



基于 P4 的 CoLoR 架构控制平面的设计与实现

刘若涵¹, 罗洪斌², 温兴泵¹

(1. 北京交通大学电子信息工程学院, 北京 100044;

2. 北京航空航天大学计算机学院, 北京 100083)

摘要: P4 (programming protocol-independent packet processor, 协议无关的数据分组处理编程) 语言可以通过软件编程的方式重新定义转发设备所识别的字段和处理逻辑。针对新型体系架构 CoLoR 在实际部署和完善时面临的难题, 基于 P4 实现 CoLoR 架构, 着重于对控制平面的实现。介绍了如何用 P4 语言定义 CoLoR 架构相关设备的首部识别和处理逻辑以及如何用 ONOS (open network operating system, 开放式网络操作系统) 实现控制机制, 并在实际中进行部署, 对其进行功能验证和性能测试。

关键词: P4; CoLoR; 开放式网络操作系统; 控制平面

中图分类号: TP393

文献标识码: A

doi: 10.11959/j.issn.1000-0801.2018177

Design and implementation of CoLoR architecture control plane based on P4

LIU Ruohan¹, LUO Hongbin², WEN Xingpeng¹

1. School of Electronic and Information Engineering, Beijing Jiaotong University, Beijing 100044, China

2. School of Computer Science and Engineering, Beihang University, Beijing 100083, China

Abstract: P4 (programming protocol-independent packet processor) can redefine the fields recognized by the forwarding device and its processing logic through software programming. Aiming at the problems that the new architecture CoLoR faces in actual deployment and improvement, the CoLoR architecture was implemented based on P4, and focused on the implementation of its control plane. The way to use the P4 language to define the header recognition and processing logic of CoLoR architecture-related devices, and to use ONOS (open network operating system) to implement the control mechanism was introduced, and it was deployed in practice to perform functional verification and performance testing.

Key words: programming protocol-independent packet processor, CoLoR, open network operating system, control plane

收稿日期: 2018-03-09; 修回日期: 2018-05-11

基金项目: 国家自然科学基金资助项目 (No.61422101); 中央高校基本科研业务费资助项目 (No.YWF-18-BJ-J-61)

Foundation Items: The National Natural Science Foundation of China (No.61422101), The Basic Scientific Research Service Fund of the Central University (No.YWF-18-BJ-J-61)



1 引言

1.1 背景

随着现代科技的进步,网络互联带来的便捷已经体现在人类生活的各个角落,层次化的结构设计为互联网带来的成就不胜枚举。然而不得不引起重视的是,随着用户需求的日益多元化和复杂化,传统网络体系结构在进行功能扩展时存在诸多问题:随着用户规模的不断扩大和各种网络新应用的不断增加,协议的更新迭代使得网络优化的难度增加,同时各种新型服务增加了运维成本。仅仅在现有网络架构的基础上改良已经难以解决其根本问题^[1],北京交通大学下一代互联网互联设备国家工程实验室提出了以信息为中心的CoLoR^[2]新网络体系架构,旨在通过重新设计互联网来实现可扩展和高效的内容分发功能。

必须引起重视的是,无论是创新思想的网络架构还是网络协议,必须在实际环境中部署,才能在最大程度上对其功能与性能进行检测^[3],但新协议或网络架构在现有运营网络中进行部署和测试是复杂的。CoLoR对互联网网络层进行了重新设计,加入了新的字段,随着研究推进,CoLoR的报头格式和逻辑在不断完善、改进。在现有网络环境中部署测试时,为了支持新的协议字段/报头,使得协议首部内容不断增加、标准更加复杂^[4],此外还需要对每个网络设备进行配置和更新操作。在这种情况下,CoLoR架构的完善与推广是一个缓慢且极具挑战性的过程,因为它涉及在网络中执行新的、非常规的处理逻辑和匹配内容,除了仍然需要解决的算法挑战,描述一些CoLoR实例显然缺乏硬件支持。因此,需要一种更灵活的系统实现方法,使CoLoR可以很方便地被部署、测试、调整、升级,这对提高对新协议、新网络构架的研究效率,加快业界对CoLoR的接纳、部署和推广,是十分必要的。

P4^[5](programming protocol-independent packet

processor)作为一种潜在的“OpenFlow2.0”的发展方向,区别于现有的OpenFlow通过流表指导固定功能^[6]的交换机转发数据流,P4通过软件编程的方式重新定义转发设备所识别的字段以及对数据分组的处理流程和逻辑,旨在解决OpenFlow可编程性和可拓展性方面的局限性问题。

P4有其特有的交换机模拟器BMv2(behavior model version 2),P4的核心思想可以用抽象转发模型进行描述,其中有两个主要操作:配置和填充^[7]。配置操作完成对分组处理器的编程,通过JSON格式的配置文件指定每个阶段处理的首部字段,设置BMv2在match+action^[5]阶段所要匹配的内容和顺序,并定义对应动作域;填充操作主要是进行具体流表条目的添加或删除。

P4的“全”可编程性体现在其可重构性、协议无关性和目标独立性3个方面^[8],其中协议无关性指的是它不与任意一个特定协议绑定,所有的流规则、流表和执行顺序都由控制器制定后下发给交换机。P4的优势体现在对新的网络协议与架构进行验证与测试以及版本的迭代更新,与传统设备需要重新修改协议栈并进行重新配置相比,这种通过可编程来重新定义交换机的方式大幅度加快了科学研究的进程。

1.2 现状

参考文献[9]使用P4编写NDN(named data networking,数据命名网络)数据平面,流表通过定义一个软件API来下发,但是当系统较为复杂、功能模块种类繁多时,仅仅使用API能否实现智能的流表计算和下发并不能得到保证。

CLI(command-line interface,命令行接口)可以对BMv2进行简单流表的下发。但是一个CLI只能连接一个BMv2,不能提供针对全局网络状态制定的智能决策。为了实现新架构可编程、自动化的网络控制能力,需要一个更高可扩展性、更智能的控制平面。ONOS^[10,11]是首款开源SDN^[12](software defined networking,软件定义网络)操

作系统,参考文献[13]在对 ONOS 的研究中加入了一个简单的针对 BMv2 的控制模型,但是对 CoLoR 提供的支持仍需完善。ONOS 只支持如 TCP/IP 和以太网 Ethernet 等现有的协议,而 CoLoR 中的各种分组格式需要自行开发,而且要制定复杂的网络策略、实现智能的转发仍有许多问题尚待解决。

ONOS 启动后需要激活对 BMv2 的驱动(driver),并开启默认监听端口 40123, BMv2 启动时指定该端口并主动请求连接,并告知 ONOS 其自身的 thrift-port^[14]。之后 ONOS 与 BMv2 通过两条 TCP 连接交互:链路 1 通过 40123 端口接收 BMv2 上传的问询消息,链路 2 通过 thrift 接口实现 ONOS 对 BMv2 的管控,如配置文件和流表的下发以及通过远程接口调用控制 BMv2 构造数据分组以及指定数据分组的出端口等。

具体原理如图 1 所示。原始 P4 文件经过编译器(p4c-bmv2)转化为 JSON 格式的配置文件^[15],转化器(interpreter)将配置文件转化成设备描述表(device-context),驱动中建立设备(device)和默认设备描述表之间的映射,之后北向抽象层通过 API 将配置文件和流表下发,并将网络视图提供给应用层。

2 CoLoR 架构及其机制

与当前仅使用一个命名空间(即 IP 地址)的

TCP/IP 网络不同,CoLoR 使用两个全局命名空间:SID(server ID, 服务标识符)、NID(node ID, 节点标识符)和两个本地命名空间:域内路由定位、PID(path ID, 路径标识符)。

区别于传统网络架构以主机为中心的寻址方式,CoLoR 贯彻以信息为中心的思想,为网络中每个服务内容分配唯一的标识符 SID, SID 仅用于代表某项内容,与其所处位置无关,服务注册和请求时均依据 SID; NID 代表网络中各节点的位置,用于认证;域内路由定位用于在域内进行路由; PID 用于在域间定义路径,由两个域协商产生,不全球通告。

在 CoLoR 中,每一个 AS(autonomous system, 自治域)中都有一个 RM(resource manager, 资源管理器),自治域内所有服务提供方都会把 SID 向本域 RM 进行注册,故而 RM 维护着一个存放内容可达性信息的注册表,用于对所请求服务的查询; BR(border router, 边界路由器)用于连接其他自治域并沿 PID 标识路径转发 CoLoR 分组。

CoLoR 的基本工作机制包括:服务注册、内容请求与数据分组转发。一个完整的 CoLoR 系统的运行流程如图 2 所示,为了便于阐述,选择域 D2 作为父域, RM2 为父域 RM, 其他域都需向 D2 通告。

2.1 服务注册

服务注册包括域内注册和域间通告两个方

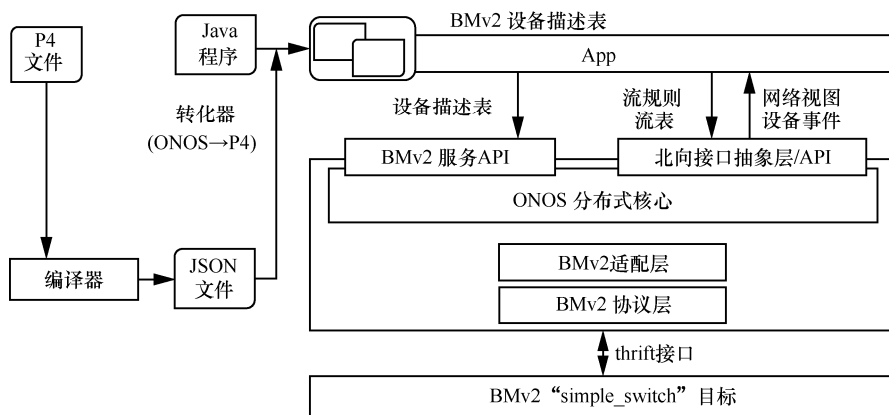


图 1 ONOS 控制 P4 原理

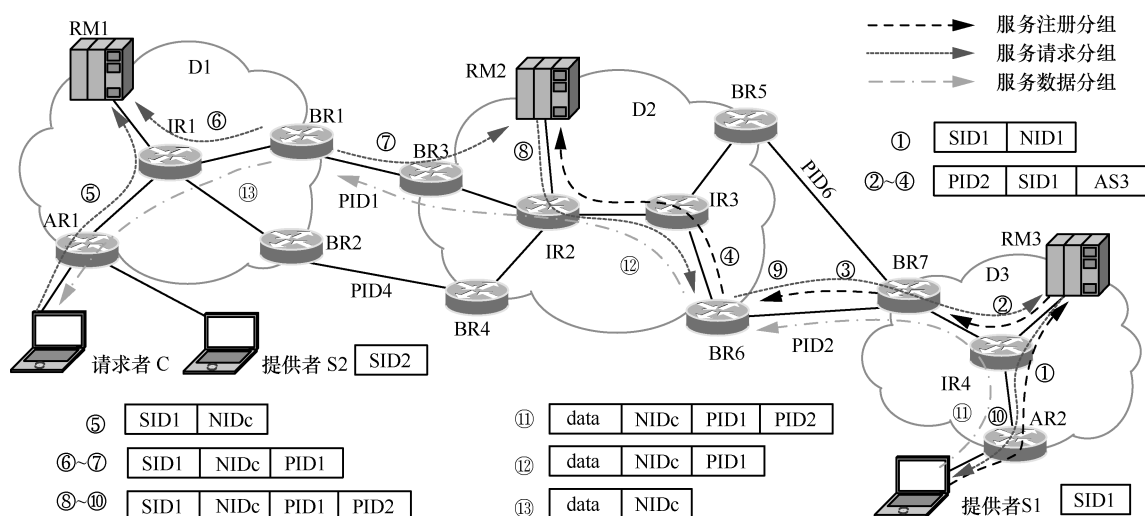


图2 CoLoR 架构运行流程

面,如图2中过程①~④所示。D3中服务提供者S1向本域RM3发送注册信息,注册内容包括其SID和提供者自身NID, RM3收到注册分组后,会在自身的注册表里为这一SID添加条目存储注册信息,此即服务的域内注册部分;随后RM3还需选择域间路径PID2向其父域的RM(D2的RM2)进行通告,通告内容为SID、通告方AS号和绑定PID,被通告方RM2需添加对该SID的通告条目,此即域间通告的流程。

同理D1中S2也向其RM1发送注册分组,注册信息为(SID2、NID2), RM1向RM2进行通告,通告分组内容为(SID2、AS1、PID1)。

2.2 内容请求

请求者期望获取某项服务时,会向其本地RM发送get请求消息。get消息应包含请求者所期望的服务的SID和其自身NID。若RM在注册表里找不到SID的条目,会发往父域问询;若SID注册在本域,则直接将请求分组转发至服务提供者。

如图2中过程⑤~⑩所示,当D1中请求者C期望获取由SID1标识的服务时,会向其本地RM1发送get请求消息;RM1在注册表里查询不到SID1的条目,会将路径标识PID1添加到请求分组尾部,并沿该路径将这一get请求消息转发至父

域RM2; RM2收到get分组后,查询到本域存储了由D3通告来的关于SID1的通告条目,则依据通告信息将PID2封在get分组尾部,并沿该路径将请求分组转发至RM3; RM3查询SID1条目发现注册在本域内,则依据注册信息查找提供者NID(NID1),将get分组转发至S1。

同理,当C请求SID2对应的服务时发送get分组(SID2、NIDc), RM1发现SID2注册在本域,直接依据注册信息查找NID2,将get分组转发至S2。

2.3 数据转发

一旦存储所需服务的源节点接收到get消息,服务提供者即可得知请求者的NID、所期望服务的SID以及途经的PID。因此,如图2中过程⑪~⑬所示, D3中S1收到由C发来的get分组后,发送封装请求者NID、PID和所请求内容DATA的数据分组回送,跨域过程逐级剥去最外层PID,从BR7经PID2到BR6后剥去外层PID2,从BR3经PID1到BR1后剥去PID1,最后依据NIDc将所需服务转发至请求者C。

同理D1中S2收到C发送的服务请求分组后,会发送数据分组(DATA、NIDc),经AR1后直接依据NIDc将所需服务转发至请求者C。

CoLoR 分组头引入了新的字段如 SID、NID、PID 等,同时通过对 CoLoR 机制的分析不难发现,在服务请求发起到请求者接收到数据分组的过程中, RM、BR 中需要经过复杂的逻辑处理, PID 一直处于动态增减的状态。在现有网络中,新字段的添加需要对整个协议栈进行修改,服务注册、请求和数据分组转发也涉及在网络层执行新的、非常规的处理逻辑和规则,需要对每个网络设备进行配置和更新,以支持新的协议字段/报头。故而 CoLoR 的完善和推广受实际部署的限制。

而 P4 的可编程性通过对交换机可识别的首部字段和处理逻辑进行重新定义,可以以一种十分便捷的方式改变对数据分组的处理方式。用 P4 实现的 CoLoR 架构继承了协议无关的特性,控制器 ONOS 通过配置文件定义 CoLoR 分组格式和底层转发设备 BMv2 处理 CoLoR 分组的逻辑,提取解析转发设备不能识别的分组决策并进行相应的处理,实现 CoLoR 架构控制平面的功能。

3 用 P4 编写 CoLoR 架构

用 P4 实现 CoLoR 架构时,各域中位于数据平面的 BMv2 设备(RM、BR、AR、IR 等)都仅仅保留其转发作用,其所能识别的首部和控制逻辑都由控制平面 ONOS 制定。ONOS 检测到 BMv2 连接时,通过 TCP 连接将配置文件以 JSON 格式下发,CoLoR 分组到达 BMv2 后询问 ONOS 获取转发所需流表,此后 BMv2 才具有独立转发的能力,后续 CoLoR 分组查询 BMv2 中的流表匹配后直接进行转发,匹配失败才会再次询问 ONOS。以 get 分组为例,介绍如何使用 P4 定义 BMv2 所能识别的首部字段和处理逻辑及相关问题。

3.1 P4 编写配置文件

(1) 首部(header)

图 3 给出了 P4 语言定义的 get 分组头类型,首部定义了 get 分组头各字段及其长度,这里给出了固定长度部分,由于跨域过程中会增删 PID,

PID 字段为可变长度,在第 3.2 节中具体讨论。每一种首部类型都有对应的首部实例来存储具体数据^[16]。

```
header_type get_t {
    fields {
        version_type : 8;
        ttl : 8;
        total_len : 16;
        port_no1 :16;
        port_no2 :16;
        minpid :16;
        pids_o :8;
        flag : 8;
        mtu : 16;
        checksum :16;
        sid :288;
        nid_c :128;
        mac :32;
        offset :32;
        len :32;
    }
}
```

图 3 P4 语言定义的 get 分组头类型

(2) 解析器(parser)

图 4 给出了 P4 语言定义的解析器,解析器规定了 BMv2 可以解析的分组头和解析顺序,在第 3.2 节中具体阐述。

```
parser parse_get{
    extract(get);
    return select(get.pids_o){
        0xFF: parse_newPid;
        default: parse_for_merge;
    }
}
parser parse_newPid{
    extract(newPid);
    return parse_for_merge;
}
parser parse_for_merge{
    return select(get.pids_o){
        0x00: ingress;
        default: parse_firstPid;
    }
}
parser parse_firstPid{
    extract(firstPid);
    return ingress;
}
```

图 4 P4 语言定义的解析器

(3) 匹配动作表(table)^[17]

图 5 给出了 RM 处理 get 分组的一条匹配动作表: table sid_nid, 表中定义了匹配字段、对应动作和表容量, RM 提取 SID 进行精确匹配,依据匹配结果分别执行对应的动作:若 SID 在域内



则修改源地址、目的地址转发,若 SID 在域外则添加 PID 后转发,若匹配失败,会在 table_miss 后执行提前设定的 default_action 即 send_to_cpu 问询控制器。

```

action inside_sid_set(srcAddr,dstAddr,res) {
    modify_field(ipv4.srcAddr, srcAddr);
    modify_field(ipv4.dstAddr, dstAddr);
    modify_field(get.res, res);
}
action outside_sid_set(srcAddr,dstAddr,res,addPid){
    modify_field(ipv4.srcAddr, srcAddr);
    modify_field(ipv4.dstAddr, dstAddr);
    modify_field(get.res, res);
    add_to_field(get.pids_o,0x1);
    add_header(newPid);
    modify_field(newPid.PID, addPid);
}
table sid_nid {
    reads {
        get.sid: exact;
    }
    actions {
        inside_sid_set;
        outside_sid_set;
        send_to_cpu;
        _drop;
    }
}

```

图5 P4定义的匹配动作表

(4) 流控制程序 (control ingress)

图6给出了P4定义的流控制程序。流控制程序规定了匹配动作表的执行条件和顺序。当RM判断分组类型为get分组时,进入流表table sid_nid进行匹配转发。

```

control ingress {
    if(etherType == ETHERTYPE_IPV4 and ipv4.protocol == IPPROTO_COLOR_GET){
        apply(sid_nid);
    }
}

```

图6 P4定义的流控制程序

3.2 变长问题

解析器提取分组头并依据现有分组头字段判断接下来解析分组头的类型,解析的最终结果是进入流控制程序处理数据分组。当解析器工作时,会将当前处理的数据分组头字节的偏移量记录在首部实例中,并在状态迁移(即调用另一个解析器)时指向分组头中下一个待处理的有效字节^[18]。现以RM处理get过程为例简要分析变长问题,解析流程如图7所示。

(1) 解析器

get分组到来时首先提取 Ethernet 分组头解

析,并依据 etherType 字段判断: 0x0800 解析 IP 分组, 否则停止解析, 进入流控制程序处理 Ethernet 分组; 解析完成后偏移至 IP 分组头字段, 提取 IP 分组头, 解析并依据 protocol 字段判断接下来解析分组头类型: 0xA0 为请求分组, 0xA1 为数据分组, 0xA2 为注册分组, 0xA3 为通告分组, 否则停止解析处理 IP 分组; 之后偏移至 get 分组头字段, 提取 get 分组头, 并依据 pid_o 字段判断: pid_o=0xFF 则解析 newPid 字段, 0xFF 为理论上不可能达到的值, 故解析过程通常不会提取 newPid, 仅用于对 deparser (逆解析) 阶段的占位, 默认进入下一个解析 for_merge, 依据 pid_o 进行判断: 若为 0 则结束解析处理 get 分组; 若不为 0 代表这个 get 分组已经跨域, 解析 firstPid 字段, 最终会仍进入流控制程序, 结束解析。

(2) 流控制程序

进入流控制程序, 满足条件后进行流表匹配, 完成相应动作。RM发现SID在域外时执行 action: add_header 添加 newPid, 在域内则依据 NID 转发。

(3) 逆解析器

将提取出来的各分组头字段组合, extract 和 add_header 动作操作的首部都会被激活变为 valid, 之前占位的 newPid 被激活, 故而组合时会增加 newPid 分组头。需要注意的是, PID 采取倒序的方式插在 get 消息之后, 最新加入的 PID 添加在所有 PID 最前边即 firstPid。

3.3 首分组问题

BMv2 通过 thrift 接口向 ONOS 提供远程接口调用服务。ONOS 为 BMv2 下配置文件和具体流表, 也可以控制 BMv2 构造 CoLoR 分组从特定端口发出。这里以 get 分组为例, 介绍远程接口调用处理首分组问题。

(1) 对于首个到达 BMv2 的 get 分组

AR 连接到 ONOS 后, 其初始转发流表 “protocol_dstAddr” 为空 (发送给控制器问询的默认流表不计), get 分组到达 AR 后进入其

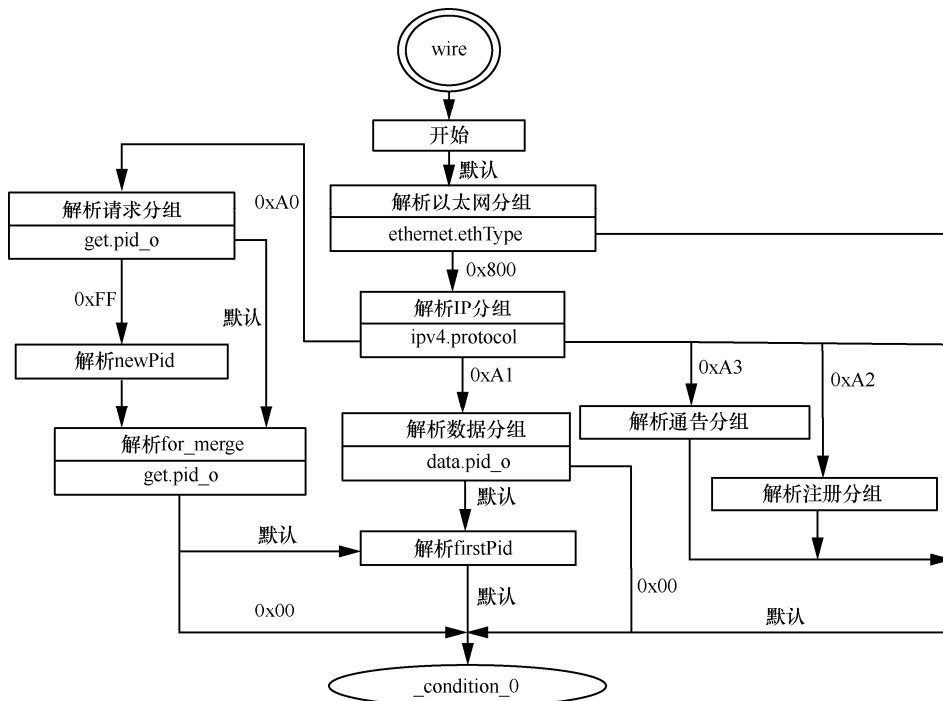


图7 解析流程

match+action 管道，由于无法正确匹配而 table_miss，执行默认动作 send_to_cpu，通过 TCP 连接问询 ONOS，此时 get 分组完成 match+action 过程，离开该管道。ONOS 对请求信息进行解析后，计算 AR 至 RM 的路径并为该链路上所有设备下发流表。同时通过远程接口调用，控制 AR 生成 get 分组并直接从其对应端口发送出去，而不再经过 AR 的 match+action 管道查询流表转发；get 在各 IR 依据下发的流表匹配转发；到达 RM 时，RM 转发流表为空，问询 ONOS，下发“sid_nid”流表并通过远程接口调用发送 get 分组。

(2) 对于之后到来的 get 分组

由于 AR、RM 等已经具有转发逻辑和流表，具有独立的转发功能，get 分组依据流表直接转发，无需问询 ONOS，匹配失败执行默认动作 send_to_cpu。

通过远程接口调用转发 CoLoR 分组时，由于 CoLoR 分组不再进入 match+action 管道，要求 ONOS 完成下发给这个 BMv2 流表的所有功能，远程接口调用只实现了对出端口的指定，但对分

组内容的修改，需要 ONOS 自身完成，这一问题将在第 4.3 节中进行分析解决。

4 基于 P4 的 CoLoR 架构控制平面的实现

图 8 给出了 ONOS 的控制平面实现框架。实现的过程中涉及多个 ONOS 提供的服务：其中 deviceService 用于控制 BMv2 切换 JSON 文件和查询流表信息等；flowService 实现对流表从 selector+treatment^[19]到 match+action 的转换和持久化下发等；packetService 接收来自 BMv2 的各种分组。

CoLoR 初始化主模块接收 BMv2 提交的包含 CoLoR 信息的 TCP 分组，接收分组分析模块解析 CoLoR 分组的来源以及分组类型，进行分流使之进入相应的处理过程：这些处理方法中调用了流规则生成模块定义的流规则，具体的匹配参数由分组头读写模块提取解析得到，并依路径计算模块选路结果为其传入流表的具体动作参数，应用流表对 CoLoR 分组进行相应处理。需要注意的是，其中首分组运用分组头读写模块修改分组相应字

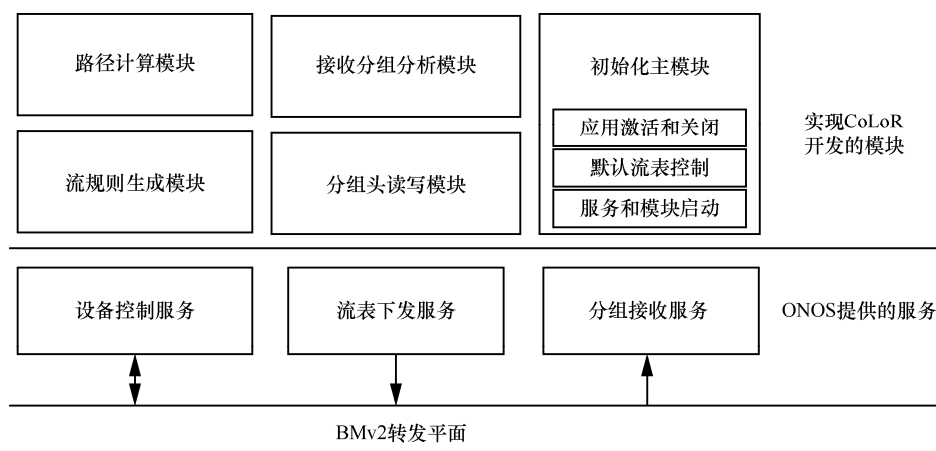


图8 ONOS 的控制平面实现框架

段的内容，并通过远程接口调用传送分组。之后到来的 CoLoR 分组到达 RM、BR 等设备时，由于已经下发配置文件和流表，可以直接查询流表，匹配成功直接转发无需问询 ONOS，匹配失败才会执行默认动作 `send_to_cpu`。

4.1 初始化主模块

CoLoR 初始化主模块为 ONOS 进行分组处理的主模块，主要实现的功能有与 BMv2 建立连接、接收含有 CoLoR 信息的 TCP 分组以及控制流表下发。

本文中流表下发模式设计为主动与被动结合。ONOS 检测到 BMv2 连接时就要主动下发一些特定的默认控制流表（如丢弃报文、对下一张流表进行匹配、转发给控制器等），其余流表当 BMv2 问询时，依据对 CoLoR 信息解析后生成，通过 TCP 连接下发。被动模式的好处是当规模扩大时，BMv2 无需时刻维护全部的流表，只有当产生实际流量时才向 ONOS 获取流表并存储。且每条流表都设有定时器，超时后删除，故而在很大程度上节省存储空间。

4.2 接收分组分析模块

接收分组分析模块对到来的 CoLoR 分组进行分流。ONOS 对 CoLoR 分组进行初步解析，获得发送端 BMv2 的设备信息（如 `deviceId`、`deviceName` 等）和 CoLoR 分组类型信息（`versionType`），并据

此分流，将 CoLoR 分组送到对应的分组处理函数，理论上所有的 BMv2 设备（RM、AR、BR 等）都应实现对注册分组、通告分组、请求分组、数据分组的处理逻辑。

4.3 分组头读写模块

（1）分组头读写模块解析 CoLoR 分组，为流规则生成模块提供匹配参数，具体解析 CoLoR 分组信息如 `SID`、`NID`、`PID` 和 `versionType` 等字段，可作为路径计算的源、目的地址依据或者作为流控制程序中流表执行的判断依据。

（2）分组头读写模块修改 CoLoR 分组内容，包括源地址、目的地址和标志位 `flag` 等，是为了解决远程接口调用时的首分组问题。ONOS 远程接口调用指定出端口，但对分组内容的修改，需要分组头读写模块完成，继续以 `get` 为例，简要介绍分组头读写模块对分组内容的修改。

- 修改源地址、目的地址是为了实现 CoLoR 分组正常转发：`get` 首分组到达 AR 后需要将目的地址修改为本域 RM 的地址，并从相应端口转发出去。而 ONOS 通过远程接口调用为 `get` 指定了出端口，但目的地址仍为 AR，从正确端口到达 IR 后，查询 `ip_port` 后由于目的地址有误最终仍旧无法正常到达 RM，因此需要在 ONOS 中完成对 CoLoR 分组内容的修改，将目的地址改为本域 RM

的地址,只有修改过目的地址的 get 分组才能依据 match+action 传递给 RM,其他过程同理,不再细述。

- 设置 flag 标志位是为了区分 CoLoR 分组路径:到达 AR 的 get 分组有两种情况,即从客户端到 RM 和从 RM 到服务器;同样到达 BR 的 get 分组有两种情况:RM 到外域 BR 和外域 BR 到 RM。标志位 flag 对分组路径进行区分,默认从客户端发出的原始服务请求分组 flag=0,经过 AR 走客户端→RM 路径;经过 RM 处理后,改为 flag=1;到达本域 BR 走 RM→BR 路线,同时使 flag=0;到达外域 BR 走 BR→RM 路线,到达 RM 再次使 flag=1;此时到达 AR 走 RM→服务器路线,完成 get 路径区分。

4.4 路径计算模块

路径计算模块用于计算路径,为流规则生成模块提供动作域参数。读取配置文件的链路信息和设备映射信息,提取并解析 CoLoR 分组的源、目的信息,计算以向 ONOS 发送问询的 BMv2 为根生成的最短路径树,并选取到目的路径的这条,得到所需路径信息(包括经过的设备以及该设备的出端口),将端口信息作为流表动作域参数发送给对应设备。

4.5 流规则生成模块

流规则生成模块是具体的流规则制定模块。

流规则的组成部分可以划分为 selector、treatment、forTable、deviceId 和 JSON^[20]。其中 selector 部分用于条件匹配,包括匹配类型和匹配字段;treatment 部分用于匹配达成后相应动作的执行;deviceId 确定流条目所对应的设备;forTable 用于指定流条目对应 JSON 文件中的流表项;JSON 用于将默认配置文件与 BMv2 当前文件对应,定期检测,发现不同后迅速切换。

5 结果检测

为对基于 P4 的 CoLoR 架构控制平面进行功能验证并测试其性能,在实验室实体机柜上部署了测试拓扑,使用 Click^[16]路由器模拟发分组。测试拓扑如图 9 所示,其中沿 PID1、PID2 出去的设备 PID1、PID2 为虚拟主机,用于监控 PID 链路上的分组信息。

5.1 功能检测

本文选择一个域 D1 为研究对象,通过对从 D1 发送出去的 CoLoR 分组和到达 D1 的 CoLoR 分组的转发结果验证其功能,其他域同理。

(1) 注册分组

本域通告给外域:D1 中的 server1 发送注册分组,注册信息为“inside”(36 byte,其余部分用“_”补全,均省略,下同)。图 10 给出了 ONOS1 收到域内注册分组提取出的相关内容,由 versionType=0xA2 判断这是注册分组,

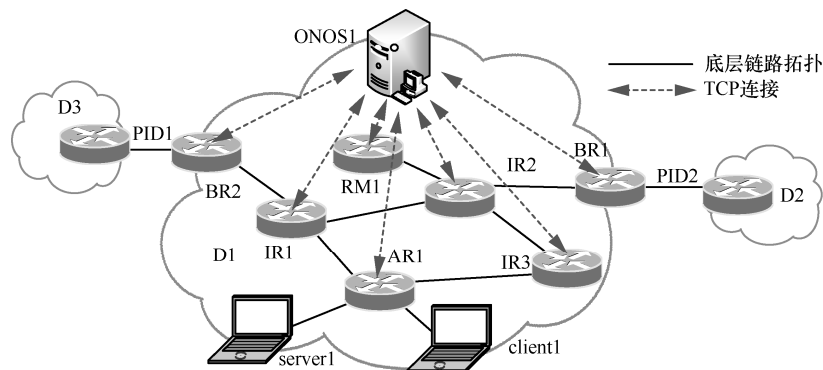


图 9 测试拓扑

Time	No.	Source	Destination	Protocol	Length	Info
3.997813000	10	192.168.11.211	192.168.85.12	IPv4	118	Unassigned
4.999560000	11	192.168.11.211	192.168.85.12	IPv4	118	Unassigned
5.998645000	12	192.168.11.211	192.168.85.12	IPv4	118	Unassigned

▶ Frame 10: 118 bytes on wire (944 bits), 118 bytes captured (944 bits) on interface 0
 ▶ Ethernet II, Src: aa:aa:aa:aa:aa:aa (aa:aa:aa:aa:aa:aa), Dst: bb:bb:bb:bb:bb:bb (bb:bb:bb:bb:bb:bb)
 ▶ Internet Protocol Version 4, Src: 192.168.11.211 (192.168.11.211), Dst: 192.168.85.12 (192.168.85.12)
 Version: 4
 Header length: 20 bytes
 ▶ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECN-Capable Transport))
 Total Length: 108
 Identification: 0x0027 (39)
 ▶ Flags: 0x00
 Fragment offset: 0
 Time to live: 250
 Protocol: Unassigned (161)

```

0000  bb bb bb bb bb bb aa aa aa aa aa 00 00 45 00  .....E.
0010  00 6c 00 27 00 00 fa 53 51 c0 a0 0b 43 c0 a0  ..U...
0020  55 0c a1 00 00 00 00 00 00 00 00 00 01 00 00 00  ..U...
0030  00 00 7f 5f 5f 5f 5f 5f 5f 5f 5f 5f 5f 5f 5f 5f  ..W
0040  5f 5f 5f 5f 5f 5f 5f 5f 5f 5f 5f 5f 5f 5f 5f 5f  ..W
0050  5f 5f 5f 5f 5f 5f 43 6c 69 65 6e 74 31 5f 5f 5f  ..cl ient1
0060  5f 5f 5f 5f 5f 5f 00 00 00 00 00 00 00 00 00 00  ..PID1
0070  00 00 50 49 44 31
  
```

图 15 PID2 抓取数据分组结果

Time	No.	Source	Destination	Protocol	Length	Info
15.953003000	20	192.168.11.211	192.168.11.222	IPv4	114	Unassigned
16.955354000	21	192.168.11.211	192.168.11.222	IPv4	114	Unassigned
17.952631000	22	192.168.11.211	192.168.11.222	IPv4	114	Unassigned

▶ Frame 20: 114 bytes on wire (912 bits), 114 bytes captured (912 bits) on interface 0
 ▶ Ethernet II, Src: aa:aa:aa:aa:aa:aa (aa:aa:aa:aa:aa:aa), Dst: bb:bb:bb:bb:bb:bb (bb:bb:bb:bb:bb:bb)
 ▶ Internet Protocol Version 4, Src: 192.168.11.211 (192.168.11.211), Dst: 192.168.11.222 (192.168.11.222)
 Version: 4
 Header length: 20 bytes
 ▶ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECN-Capable Transport))
 Total Length: 100
 Identification: 0x0006 (6)
 ▶ Flags: 0x00
 Fragment offset: 0
 Time to live: 250
 Protocol: Unassigned (161)

```

0000  bb bb bb bb bb bb aa aa aa aa aa 00 00 45 00  .....E.
0010  00 64 00 00 00 00 fa 53 ba c0 a0 0b 43 c0 a0  ..U...
0020  0b de a1 00 00 00 00 00 00 00 00 00 00 00 00 00  ..U...
0030  00 00 71 5f 5f 5f 5f 5f 5f 5f 5f 5f 5f 5f 5f 5f  ..q
0040  5f 5f 5f 5f 5f 5f 5f 5f 5f 5f 5f 5f 5f 5f 5f 5f  ..W
0050  5f 5f 5f 5f 5f 5f 43 6c 69 65 6e 74 31 5f 5f 5f  ..cl ient1
0060  5f 5f 5f 5f 5f 5f 00 00 00 00 00 00 00 00 00 00  ..
0070  00 00
  
```

图 16 client1 抓取数据分组结果

“client1”，因为 pid_o=0，表示请求者在本域，直接依据 NIDc 将 data 分组转至 client1。

5.2 性能测试

(1) ONOS 流表下发速率

ONOS 通过 thrift 端口对 BMv2 下发流表，其流表的下发性能标志着 ONOS 对 BMv2 的控制能力。测试中 ONOS 和 BMv2 分别部署在两台物理机上，并通过普通的二层交换机建立 TCP 连接，测试流表分为两种：规模较小的流表“ip_port”匹配域和动作域种类少，字段长度为 5 byte，信息量较少；规模较大的流表“sid_nid”匹配域和动

作域种类多，字段长度为 65 byte，信息量较多。

如图 17 所示，测试结果表明，ONOS 对 BMv2 流表下发的平均性能大致在 750 条/s（约每 1.3 ms 添加一条流表），而且流表规模较小时性能更稳定，随着流表条目增加，性能下降较为缓慢。在测试过程中同时发现，当 ONOS 在下发流表总数达到 13 万时到达瓶颈，下发速率趋近于零，无法完成测试。ONOS 下发流表后，同时自己缓存了一份相同的流表并定期下发，用于 BMv2 断开重连后的流表恢复，下发总流表数目越多，ONOS 需要缓存的流表数越多，因此导致性能到达瓶颈后急剧下降。因此在 CoLoR 中的 SID 的映射转发不能全部存储到 BMv2 中，过期的 SID 条目会被 BMv2 和 ONOS 移除，get 分组匹配失败后向 ONOS 询问该流表即可。

(2) JSON 文件切换时延

ONOS 切换 JSON 文件有两种情况：一是 BMv2 设备中的 JSON 文件出现异常如重启时，ONOS 定时检测，发现其与默认文件不匹配时切换；二是 CoLoR 版本迭代更新时 ONOS 主动更新默认配置文件。JSON 文件切换的过程会不可避免地出现丢失分组从而导致服务中断，JSON 文件切换时延越短越好，如图 18 所示。测试结果表明，JSON 文件的切换时延与大小的关系为正相关，如本次测试对象为 AR、RM、IR、BR 4 种设备对应的 JSON 文件，其中 BR 处理逻辑最为复杂，JSON 文件最大为 59.1 KB，切换时延为 4.648 ms。

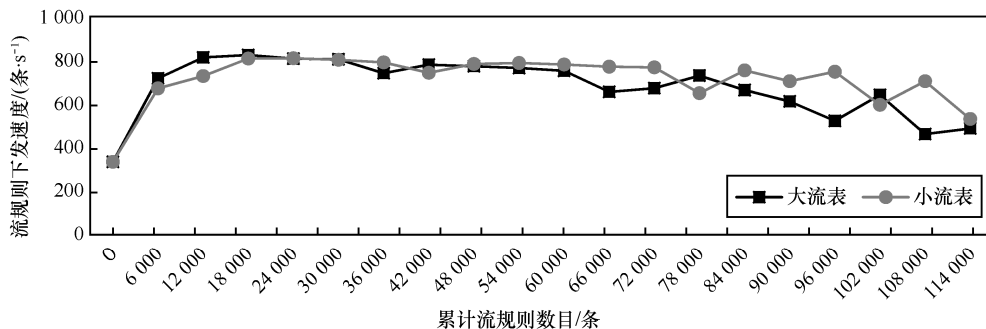


图 17 下发速率随流表数目变化

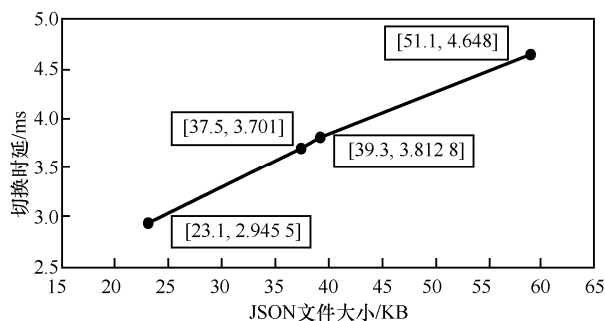


图 18 JSON 文件切换时延

6 结束语

本文简要介绍了新网络架构 CoLoR 的运行原理, 分析了 CoLoR 架构推广受到限制的因素, 在此基础上用 P4 实现 CoLoR 架构, 详细阐述了基于 P4 的 CoLoR 架构控制平面的设计与实现, 介绍如何用 P4 语言定义 RM、BR、AR、IR 等设备的可以识别的首部字段和处理逻辑以及如何用 ONOS 控制 P4 实现 CoLoR 架构流程, 并对其进行功能验证和性能测试。

然而, 围绕基于 P4 实现 CoLoR 架构, 仍然有许多亟待解决的问题。本文重点侧重于功能实现, 如何在保证系统稳定运行的基础上提升性能以及配置文件的不中断切换等, 有待进一步实现。

参考文献:

- [1] 罗洪斌, 张宏科. 智慧协同标识网络体系: 研究背景、思路与进展[J]. 电信科学, 2015, 31(2): 11-21.
LUO H B, ZHANG H K. Smart and cooperative networks: background, basic ideas and progress[J]. Telecommunications Science, 2015, 31(2): 11-21.
- [2] LUO H, CHEN Z, CUI J, et al. CoLoR: an information-centric internet architecture for innovations[J]. IEEE Network, 2014, 28(3): 4-10.
- [3] 王歆平, 王茜, 刘恩慧, 等. 基于 SDN 的按需智能路由系统研究与验证[J]. 电信科学, 2014, 30(4): 8-14.
WANG X P, WANG Q, LIU E H, et al. Research and verification on SDN-based on-demand smart routing system [J]. Telecommunications Science, 2014, 30(4): 8-14.
- [4] 王晓峰, 吴建平, 崔勇. 互联网 IPv6 过渡技术综述[J]. 小

型微型计算机系统, 2006, 27(3): 385-395.

- WANG X F, WU J P, CUI Y. Survey of internet IPv6 transition technologies[J]. Journal of Chinese Computer Systems, 2006, 27(3): 385-395.
- [5] BOSSHART P, IZZARD M, IZZARD M, et al. P4: programming protocol-independent packet processors[J]. ACM SIGCOMM Computer Communication Review, 2014, 44(3): 87-95.
- [6] 王振法, 王雷, 高翔宇, 等. POF 环境下的一种内容中心网络架构及其实现[J]. 小型微型计算机系统, 2016, 37(9): 1959-1963.
WANG Z F, WANG L, GAO X Y, et al. Architecture and implementation of content-centric networking under POF environment[J]. Journal of Chinese Computer Systems, 2016, 37(9): 1959-1963.
- [7] Behavioral-model[EB]. 2016.
- [8] p4language[EB]. 2017.
- [9] SIGNORELLO S, STATE R, FRANÇOIS J, et al. NDN.p4: programming information-centric data-planes[C]//Netsoft Conference and Workshops, June 10, 2016, Seoul, Korea. Piscataway: IEEE Press, 2016: 384-389.
- [10] What is ONOS?[EB]. 2015.
- [11] VOELLMY A, WANG J. Scalable software defined network controllers[J]. ACM SIGCOMM Computer Communication Review, 2012, 42(4): 289-290.
- [12] 何晓明, 冀晖, 毛东峰, 等. 电信 IP 网向 SDN 演进的探讨[J]. 电信科学, 2014, 30(6): 131-137.
HE X M, JI H, MAO D F, et al. Discussion of evolution of carrier IP network to SDN [J]. Telecommunications Science, 2014, 30(6): 131-137.
- [13] P4 experimental support via BMv2[EB]. 2018.
- [14] BERDE P, GEROLA M, HART J, et al. ONOS: towards an open, distributed SDN OS[C]//The Workshop on Hot Topics in Software Defined Networking, August 22, 2014, Chicago, IL, USA. New York: ACM Press, 2014: 1-6.
- [15] The P4 language specification, version 1.1.0-release candidate[EB]. 2018.
- [16] P4 status update: where are we now and what's next?[EB]. 2017.
- [17] 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.
- [18] Openstate.p4: supporting stateful forwarding in P4[EB]. 2018.
- [19] HAN Y, RYU S, SUH Y J, et al. Design and implementation of LISP controller in ONOS[C]//Netsoft Conference and Work-

shops, June 10, 2016, Seoul, Korea. Piscataway: IEEE Press, 2016: 417-422.

[20] PARULKAR G, TOFIGH T, LEENHEER M D. SDN control of packet over optical networks[C]//Optical Fiber Communications Conference and Exhibition, March 22-26, 2015, Los Angeles, CA, USA. Piscataway: IEEE Press, 2015: W1G4.

[21] KOHLER E. The click modular router[M]. New York: ACM Press, 2001.

[作者简介]



刘若涵（1993-），女，北京交通大学电子信息工程学院下一代互联网互连设备国家工程实验室硕士生，主要研究方向为未来网络体系架构和协议无关转发机制。



罗洪斌（1977-），男，北京航空航天大学计算机学院教授、博士生导师，中国通信学会第四届青年工作委员会委员，北京通信学会第八届理事会理事，入选教育部新世纪优秀人才支持计划，并获得国家自然科学基金委员会优秀青年基金资助。主要从事未来互联网体系架构、理论与关键技术的研究工作，主持和参与多项国家“973”计划、“863”计划、国家自然科学基金项目。



温兴泵（1992-），男，北京交通大学电子信息工程学院下一代互联网互连设备国家工程实验室硕士生，主要研究方向为高性能的查找与转发机制。