doi:10.11887/j.cn.202405021 http://journal.nudt.edu.cn

网络切片可编程数据平面模型

刘忠沛,吕高锋*,王继昌,杨翔瑞
(国防科技大学 计算机学院,湖南 长沙 410073)

摘 要:可重构匹配表(reconfigurable match table, RMT)是一种可编程的数据包处理流水线架构。为了实现可编程数据平面对更多不同网络协议的支持,在该架构的基础上扩展逆解析器,使用扩展后的逆解析器以及两个RMT流水线组成一个协议无关的网络切片可编程数据平面模型。由于RMT架构中采用精简指令集,扩展后的逆解析器采用复杂指令集,因此称扩展后的架构为混合指令RMT(hybrid-instruction RMT, HiRMT)。HiRMT能够支持基于IPv6转发平面的段路由、SID(segment ID)的多语义、微分段技术、多协议标签交换技术,以及虚拟扩展局域网技术。该架构具有广阔的应用场景。在Corundum原型平台上进行了逆解析器模块的性能测试,结果表明扩展后的逆解析器能够使用较少的资源在数据包大小达到512 B时以100 Gbit/s的吞吐量进行处理。

关键词:可重构匹配表;混合指令;协议无关;网络切片
中图分类号:TP393 **文献标志码**:A **文章编号**:1001-2486(2024)05-200-09

论文拓展

Network slice programmable data plane model

LIU Zhongpei, LYU Gaofeng, WANG Jichang, YANG Xiangrui*
(College of Computer Science and Technology, National University of Defense Technology, Changsha 410073, China)

Abstract: RMT (reconfigurable match table) is a programmable pipeline architecture for packet processing. In order to enable the programmable data plane to support more different network protocols, the deparser based on RMT was extended. A protocol-independent network slicing programmable data plane model was formed by using the extended deparser and two RMT pipelines. Since reduced instruction set was used in RMT architecture and complex instruction set was used in the extended deparser, the extended architecture was called HiRMT(hybrid-instruction RMT). HiRMT can support segment routing IPv6, multiple semantics for SID(segment ID), micro SID, multi-protocol label switching and virtual extensible local area network. This architecture has broad application prospects. The performance of the deparser module was tested on the Corundum prototype platform, and the results show that the extended deparser can process the packet size up to 512 B with a throughput of 100 Gbit/s with fewer resources.

Keywords: reconfigurable match table; hybrid instruction; protocol-independent; network slice

网络技术的发展往往是服务于实际需求的。近几年来,网络的架构发生了深刻的变化,尤其是在云计算的模型中,要求云中的网络资源能够得到足够的抽象和灵活的调度,即虚拟化的能力。随着网络虚拟化与网络功能的发展,对可编程数据平面的要求也不断提高,开发者需要能够根据自身需求对网络的转发过程进行重编程。

网络虚拟化是指虚拟网络节点之间的连接并不使用物理线缆连接,而是依靠特定的虚拟化链路相连。其主要是对网络资源(端口、宽带、IP地址)进行抽象,支持按照租户和应用进行动态分配和管理。作为虚拟化技术的分支,网络虚拟化本质上还是一种资源共享技术[1]。

网络分片是一种按需组网的方式,可以让运营商在统一的基础设施上分离出多个虚拟的端到端网络,每个网络切片从无线接入网到承载网再到核心网上进行逻辑隔离,以适配各种各样类型的应用[2]。

收稿日期:2022-05-26
基金项目:国家重点研发计划资助项目(2020YFB1805603)
第一作者:刘忠沛(1998—),男,河南平顶山人,硕士研究生,E-mail:747541120@qq.com
***通信作者**:吕高锋(1980—),男,陕西宝鸡人,副教授,博士,硕士生导师,E-mail:gflv@163.com
引用格式:刘忠沛,吕高锋,王继昌,等. 网络切片可编程数据平面模型[J]. 国防科技大学学报, 2024, 46(5): 200-208.
Citation: LIU Z P, LYU G F, WANG J C, et al. Network slice programmable data plane model[J]. Journal of National University of Defense Technology, 2024, 46(5): 200-208.

段路由(segment routing, SR)是基于源路由理念而设计的在网络上转发数据包的一种协议。SR 将网络路径分成一个个段,并且为这些段和网络节点分配 SID(segment ID)。通过对 SID 进行有序排列,就可以得到一条转发路径[3]。SR 是替代多协议标签交换(multiple protocol label switching, MPLS)的隧道技术,它的应用场景与 MPLS 类似。一些常见的使用 MPLS 隧道的业务都可以平滑地切换到 SR 隧道。

无论是对可编程交换机[4]还是对智能网卡[5]来说,数据平面的可编程性都是重要的发展趋势[6]。针对交换芯片和 OpenFlow[7]协议中的两个限制——只允许在一组固定的字段上进行匹配 - 动作处理、OpenFlow 规范仅定义了有限的包处理动作集,The McKeown Group 提出了可重构匹配表(reconfigurable match tables, RMT)模型[8]。这是一种新的受精简指令集计算机(reduced instruction set computing, RISC)启发的流水线架构,用于交换芯片分组转发,确定了基本的最小动作原语集,以指定报头在硬件中如何处理。RMT 允许在不修改硬件的情况下通过修改匹配字段和算术逻辑单元(arithmetic logic unit, ALU)程序以更改转发平面。与在 OpenFlow 中一样,设计者可以在资源允许的范围内指定任意宽度和深度的多个匹配表,每个表可经过配置后具备在任意字段上进行匹配的能力。然而,RMT 相比于 OpenFlow 允许更全面地修改所有报头字段。在 OpenFlow Switch 通用转发模型的基础上,研究人员尝试建立更加通用的网络数据处理抽象模型。华为美国研究所的宋浩宇[9]提出了协议无关转发(protocol oblivious forwarding, POF)框架,通过对网络转发处理行为进行再次抽象,实现协议无关转发处理。在 POF 架构中,POF 交换机并没有协议的概念,它仅在 POF 控制器的指导下通过{offset, length}来定位待匹配的数据,从而完成网络数据转发处理。随着网络可编程性的成熟,单个设备将不得不同时支持多个独立开发的模块。数据平面可编程性需要能够在单个设备中运行多个包处理模块,这些可能是由不同独立的团队开发的。因此,需要隔离机制来确保同一设备上的模块不会相互干扰。Menshen 通过在 RMT 的基础上添加轻量级的隔离机制实现了同一设备上对不同模块处理流程的隔离[10],并允许加载新模块而不影响已经运行的模块,保证了可编程数据平面资源的安全性。dRMT[11]将计算资源(例如,匹配/动作处理器)从内存中分离出来,它们通过一个 Crossbar 互连。每个匹配动作(match-action, MA)处理器都持有一个 P4 程序,并以运行至终结(run-to-completion, RTC)的方式处理数据包。FlexCore[12]体系结构在 dRMT 部分分解设计中引入了另一个创新点,在 MA 处理器的计算本地内存持有一个称为程序描述表的间接数据结构。该表包含关于程序控制流的元数据,可用于重新配置该处理器的处理流程。本文中扩展后的 Deparser 借鉴了 dRMT 中以 RTC 方式处理数据包的做法,并在 MA 处理器中引入过程描述表(processing description table, PDT)来控制数据包的处理流程。

当前 RMT 架构中的 Deparser 模块仅负责执行简单的合并操作。它将包头向量(packet header vector, PHV)写入包头中相应的字节偏移位置,将包头与包缓存中相应的有效负载合并,并将合并后的包从流水线中传输出去。RMT 在每个阶段进行匹配动作,通过超长指令字(very long instruction word, VLIW)简单地对数据包头进行解析与改写,无法支持更为复杂的网络协议处理。本文通过扩展 RMT 中的 Deparser 模块来扩展 RMT 架构的功能,使 Deparser 模块不再是简单合并 PHV 与报文,而是通过解析容器生成的 PHV 完成更加复杂的报文处理,如插入报文头、封装报文头、压缩报文头等有状态的操作,使其能够更好地支持不同协议的处理与转发,实现协议无关的流水线处理架构。

本文介绍了混合指令 RMT(hybrid-instruction RMT, HiRMT)的具体设计思路以及架构,最后分析了 HiRMT 在各个应用场景下的性能,实验表明扩展后的逆解析器能够使用较少的资源在数据包大小达到 512 B 时以 100 Gbit/s 的吞吐量进行处理。

1　协议无关的流水线设计

CacheP4[13]的思想来源于转发设备在数据平面转发报文时,对同一无状态报文流中的报文进行重复的表项匹配。CacheP4 通过避免这种重复匹配,加快了报文转发的处理速度。受 CacheP4 的启发,我们认为不需要在 RMT 的起始位置放置匹配动作表(match action table, MAT)缓存,而是应该将这条快速路径视作多级流水,即每个处理阶段视作一级缓存,并在末端补充一条慢速路径用来进行复杂的有状态的报文处理,使该慢速路径成为 RMT 架构的 Cache ALU。参照 CacheP4,在 RMT 架构的基础上扩展 Deparser 为 Deparser

plus,其中 RMT 架构主要负责简单无状态的处理,即采用精简指令集对字段进行修改,是传统专用网络处理器(network processor, NP)转发模型中的流水线(Pipeline)架构;Deparser plus 主要负责复杂且有状态的操作,如报文头的封装与协议头的插入等,采用复杂指令集,是 NP 转发模型中的 RTC 架构。RTC 架构与 Pipeline 架构的对比如表 1 所示。通过 RTC 与 Pipeline 架构的结合实现混合指令的协议无关报文处理架构设计。

表 1 RTC 对比 Pipeline

Tab. 1 RTC versus Pipeline

对比指标	Pipeline	RTC
运行方式	将一个功能(大于模块级的功能)分解成多个独立阶段,不同阶段间通过队列传递数据	一个程序中一般会分为几个不同的逻辑功能,但这几个逻辑功能会在一个 CPU 核上运行
开发情况	当芯片设计好,它支持的处理能力就固定了	后期可以通过软件代码的方式加入新的功能
性能	效率高,多核之间会有缓存一致性问题	效率低,会有处理流程上的依赖问题
扩展性	不好	较好

1.1 HiRMT 总体架构

如图 1 所示,HiRMT 由 Ingress RMT、Deparser plus 与 Egress RMT 三部分组成。Ingress RMT 中原有的 Deparser 模块被 Deparser plus 替代,所以 Ingress RMT 的最后阶段结束后不对报文数据与报文头进行合并,而是将解析结果(metadata 与 PHV)与缓存中的报文直接发送给 Deparser plus 模块。

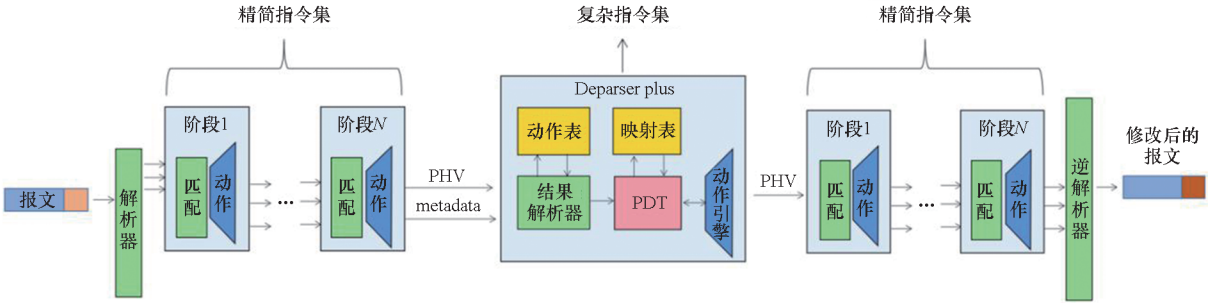


图 1 HiRMT 总体架构

Fig. 1 HiRMT overall architecture

Ingress RMT 模块采用精简指令集,主要负责接收报文并判断报文类型,完成对报文简单的预处理。将处理报文所需的解析结果写入 metadata 中,其中包含报文类型以及流 ID,之后交由 Deparser plus 模块进行处理。

Deparser plus 模块主要负责接收由 Ingress RMT 发送的 PHV 与 metadata,根据 metadata 中的流 ID 与报文类型对数据包进行相应的有状态的处理,并根据动作表中的复杂指令对 PHV 任意字段进行修改,然后将修改后的 PHV 传送给 Egress RMT 模块进行最后的修改与转发。

Egress RMT 模块采用精简指令集,主要负责接收由 Deparser plus 模块发送的 PHV,进入 RMT 管道进行目的 IP 地址的替换,查找转发表与邻接表生成第二层头部并完成转发。

Ingress RMT、Deparser plus 与 Egress RMT 三部分均为可编程模块,共同构成可编程混合指令流水线 HiRMT。编程人员可根据需要改变 HiRMT 的功能,符合可编程数据平面的要求,实现协议无关的流水线处理架构。

1.2 可编程的 MAT

MAT 是 RMT 架构中的核心模块。每个阶段都在 MAT 中通过精确匹配查找由键提取器构造的固定大小的键,这是在匹配表中实际查找的条目。查找结果用作 VLIW 操作表的索引,以标识要执行的相应操作。

每个 VLIW 操作表项表明 PHV 中的哪些字段用作 ALU 操作数(每个 ALU 的操作数 Crossbar 的配置),以及由 VLIW 指令控制的每个 ALU 应该使用哪些操作码(加减等)。每个 ALU 根据其操作数和操作码输出一个值。每个 PHV 容器有一个 ALU,不需要在输出上使用 Crossbar,因为每个 ALU 的输出直接连接到相应的 PHV 容器。当一级 ALU 修改其 PHV 后,将修改后的 PHV 传递到下一级。

1.3 协议无关的可编程的 Deparser plus

Deparser plus 的功能是接收由 Ingress RMT

发送的 metadata 与 PHV,metadata 中携带有报文类型与流 ID,可据此判断需要对报文进行的操作。通过分析 metadata 查找动作表得到操作指令,并对报文的任意位置进行相应的处理,例如进行段路由头部(segment routing header, SRH)的插入或 VXLAN 报文的封装,然后将修改后的报文与解析结果传送给 Egress RMT 模块进行最后的修改与转发。Deparser plus 主要负责复杂且有状态的操作,如报文头的压缩、封装与协议头的插入等,采用复杂指令集,是 NP 转发模型中的 RTC 架构。

将 PDT 放入 Deparser plus 的 PDT 模块中,以此来控制 Action 模块对数据包进行 RTC 处理。该模块可由结果解析器模块配置,也可由开发人员通过 P4 程序编译而来。每个 MA 处理器维护一个本地 PDT,所有到达端口的数据包将首先命中 PDT 中的一个默认条目,以激活数据包处理。每个条目存储一个程序元素的指令,用来表示一个具体操作,并包含指向下一指令的指针。指令包含操作码以及其他处理数据所需要的信息,例如指向该指令实现的资源指针。指针地址可以是静态随机存储器(static random access memory, SRAM)位置或三态内容寻址存储器(ternary content addressable memory, TCAM)位置,因为对于一个 PDT 条目,只有一种指针类型是有效的。以 SRH 的插入操作为例,PDT 的表项与控制流程如图 2 所示。

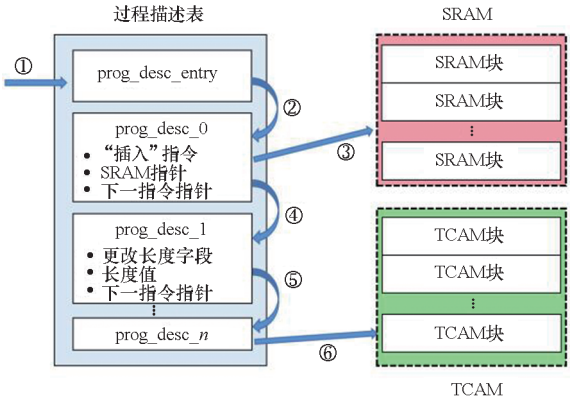


图 2　PDT 处理流程实例

Fig. 2　PDT process example

Deparser plus 由结果解析器、PDT 与 Action 模块组成,其中结果解析器模块与动作表相连接,PDT 模块与映射表相连,Action 模块与 PDT 模块相连接。两个先入先出(first in first out, FIFO)队列分别向结果解析器模块发送 metadata 和 PHV。结果解析器模块根据 metadata 确定数据包类型,查找动作表得到动作指令。指令内容包括操作码和其他信息的附加字段。动作表中包含每一条指令对应的一系列具体操作,如插入数据、修改报文长度等。结果解析器以这种方式来解析指令并配置 PDT 模块。之后由 PDT 模块按照指令设置程序对 Action 模块进行调用以处理 PHV。映射表中存有 PDT 处理报文头时所需要的数据,如 SRH。Action 模块由 PDT 模块控制,以获得偏移量、偏移长度和要插入或修改的信息。偏移量表示要操作的报文的位置。最后,修改后的 PHV 被发送到 Egress RMT。

对于一些基本的简单指令,可以在 RMT 流水线中完成,如加、减、替换指令等,因此只需在 Deparser plus 中定义复杂操作的指令。目前共定义了 7 种指令:向报文头中插入数据、删除报文头数据、封装报文头、解封装报文头、向动作表中插入指令、报文头的压缩以及解压缩。这 7 种指令的定义及格式如表 2 所示。

表 2　Deparser plus 指令定义

Tab. 2　Deparser plus instruction definition

指令	描述	格式
insert	在报文头的相应位置插入一段数据	{opcode, offset, mapping table index, inserted data length, packet length field offset}
delete	在报文头的相应位置删除一段数据	{opcode, offset, deleted data length, packet length field offset}
compress	对报文头的相应字段进行压缩	{opcode, SID prefix, SID length, packet length field offset}
decompress	对报文头的相应字段进行解压缩	{opcode, uSID prefix, uSID length, packet length field offset}
encapsulate	封装报文头	{opcode, encapsulated info, offset of altered field, altered field info, packet length field offset}
decapsulate	解封装报文头	{opcode, offset of encapsulated info, offset of altered field, altered field, packet length field offset}
alter	修改动作表	{opcode, action table index, instruction length, altered instruction}

1.4 面向网络切片的可编程 SR 转发

本文提出的流水线具有广阔的应用场景，例如可应用于 SR、VXLAN、基于用户数据报协议的段路由（segment routing over UDP，SRoU）、带内网络遥测（in-band network telemetry，INT）以及服务函数链（service function chain，SFC）编排等技术中。其中对 SR 技术的支持包含 MPLS SR[14]、SRv6、SID 的多语义与 Function 三方面。

MPLS SR 和 SRv6 是两种 SR 技术的主要应用形式。相比于 MPLS SR，SRv6 有着更加强大的网络可编程空间，这归功于其灵活简易的 SRH 扩展头，其中的关键之一是 128 bit 的 SRv6 SID 的使用。SRv6 的每个 Segment 可以灵活分为多段，每段的长度也可以变化，由此为编程提供更大空间。SID 中的 Function 字段使 SID 具有多语义特点，该字段占用较少比特位用于定义所属设备收到该标签后的操作。

1.4.1 SRv6

SRv6 简单来讲即 SR + IPv6，是新一代 IP 承载协议。其采用现有的 IPv6 转发技术，通过灵活的 IPv6 扩展头，实现网络可编程。为基于 IPv6 转发平面实现 SR 技术，在 IPv6 路由扩展头新增 SRH 扩展头，该扩展头指定一个 IPv6 的显式路径，存储 IPv6 的 Segment List 信息。Segment List 即为对段和网络节点进行有序排列得到的一条转发路径。报文转发时，依靠 Segments Left（指示当前活跃的 Segment List）和 Segment List 字段共同决定 IPv6 目的地址信息，从而指导报文的转发路径和行为。

数据包的 SRH 由 SRv6 传输域的入口节点产生，这类节点被称为源节点。当产生新的 SRv6 数据包或收到需要进行 SRv6 转发的一般 IPv6 数据包时，源节点会为数据包创建 SRH，并将这个 SRH 插入数据包的报文头中。

在 HiRMT 中，通过扩展 RMT 流水线中的 Deparser 模块，使其成为 Deparser plus 以支持 SRv6 的转发。如图 3 所示，以源节点为例展示了 SRv6 报文转发流程在该流水线上的映射。

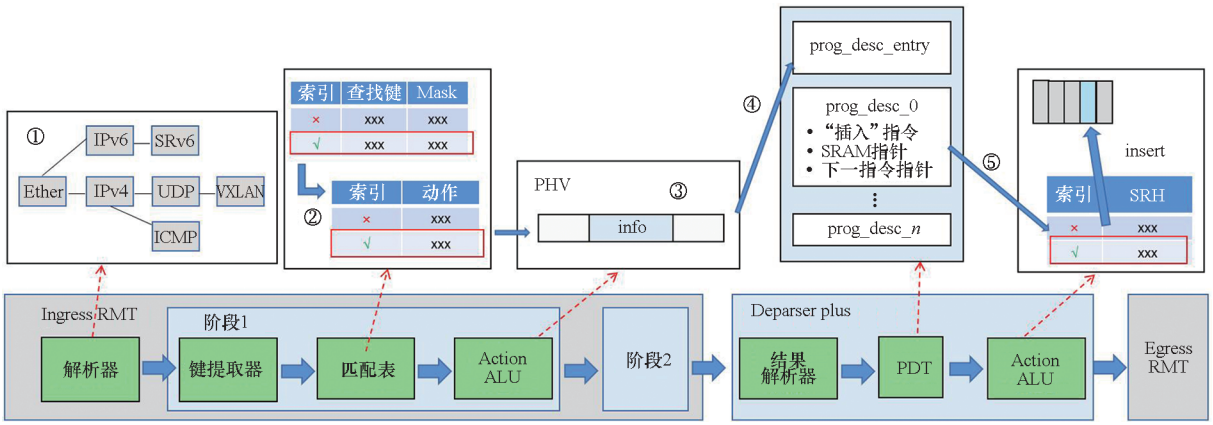


图 3 源节点的 SRv6 报文转发流程映射

Fig. 3 SRv6 packet forwarding process mapping of source nodes

Ingress RMT 负责解析报文，提取其流 ID 并写入 metadata。若该数据包为需要插入 SRH 的数据包，则将 metadata 发送给 Deparser plus 模块。

在 Deparser plus 模块中，根据 metadata 判断报文类型，若为源节点 SRv6 报文，则根据 metadata 中的流 ID 和包类型查找动作表，得到 insert 指令并配置 PDT，指令中包含 insert 操作码、偏移量、映射表索引、插入数据长度、报文长度字段偏移量。PDT 模块通过映射表索引得到 SRH，通过 insert 指令中的偏移量和偏移长度调用 Action 模块将 SRH 插入 IPv6 报文中，并将 IPv6 报文头中的 PayLoad Length 字段增加对应的长度值。最后将解析结果与报文发送给 Egress RMT 进行常规的 SRv6 报文处理，如替换目的 IP 地址、改写 Segment Left 字段、替换介质访问控制（media access control，MAC）地址等，并查找转发表得到输出端口将报文发送出去。使用 P4 程序描述 HiRMT 对 SRv6 报文的具体处理行为如算法 1 所示。该算法的复杂度仅与转发表的大小以及邻居表的大小有关，因此该算法的复杂度为 $O(n)$。

1.4.2 SID 的多语义与 Function

SRv6 的可编程能力来源于三部分：①可以将多个 Segment 组合起来，形成 SRv6 路径，即路径可编辑。②SRv6 Segment 定义了 SRv6 网络编程中的网络指令，指示要去哪，怎么去。标识 SRv6 Segment 的 ID 被称为 SRv6 SID。SRv6 SID 是一个 128 bit 的值，为 IPv6 地址形式，由 Locator、Function 和 Arguments 三部分组成。SRv6 的每个

算法 1　P4 程序描述 SRv6 报文处理流程实例

Alg. 1　P4 program description example of the SRv6 packet processing flow

输入:接收报文 packet
输出:修改后的报文 modified_packet

```
if (hop_limit = =0) drop (packet);
else if(SRH){
    if (Des_IP = Local_IP){
      if (Segment_Left = =0){
        Des_IP = Segment_List[0];
        Delete(SRH);
        hop_limit - -;
        Submit(modified_packet);
      }
    else if{
      Des_IP = Segment_List[Segment_Left];
      Segment_Left - -;
      hop_limit - -;
      out_port = Look_up(FIB,Des_IP);
      Des_MAC = Look_up(ADJ,Des_IP);
      Forward(out_port,modified_packet);
    }
    else{
      hop_limit - -;
      out_port = Look_up(FIB,Des_IP);
      Des_MAC = Look_up(ADJ,Des_IP);
      Forward(out_port,modified_packet);
    }
}
else{
    flow_id = Look_up(Flow_table);
    if(flow_id = =IPv6){
      out_port = Look_up(FIB,Des_IP);
      Des_MAC = Look_up(ADJ,Des_IP);
      Forward(out_port,modified_packet);
    }
    else if(flow_id = =source_SRv6){
      instruction = Look_up(Action_table,flow_id)
      Config(PDT,instruction);
      SRH = Look_up(Mapping_table,instruction[index]);
      Insert(PHV,SRH,offset,SRH_length);
      payload_length = payload_length + SRH_length;
      Des_IP = Segment_List[Segment_Left];
      Segment_Left - -;
      out_port = Look_up(FIB,Des_IP);
      Des_MAC = Look_up(ADJ,Des_IP);
      Forward(out_port,modified_packet);
    }
}
```

Segment 是 128 bit,可以灵活分为多段,每段功能和长度可以自定义,由此具备灵活编程能力,即业务可编辑。③Segment 序列之后的 Optional TLV(type-length-value)。报文在网络中传送时,需要在转发面封装一些非规则的信息,可以通过 SRH 中 TLV 的灵活组合来完成,即应用可编辑。

在本文的流水线架构中,可以通过在 Deparser plus 的动作表中添加 SID 表项,即使用 SID 的 Function 字段查找所需要执行的指令。将指令传递给 PDT 模块后由 PDT 模块控制 Action 模块执行 Function 所表示的指令,完成对数据包的复杂处理。

1.4.3　对微分段的支持

SRv6 引入的协议开销远大于 MPLS SR,Segment 所对应的操作也比 MPLS SR 复杂,因此 SRv6 对网络设备提出了非常高的要求。SRv6 在网络可编程性和负载均衡方面有着巨大的优势,但要发挥其优势,迫切需要解决 SRv6 在协议开销、承载效率、最大传输单元(maximum transmission unit, MTU)和对硬件要求方面的问题。这几个问题,其实本质上都是同一个问题:如何提高 SRv6 Segment 效率? 为了解决这个问题,微分段(micro SID, uSID[15])与 G-SID[16](general SID)头压缩方案被提出。思科系统公司对现有 SRv6 框架做了扩展,定义了新的 Segment 类型 uSID。uSID 将极大加速 SRv6 在网络侧的部署,并成为 SRv6 新范式。

在一个正常的 SRv6 包头(SRH),每一个 SID 都需要保存相同的公共前缀,但这些相同的部分实在多余。因此,可把 Segment List 中相同的公共前缀(网段)提取作为压缩包头的前缀块,将 Node ID 和 Function ID 作为 uSID 与公共前缀一起形成压缩后的 SID 列表。可见一个 128 bit 的 IPv6 地址被分为 8 份,第 1 份(16 bit)用于表示 uSID 块信息,另外 7 份每份用于表示一个 Segment 信息,这意味着 SRv6 Segment 效率提高了 7 倍。当 uSID 小于 7 跳,uSID 甚至不需要额外的 SRH 包头,只需要将 IPv6 目的地址重新规划和定义,即可实现 Segment Routing 的功能。这非常有利于主机和网络设备的硬件处理,大幅度地降低了 SRv6 实现难度、缩短了数据包的长度。SRv6 数据包完全没有产生额外的包头开销,带宽完全被节约下来了。

HiRMT 可以很好地支持 uSID 头压缩方案,这是一种有状态的处理。当 SRv6 报文进入流水线时,只需将报文类型与流 ID 写入 metadata,结果解析器通过分析 metadata 查找动作表,得到需

要执行的压缩动作指令，内容包含压缩操作码、SID 前缀、SID 长度、报文长度字段偏移量等，以此配置 PDT 模块，如图 4 所示为 PDT 中压缩 uSID 头部的程序。PDT 模块将 compress 指令与 SID 前缀发送给 Action 模块进行 SRH 的压缩。Action 模块将 uSID 前缀放在 128 bit SID 的前 16 bit，而后顺序地将拥有相同前缀的 SID 的后 16 bit 压缩进第一个 uSID 前缀之后，当 7 个 SID 压缩完毕后再创建一个 128 bit SID 进行接下来的压缩，直到遇到不同的前缀或 SID 压缩完毕为止。最后计算压缩后的报文头长度与压缩前报文长度的差值，将差值作为偏移量指导报文的重新分片并转发。

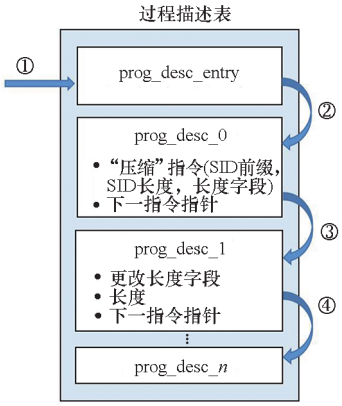


图 4 PDT 中压缩 uSID 时的程序

Fig. 4 Compression uSID program in PDT

2 HiRMT 应用与验证

本文基于 VCU118 FPGA 开发板实现了 HiRMT 原型系统，分析了 HiRMT 的应用场景，并进行实验验证。

2.1 原型系统

本文的原型系统是使用基于 VCU118 FPGA 开发板的 Corundum[17] 原型平台实现的。Corundum 是一个开源的、基于 FPGA 的 NIC 原型平台，用于 100 Gbit/s 和更高速率的网络接口开发。测试平台的搭建如图 5 所示，在原型系统中，Corundum 通过高速串行计算机扩展总线（peripheral component interconnect express，PCIe）连接到一台配备 Intel Xeon E5645 CPU 的机器，时钟频率为 2.40 GHz，并通过四通道小型可插拔光模块（QSFP28）连接到 Spirent FX3 – 100GO – T2 测试仪。HiRMT 由 2 个 RMT 流水线和 1 个 Deparser plus 模块组合而成。

2.2 原型系统应用场景

HiRMT 可支持更多丰富的应用场景，例如

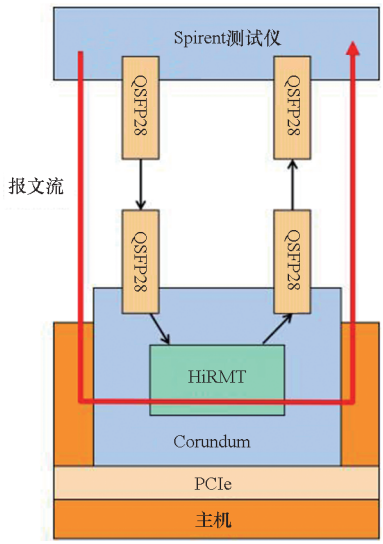


图 5 原型系统测试环境

Fig. 5 Prototype system test environment

VXLAN、SRoU、INT 与 SFC。如表 3 所示为这 4 种应用场景的对比与分析。

VXLAN 将虚拟网络中的数据帧封装在实际物理网络中的报文中进行传输，主要用于解决虚拟内存系统的可移植性限制以及云数据中心之间虚拟机的迁移[18]。可通过 HiRMT 中的 encapsulate 指令对 VXLAN 报文头进行封装。

SRoU 本质上是把 SRH 插入用户数据报协议（user datagram protocol，UDP）载荷中，在 HiRMT 中的处理方式类似于 SRv6 报文，可通过改变 SRH 的插入位置实现 SRoU 报文的生成与处理。

INT 是一种不需要网络控制平面干预、网络数据平面收集和报告网络状态的框架，工作在 L7 层[19]。可通过 HiRMT 中的 insert 指令插入其头部信息与遥测信息，实现 INT 报文的生成与处理。

SFC 引导网络流量按照业务逻辑所要求的既定的顺序经过这些业务点[20]。SR 转发技术的处理思路同样可以用在 SFC 中[21]。可通过调用 insert 指令为不同网络流量添加 SFC Header，也可调用 delete 指令将 SFC Header 从报文中删去，并提交给主机进行网络功能的处理。

2.3 性能分析

使用不同大小的 VXLAN 报文、SRv6 报文以及 SFC 报文对流水线架构进行测试。

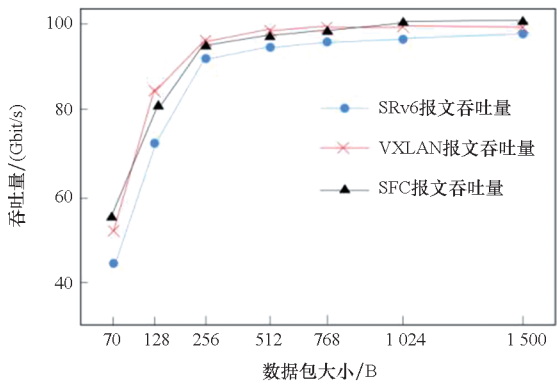
对于 SRv6 报文，Deparser plus 负责查找映射表来得到 SRH，因此相比于 VXLAN 报文的处理具有更多延迟，具体延迟取决于映射表中的 SRH 长度。

表 3　应用场景对比与分析

Tab. 3　Comparison and analysis of application scenarios

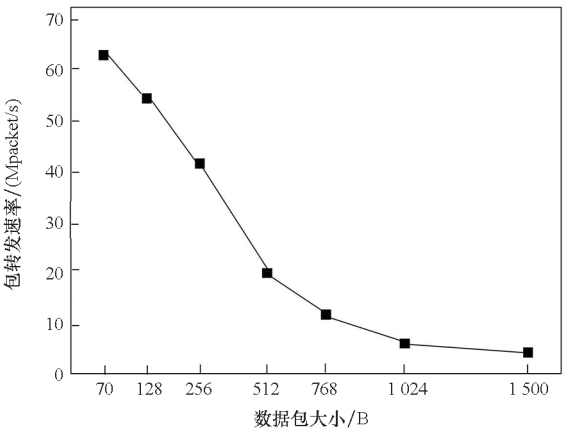
对比内容	VXLAN	SRoU	INT	SFC
原理	将虚拟网络中的数据封装在实际物理网络中的报文中进行传输	本质上是把 SRH 放入 UDP Payload 中	当遥测数据包经过交换设备时,遥测指令告诉具备网络遥测功能的网络设备应该收集并写入何种网络状态信息	网络流量按照业务逻辑所要求的既定的顺序经过这些业务点
写入信息	VXLAN 头部	SRH	INT 头部与 metadata	SFC 头部
网络层次	L2over L4	L4	L7	L4 – L7
应用场景	解决虚拟内存系统的可移植性限制、云数据中心之间虚拟机迁移	针对 IPv4 网络中 NAT 的问题,实现了 NAT 穿越和中继节点无状态转发等功能	网络运维可视化、故障定位网络测量、拥塞控制、路径决策、流量工程	业务资源池化、NFV 资源池的动态创建和自动化部署、租户业务灵活编排

在当前的实现中,在流水线中处理一个包所需的时钟周期数取决于包的大小。这是因为处理报头和负载的周期数取决于报头和负载长度。如图 6 所示,Corundum 原型平台的接口速率可达到 100 Gbit/s,经测试当数据包大小达到512 B时,Deparser plus 可以在 Corundum 中以线速度处理报文。



(a) 吞吐量

(a) Throughput



(b) 数据包传输速率

(b) Packet transfer rate

图 6　性能测试结果

Fig. 6　Performance testing result

Deparser plus 在 Corundum 原型平台中的资源利用情况如表 4 所示,表格中的内容为所占 FPGA 开发板资源的大小以及占 FPGA 开发板总资源的百分比。实验结果表明,Deparser plus 能够使用较少的资源以线速度进行数据包的协议无关处理。

表 4　Deparser plus 资源利用率

Tab. 4　Deparser plus resource utilization

资源	Deparser plus		Corundum	
	资源大小/个	百分比/%	资源大小/个	百分比/%
CLB LUTs	3 281	0.28	61 460	5.20
CLB Registers	2 297	0.10	38 739	1.69
F7Muxes	495	0.08	5 315	0.86
CLB	665	0.45	7 340	4.97
Block RAM Tile	16	0.74	315	14.58

3　结论

本文扩展 RMT 架构中的 Deparser 模块为 Deparser plus,采用复杂指令集设计,使之支持有状态的复杂报文处理。将 Deparser plus 与采用精简指令集设计的 Ingress RMT、Egress RMT 结合构成一种可编程混合指令集架构 HiRMT,共同完成协议无关的报文处理。经测试,Deparser plus 可以只增加适度的开销到现有 RMT 流水线并提供

线速度的报文处理。

HiRMT 的设计表明,复杂的包操作可以在可编程流水线中处理。但它没有针对性能进行优化,例如:可以通过额外的设计工作来增加流水线的数据宽度,从而提高吞吐量、减小延迟;可以通过优化数据包,从数据包缓存中取出、更新和发送的方式来改善延迟;从 FPGA 转移到 ASIC 将提高频率和数据宽度;更深入的流水线或多个并行解析器也可以提高吞吐量。

当前 Deparser plus 中定义了 7 种复杂指令,它们配合 RMT 架构中的精简指令包含了大多数复杂报文处理所需要的操作。未来的工作可以着眼于拓展 Deparser plus 中的复杂指令集,通过目前已有的 alter 指令向动作表中插入新的指令以支持更多复杂的报文处理操作。

参考文献(References)

[1] CHOWDHURY N M M K, BOUTABA R. A survey of network virtualization[J]. Computer Networks, 2010, 54(5): 862 – 876.

[2] WIJETHILAKA S, LIYANAGE M. Survey on network slicing for Internet of Things realization in 5G networks[J]. IEEE Communications Surveys & Tutorials, 2021, 23(2): 957 – 994.

[3] SHVEDOV A V, GADASIN D V, KLYGINA O G. Determining shortest paths between two arbitrary nodes in a composite transport network using segment routing[C]// Proceedings of the 2021 Systems of Signals Generating and Processing in the Field of on Board Communications, 2021: 1 – 8.

[4] 及晓萌, 梁满贵. 一种向量网可编程交换机实现[J]. 软件, 2013, 34(7): 95 – 99.
JI X M, LIANG M G. Implementation of a vector network programmable switch[J]. Software, 2013, 34(7): 95 – 99. (in Chinese)

[5] 马潇潇, 杨帆, 王展, 等. 智能网卡综述[J]. 计算机研究与发展, 2022, 59(1): 1 – 21.
MA X X, YANG F, WANG Z, et al. Survey on smart network interface card[J]. Journal of Computer Research and Development, 2022, 59(1): 1 – 21. (in Chinese)

[6] 肖征荣, 肖志刚, 王斌, 等. 可编程网络技术概述[J]. 世界电信, 2001(10): 32 – 35.
XIAO Z R, XIAO Z G, WANG B, et al. Programmable network technology overview[J]. World Telecommunications, 2001(10): 32 – 35. (in Chinese)

[7] 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.

[8] 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.

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

[10] WANG T, YANG X R, ANTICHI G, et al. Isolation mechanisms for high-speed packet-processing pipelines[C]// Proceedings of the 19th USENIX Symposium on Networked Systems Design and Implementation, 2022: 1289 – 1305.

[11] CHOLE S, FINGERHUT A, MA S, et al. dRMT: disaggregated programmable switching[C]//Proceedings of the Conference of the ACM Special Interest Group on Data Communication, 2017: 1 – 14.

[12] XING J R, HSU K F, KADOSH M, et al. Runtime programmable switches[C]//Proceedings of the 19th USENIX Symposium on Networked Systems Design and Implementation, 2022: 651 – 665.

[13] MA Z J, BI J, ZHANG C, et al. CacheP4: a behavior-level caching mechanism for P4[C]//Proceedings of the SIGCOMM Posters and Demos, 2017: 108 – 110.

[14] SANTOS AMORIM K, PAVANI G S. Ant colony optimization-based distributed multilayer routing and restoration in IP/MPLS over optical networks[J]. Computer Networks, 2021, 185: 107747.

[15] TULUMELLO A, MAYER A, BONOLA M, et al. Micro SIDs: a solution for efficient representation of segment IDs in SRv6 networks[J]. IEEE Transactions on Network and Service Management, 2022, 20(1): 774 – 786.

[16] CHENG W Q, LIU Y S, JIANG W Y, et al. G-SID: SRv6 header compression solution[C]//Proceedings of the 2020 IEEE 20th International Conference on Communication Technology (ICCT), 2020: 126 – 130.

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

[18] ZHANG Y, PAN T, ZHENG Y, et al. VXLAN-based INT: in-band network telemetry for overlay network monitoring[C]//Proceedings of the IEEE INFOCOM 2021-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), 2021: 1 – 2.

[19] TAN L Z, SU W, ZHANG W, et al. In-band network telemetry: a survey[J]. Computer Networks, 2021, 186: 107763.

[20] ABUJODA A, KOUCHAKSARAEI H R, PAPADIMITRIOU P. SDN-based source routing for scalable service chaining in datacenters[C]// Proceedings of the 14th International Conference on Wired/Wireless Internet Communication, 2016: 66 – 77.

[21] MECHTRI M, GHRIBI C, SOUALAH O, et al. NFV orchestration framework addressing SFC challenges[J]. IEEE Communications Magazine, 2017, 55(6): 16 – 23.