 OVS 直接相连虚机的首包时延和平均时延，这样就能够评估 MAT 隧道带来的额外开销. 如图 17 所示，我们可以看到尽管 MAT 隧道的平均时延更小，但是其首包时延明显高于 OVS 实现的标准隧道，这是因为控制器需要分别处理隧道两端点的 ARP 消息，相对于 OVS 实现的标准隧道，MAT 隧道额外多出来两次 Packet_In 和 Packet_Out 的开销. 但这种开销只会影响一个流最开始的有限个数据包，因此总体而言，MAT 隧道在性能上更具有优势.

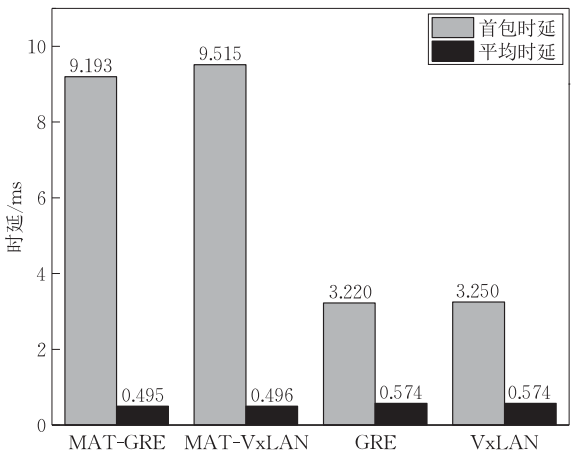


图 17　隧道的首包时延与平均时延

6　总　结

IP 隧道技术在网络中具有十分广泛的应用，而在 SDN 网络中，现有的 IP 隧道存在难以维护、配置复杂和灵活性不足的问题. 本文提出了一种基于匹配动作表的隧道模型，并引入了基于控制器的 ARP 代理解决了该隧道模型中主机和隧道交换机对于 ARP 报文处理的问题，同时也在该隧道模型中考虑了隔离性的支持问题. 我们基于开源软件交换机 Open vSwitch 和开源控制器 Floodlight 初步实现了基于匹配动作表的 GRE 和 VxLAN 隧道，并利用 DPDK 做了一定的性能优化. 主机时延的实验评估表明，基于匹配动作表的隧道可以将隧道转发的时延降低约 10%，通过 DPDK 加速后可以进一步降低约 20%，而路径切换的实验评估表明，基于匹配动作表的隧道可以将隧道切换过程中的最大抖动降低近 3 个数量级，同时对吞吐量的影响可以降低 50%.

参　考　文　献

[1] Chowdhury N M M K, Boutaba R. A survey of network virtualization. Computer Networks, 2010, 54(5): 862-876

[2] Kurian J, Sarac K. A survey on the design, applications, and enhancements of application-layer overlay networks. ACM Computing Surveys, 2010, 43(1): 5:1-5:34

[3] Lua E K, Crowcroft J, Pias M, et al. A survey and comparison of peer-to-peer overlay network schemes. IEEE Communications Surveys & Tutorials, 2005, 7(2): 72-93

[4] Ahlgren B, Dannewitz C, Imbrenda C, et al. A survey of information-centric networking. IEEE Communications Magazine, 2012, 50(7): 26-36

[5] Open Networking Foundation. Software-defined networking: The new norm for networks. ONF White Paper, 2012

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

[7] Pfaff B, Pettit J, Koponen T, et al. The design and implementation of open vSwitch//Proceedings of the USENIX Symposium on Networked Systems Design and Implementation (NSDI). Oakland, USA, 2015: 117-130

[8] Pfaff B, Davie B. The Open vSwitch database management protocol, IETF RFC 7047, December 2013

[9] Mahalingam M, Dutt D, Duda K, et al. Virtual extensible local area network (VXLAN): A framework for overlaying virtualized layer 2 networks over layer 3 networks, IETF RFC 7348, August 2014

[10] Hanks S, Meyer D, Farinacci D, et al. Generic routing encapsulation (GRE). IETF RFC 2784, March 2000

[11] Bari M F, Boutaba R, Esteves R, et al. Data center network virtualization: A survey. IEEE Communications Surveys & Tutorials, 2013, 15(2): 909-928

[12] Garg P, Wang Y S. NVGRE: Network virtualization using generic routing encapsulation, IETF RFC 7637, September 2015
[13] Gross J, Ganga I, Sridhar T, et al. Geneve: Generic network virtualization encapsulation, IETF Draft, March 2018
[14] Maino F, Kreeger L. Elzur U, et al., Generic protocol extension for VXLAN, IETF draft, April 2018
[15] Touch J, Townsley M. IP tunnels in the Internet architecture. IETF draft, January 2018
[16] Peter S, Javed U, Zhang Q, et al. One tunnel is (often) enough//Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM). Chicago, USA, 2014: 99-110
[17] Farinacci D, Lewis D, Meyer D, et al. The locator/ID separation protocol (LISP), IETF RFC 6830, January 2013
[18] Jain S, Kumar A, Mandal S, et al. B4: Experience with a globally-deployed software defined WAN//Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM). Hong Kong, China, 2013: 3-14
[19] Rodriguez-Natal A, Portoles-Comeras M, Ermagan V, et al. LISP: A southbound SDN protocol? IEEE Communications Magazine, 2015, 53(7): 201-207
[20] Heinonen J, Partti T, Kallio M, et al. Dynamic tunnel switching for SDN-based cellular core networks//Proceedings of the 4th Workshop on All Things Cellular: Operations, Applications, & Challenges. Chicago, USA, 2014: 27-32
[21] Firestone D. VFP: A virtual switch platform for host SDN in the public cloud//Proceedings of the USENIX Symposium on Networked Systems Design and Implementation (NSDI). Boston, USA, 2017: 315-328

ZHANG Ke-Yao, born in 1993, M. S. candidate. His research interests include inter-domain routing in SDN and programmable data plane.

BI Jun, born in 1972, Ph. D., Changjiang scholar distinguished professor, Ph. D. supervisor. His research interests include new network architecture (SDN, NFV and network security architecture).

WANG Yang-Yang, born in 1979, Ph. D., postdoctoral researcher. His research interests include Internet architecture, SDN, and Internet measurement.

Background

An IP Tunnel is a communication channel which can be created by using encapsulation technologies. Tunneling over IP has been widely used in various networking environments. For examples, it can be used in network virtualization for resource and user isolation. In addition, bypassing native Internet routing path via tunneling among overlay nodes can effectively improve end-to-end communication performance. Tunneling is also used for connecting disjoint network innovations (e. g., ICN, IPv6).

In recent years, Software Defined Networking has been deployed increasingly. SDN provides open and unified APIs, which greatly simplifies and enhances the network management efficiency. However, as a significant southbound interface, OpenFlow does not primitively support the establishment of IP tunnels, while only supporting tag-based tunnels, such as MPLS, which makes OpenFlow limited in network applications.

Nowadays, IP tunnels are supported by other mechanisms on OpenFlow Switches. For instance, IP tunnels can be created and maintained as tunnel ports via OVSDB in Open vSwitch. Nonetheless, there exists maintenance difficulty, management complexity, low efficiency in management and configuration.

To make IP tunnels easier in SDN, this paper adopts Match-Action Table programming model of data plane and proposes a new IP tunnel mechanism, called the MAT tunnel. The MAT tunnel can encapsulate and decapsulate directly by installing flow rules instead of manually configuring tunnel ports.

This paper also implements the MAT tunnel prototype based on Open vSwitch and Floodlight. We construct a simulation environment based on a real topology. Comparing traditional tunnels, we find that the MAT tunnel can reduce the average delay by 10 percent. In addition, the tunnel path switching tests suggest that the MAT tunnel can significantly decrease the delay jitter and throughput loss.

This work is supported by the National Key Research and Development Program of China "Cyberspace Security" Project (2017YFB0801701), and the National Natural Science Foundation of China (No. 61472213). This work is an important part of inter-domain SDN and cyberspace security architecture.

Our group has been working on SDN since 2012. The previous work includes SDN architecture, method design, method implementation, deployment and comparative evaluation. Some papers have been published or accepted by SIGCOMM, INFOCOM, ICNP, IEEE Communications Magazine, IEEE Network and other international conferences and journals.