



OVS 的编程扩展技术

杨帆, 晏思宇, 黄韬

(北京邮电大学, 北京 100876)

摘要: OVS (open vSwitch) 作为一款开源的 SDN 软件交换机, 得到了广泛关注。但在复杂多变的网络环境下, 易出现数据层失去控制器控制和流表项学习老化等故障问题, 严重制约了 SDN 性能。针对以上问题, 研究设计了 OVS 的编程扩展技术架构, 包含链接状态处理、故障感知、二三层自学习转发等核心功能的扩展, 使 OVS 实现自动快速恢复、流表项及时更新, 保障了 SDN 吞吐量, 从而减小了控制器失联对通信的影响, 增强了 SDN 的稳定性和可靠性。

关键词: OVS; SDN; OpenFlow; 软件交换机

中图分类号: TP393.4

文献标识码: A

doi: 10.11959/j.issn.1000-0801.2017094

Programming extension technology of open vSwitch

YANG Fan, YAN Siyu, HUANG Tao

Beijing University of Posts and Telecommunications, Beijing 100876, China

Abstract: As an open source SDN software switch, open vSwitch (OVS) has been widespread concerned. But in complex and changeable network environment, data layer loses control of controller and flow table entry learning aging appears easily and the performance of SDN is seriously hampered. To solve the problems, the programming extension technology architecture of OVS was designed, including link state processing, fault perception, two three-tier self-learning forwarding and other expansion of core functions, so that OVS could recover automatic and rapidly, flow table entries could update in time, throughput of SDN was guaranteed, thereby the influence of controller loss control for communication was reduced, SDN stability and reliability was enhanced.

Key words: open vSwitch, software defined networking, OpenFlow, software switch

1 引言

Open vSwitch (OVS)^[1]是开源的 SDN 数据平面软件交换设备^[2,3], 是 SDN 的重要组成部分。随着通用处理器性能的不不断提升, SDN 软件交换设备性能逐步增强^[4,5], 基于 OVS 构建的 SDN 越来越受到业界的追捧和青睐。但由于

SDN 的集中式架构, OVS 很容易受到诸如控制器宕机或切换、控制器应用故障、控制层链路拥塞或断开、人为控制层配置错误等故障问题影响^[6-8], 造成通信中断, 给 SDN 的安全性和可靠性带来了极大的挑战^[9]。本文设计了一种 OVS 的编程扩展技术架构, 建立完全兼容 OpenFlow 通道^[10]的 OVS 自学习体制, 可以提高 SDN 恢复



时间,保障故障场景下二层、三层正常通信,提升 SDN 的稳定性和可扩展性^[11]。

2 技术架构

2.1 总体架构设计

OVS 架构可分为用户层、内核层、配置管理层 3 个主要部分^[1,10]。其中,内核层包含了流表和一个或多个数据通路模块,主要负责对数据分组进行操作;用户层中运行着 OVS 的守护进程和数据库,它们是 OVS 的核心功能模块;配置管理层主要用于和守护进程、数据库之间进行交互操作以及 OVS 的安装配置和部署。

目前 OVS 使用双通道来实现故障场景下的转发性能^[1,10],即允许进入 OVS 的数据分组先进入 OpenFlow 通道,然后通过 normal 或 flood 保留端口再进入传统通道,按照传统协议完成二层转发功能,以此来保障控制器失效后数据分组的转发。为了改善 normal 模式目前无法支持三层路由功能、使用双通道进入故障场景状态时间较长、安全性能不足且使用传统通道与 SDN 思想相违背等问题,设计实现 OVS 的编程扩展技术架构,具有以下特点。

(1) 完全兼容 OpenFlow

传统 OVS 使用双通道处理数据分组,兼容性较差,基于 OVS 编程扩展技术实现新的转发体系,在 OpenFlow 基础上进行设计,故障情况下,数据分组经由 OpenFlow 通道处理,与 OpenFlow 完全兼容,保留 OpenFlow 与 SDN 技术灵活性、扩展性等优点。

(2) 支持三层路由功能

目前使用 normal 模式处理数据分组无法支持三层路由功能,仅能实现二层转发,基于 OVS 扩展技术的新的转发体系设计使用 RIP (routing information protocol, 路由信息协议) 以实现三层路由功能。

(3) 减短故障恢复时间

在传统 OVS 解决方案中,不能及时检测到控制器连接失效,故障恢复时间较长,严重影响 SDN

控制器失效场景下底层网络通信的性能。OVS 编程扩展技术通过设计连接故障检测机制,快速感知故障的发生,并且通过状态匹配进行故障的平滑切换,缩短故障恢复时间。

(4) 增强安全管理

使用传统 OVS 通道转发的数据分组不受控制,无法对数据分组进行安全检测和控制流向,存在安全性问题。研究设计的 OVS 编程扩展技术可通过连接状态匹配,完成断开状态下对数据分组的控制策略,增强控制范围和安全性。

(5) 及时适应网络拓扑变化

OVS 编程扩展技术设计新的转发体系,其转发机制完全受 OVS 中流表的控制,控制转发的流表能够及时学习以及老化,使 OVS 能迅速适应网络拓扑的变化,避免造成网络拓扑变化导致数据分组错误转发的情况。

图 1 是设计的 OVS 编程扩展技术架构,通过开发 OpenFlow 消息处理模块、状态的跟踪/分发/添加模块、应用感知服务模块、vPort 数据分组处理模块、数据通道流表匹配模块等实现故障检测机制、二层自学习、三层路由转发和 OVS 灾备流表的设计,其扩展功能主要包括状态检测扩展、转发扩展和信息更新扩展。

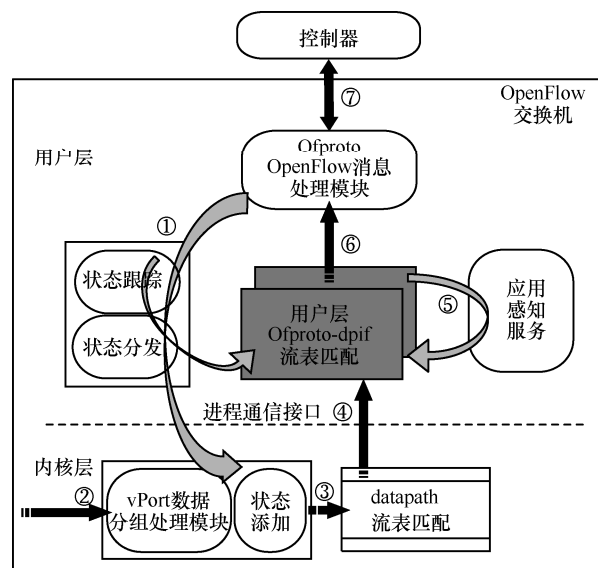


图 1 OVS 编程扩展技术架构

2.2 状态检测扩展

通过状态检测扩展，OVS 交换机可以实时动态地监测控制器的可用性。控制器发生故障失效后，扩展后的 OVS 交换机可以依赖连接状态检测机制使得 SDN 平滑进入灾备模式，保持数据平面通信，实现 SDN 的稳定性和可靠性。状态检测机制通过探测消息来动态检测控制器的活动状态，并且为了保证探测的及时性，会通过往返时延动态调整探针的发送间隔。对于 SDN 的可靠性保障，连接状态主要分为连接正常和连接中断两种状态。当连接状态正常时，OVS 交换机由 SDN 控制器正常管控，完成网络的路由和通信；当连接状态处于中继状态时，网关设备进入灾备模式。OVS 编程扩展技术中的状态检测机通过拓展 OpenFlow 协议匹配字段，设计连接状态检测和匹配功能，当数据分组到达 OVS 交换机时，会得到实时的控制器与交换设备间的连接状态，以此通过匹配流表状态完成相应动作。

2.3 转发扩展

转发扩展包含两种情况：二层转发扩展和三层转发扩展。根据实际故障场景和组网的差异性，在 SDN 中会出现：一种是基本的二层转发，数据分组不需要修改 MAC 地址直接进行匹配转发即可；另一种是需要不同网段的路由转发，对数据分组头进行地址修改。

二层转发扩展：通过设计的状态匹配自学习模块即可实现。对于进入此故障域的数据分组，

匹配自学习表项，然后跳转到流表进行匹配转发，如果匹配失败，则只需要通过简单的泛洪机制转发至网络域内所有设备即可。

三层转发扩展：除结合二层转发扩展外，需要在转发前对原 MAC 地址和目的 MAC 地址进行修改。通过拓展 OpenFlow 的 ARP 机制来探测获取目的地址，然后修改数据分组的目的地址进行转发。此外，对于三层转发扩展的自学习功能，学习到的新表项指令需要含有二层 MAC 地址修改动作。

2.4 信息更新扩展

信息更新扩展用于故障恢复后帮助控制器获取最新的网络信息，主要包括当前拓扑信息和最新流表信息。控制器故障恢复重新接入后，会通过向 OVS 交换机发送流表请求消息，重新读取交换机和网关的最新流表信息，完成控制器在故障恢复后的信息更新工作。

故障恢复控制器重接入 SDN 后，网络拓扑可能已经发生变化，某些交换机中原有的表项信息可能已经过期失效。因此，控制器重新接入网络后，会通过获取的最新拓扑信息和旧的拓扑信息，进行拓扑节点连接对比，然后对数据库中获取到的最新流表信息进行拓扑的有效性检测，将错误的流表从数据库中删除，并下发指令从相应交换机中删除或更新。如图 2 所示，控制器重新连接后，网络拓扑已经发生变化，控制器保存的原有交换机 1 的流表信息会

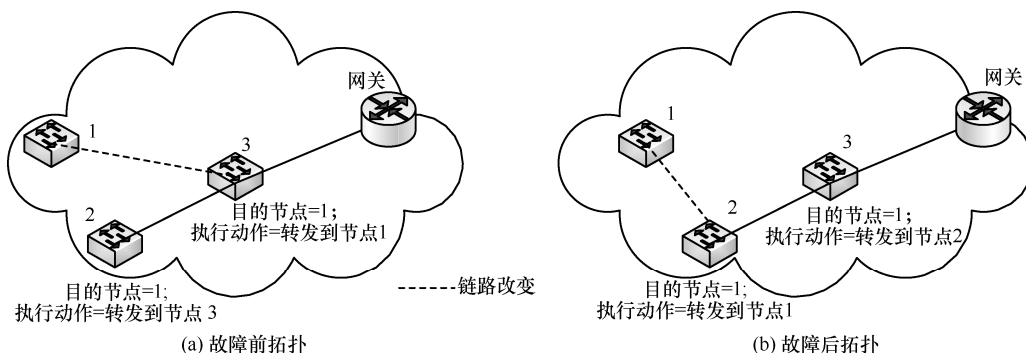


图 2 信息更新扩展



出现错误, 通过信息更新扩展, 控制器下发指令进行信息更新与删除。

3 原型实现

OVS 是在开源的 Apache 2.0 许可下的产品级质量的多层虚拟交换机, 支持标准的管理接口和协议, 目前广泛应用于 SDN 中^[12-14]。OVS 的编程扩展技术基于 OVS 进行设计实现, 兼容 OpenFlow 通道, 主要设计开发连接状态检测机制、自学习机制、灾备机制、三层路由转发等功能模块, 实现保障故障场景下 SDN 的通信吞吐能力。

3.1 连接状态检测机制实现

连接状态检测机制通过扩展 OpenFlow 的无状态字段, 向流表匹配域中增加状态匹配字段, 主要通过控制器连接状态检测和状态同步与封装来完成。连接状态检测机制如图 3 所示。

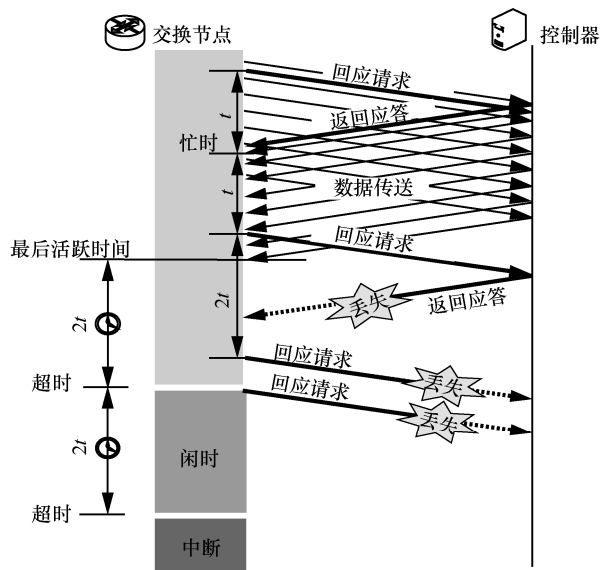


图 3 连接状态检测机制

(1) 控制器连接状态检测

根据状态机和数据分组传输速率跟踪机制完成状态的实时监测, 控制器连接状态码设计为 4 种: 中断 (disconnected)、连接 (connected)、闲时 (idle)、忙时 (active)。对于连接与中断状态, 控

制器连接状态检测机制主要基于有限状态机实现。扩展的有限状态机包括 active、idle 和 disconnected 等状态。当 OVS 与 SDN 控制器间建立起 OpenFlow 连接, 即进入 active 状态, 之后会根据是否接收到来自 SDN 控制器 OpenFlow 通道消息, 进行 active、idle 状态间的互相转换。当收到控制器断开的消息时, 则直接跳转到 disconnected 状态。

(2) 状态同步与封装

首先, 将 SDN 控制器与扩展的 OVS 设备间控制链路的连接状态值通过通用进程通信接口高效传递给 OVS 的数据通路中。然后, 当数据分组进入扩展的 OVS 设备时, 会将连接状态值作为数据分组的一个元数据, 连同端口号等其他数据分组元数据字段共同构造带有连接状态的匹配查找 key, 完成扩展的 OVS 设备流表的快速匹配。

3.2 自学习机制实现

自学习机制通过拓展 OpenFlow 的动作, 保障故障场景下 SDN 数据层面的转发能力和效率。

自学习功能嵌入 OVS 数据层面, 数据分组有时候不需要发送给控制器处理而直接在数据层面处理。为了符合 SDN 思想和增加灵活性, 控制器利用 OpenFlow API 对自学习等功能进行灵活调用, 在数据层面实现部分服务。此外, 自学习模块产生的流表项和原有的 OpenFlow 流表规则相同, 可以无差异地插入原有的 OpenFlow 流表, 保持 OpenFlow 和 SDN 的原有特性。

具体实现流程如图 4 所示。首先, 将执行自学习行为的数据分组进行分组头解析, 提取数据分组源 MAC 地址和入端口字段; 其次, 根据这些信息构造学习新 OpenFlow 流表项, 新的流表项的地址匹配字段就是数据分组的源 MAC 地址, 行为的转发出口就是数据分组的入端口; 最后, 检查流表中是否存在相似流表项, 存在则对旧的流表项进行更新, 不存在则插入新学习到的流表项。

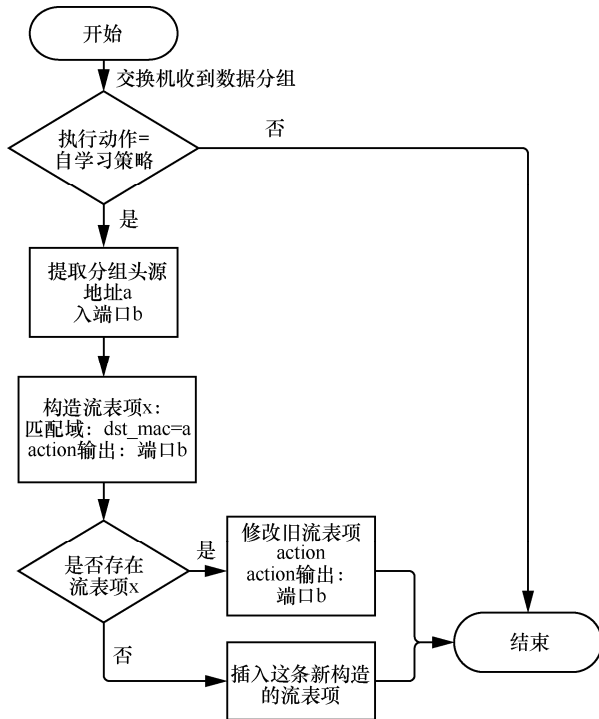


图4 自学习实现流程

3.3 灾备机制实现

灾备机制基于连接状态检测机制和自学自机制来实现, 设计的灾备方案流程如下。

(1) 控制器运行灾备 App 向扩展 OVS 下发启动灾备机制。灾备机制主要由 3 张流表项组成, 具体见表 1~表 3。

表1 状态流表项

优先级	流表
0	连接状态=2, 执行动作=自学习策略, 跳转到表 3
1	连接状态=1, 执行动作=跳转到表 2

表2 连接流表项

优先级	流表
0	连接状态=1, 执行动作=转向控制器
1	普通流表项: 连接状态=1

表3 灾备流表项

优先级	流表
0	匹配模式: 任意, 执行动作=泛洪
1	自学习流表项: 超时时长=10
2	安全流表项: 目的 IP 地址=x, 执行动作=丢弃

其中, 表 1~表 3 分别为状态流表项、连接流表项和灾备流表项。状态流用于实现 SDN 控制器与 OVS 在不同连接状态下的不同执行策略; 连接流用于实现 OVS 与 SDN 控制器在正常连接状态下数据分组的转发动作。灾备流用于实现出现故障场景时的自学习流表项的执行。匹配域连接状态为 1 时表示控制器连接正常, 为 2 则断开。

(2) 当数据分组进入 OpenFlow 通道后, 首先进入表 1 进行匹配, 然后根据 OVS 与 SDN 控制器当前的连接状态, 对数据分组进行相应的匹配处理。

(3) 当 OVS 与控制器连接正常时, 数据分组匹配表 1 的第二条流表项, 即跳转到表 2。表 2 是连接状态下控制器下发给 OVS 的流表项, 实现在连接状态下的数据分组转发策略。

(4) 当 OVS 与控制器连接断开时, 数据分组匹配表 1 的第一条流表项。需执行两个动作: 第一个动作是执行自学习策略, 把学习到的表项插入表 3 中; 随后执行第二个动作, 即跳转到表 3。

(5) 在表 3 中完成故障场景下的灾备模式, 匹配二层或三层的自学习流表项, 完成 OVS 与 SDN 控制器断开后的正常通信。

3.4 三层路由转发

区别于传统 OVS 仅支持二层网络, 直接使用 MAC 地址进行通信, 扩展的 OVS 交换机引入支持三层网络, 可以实现不同 IP 网段之间的通信, 具备全局路由表能力, 指导不同 IP 网段之间主机的通信。为了使扩展的 OVS 交换机获得全局的路由表, 结合传统网络三层路由转发的体系结构, 基于已扩展的处理模块添加使用 RIP 内部网关协议进行路由表的构建。

(1) 自主进行 RIP 数据分组的发送和终止发送防止因网络情况不好导致误判控制器连接失败的情况, 设置当连续检测到 10 次 connect_state=2 时, 表示与控制器连接已中断, 开始进入自主发送 RIP 数据分组状态。当检测机制检测到与控制器链路中断时, OVS 将立即进入自



主发送 RIP 数据分组状态, 开始进行 RIP 数据分组的封装和定时发送。通过相邻 OVS 间 RIP 数据分组的交换, OVS 将逐渐建立全局的路由表, 以实现不同网段间的数据转发。

(2) 解析 RIP 数据分组并由此生成路由表

将 RIP 数据分组的解析扩展 OpenFlow 的动作, 在拥有灾备流表的情况下, 扩展的 OVS 能进行 RIP 数据分组的解析动作, 以逐渐建立全局路由情况, 并根据全局路由情况下发相应流表, 实现一次路由多次转发的能力 (即下发流表后, 数据分组只需匹配流表即可正确转发, 无需进行路由查询)。在数据层面可以直接调度三层自学习功能, 保障数据平面转发能力和效率。

(3) 支持 IP 地址到 MAC 地址的转换功能

即传统网络中的 ARP 功能, 以保障能正确通过 IP 地址查询出主机对应 MAC 地址进行转发。由于目前 OVS 无法支持主动发送 ARP 请求以及缓存 ARP, 为了实现三层的路由转发, 将 ARP 学习扩展为 OpenFlow 的动作, 与二层自学习类似, 在 OVS 收到 ARP 数据分组时, 进行相应 MAC 地址与 IP 地址对应关系的学习, 并将结果直接下发至流表中。

3.5 灾备流表设计

为了使交换机在与控制器失去连接的情况下仍然能够使用 OpenFlow 通道进行数据分组的处理, 数据分组的处理流程应该完全受到流表的控制, 所以为三层路由转发设计了一系列灾备流表。当交换机失去控制后, 数据分组到来后将会自动匹配灾备流表, 在一系列灾备流表的引导下进行正常的路由转发。

基于灾备流表, 扩展的 OVS 将数据分组分为 ARP 数据分组、RIP 数据分组以及 IP 数据分组 3 种, 并针对 3 种数据分组的不同情况进行细分, 依次进行相应的处理。采用多级流表的设计, 每级流表拥有不同的功能, 各级流表配合实现对数据分组的处理。使 OVS 能在与控制器失联的情况下,

能根据灾备流表处理所有可能收到的数据分组, 保障网络的通信能力。

4 实验与结果

为了加强对比性, 同时引入二层、三层转发实验。为了清晰方便地描述, 对于设计扩展的 OVS 设备方案分别在二层和三层实验中命名为 AL2 和 AL3, 此外, 将对比的传统 OVS 设备的实验分别命名为 BL2 和 BL3。转发实验在相同的时间段内随机地触发 SDN 控制链路故障, 测试多次 UDP 会话流在不同吞吐量下的切换时延, 图 5 为二层、三层转发实验切换时延概率分布。

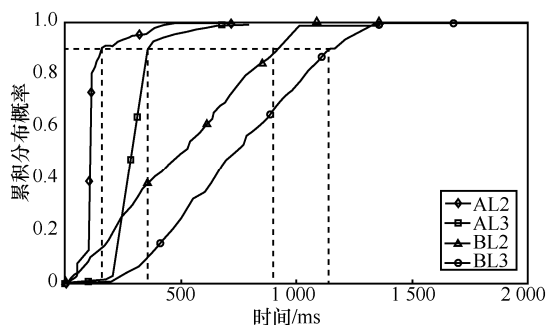


图 5 切换时延累积概率分布

由图 5 结果可知, 本文方案的切换时延整体小于 OVS 自身机制的切换时延。以 90% 位数为准, 本文方案的切换时延依然保持在 RTT 的 200 ms 级别。由于三层 AL3 转发需要一定的路由收敛时间, 因此切换时延相对二层 AL2 较长, 但依然处于 200 ms 级别左右。整体来说, 扩展后的 OVS 设备故障切换时延相比 OVS 自身的机制有了很大的性能提升。

在二层、三层转发实验中观测故障发生时不同 OVS 设备下的最大吞吐量变化, 实验分别在运行 5 s、12 s 进行故障场景触发, 并在 4 s 后故障场景恢复, 控制器重新连接网络。图 6 为二层、三层转发实验故障场景吞吐量对比。

从图 6 中可以看到, OVS 设备自身策略在吞吐量上不稳定, 且故障恢复时间长, 主要因为 OVS 设备自身方案需要定期维持表项, 其性能受控制

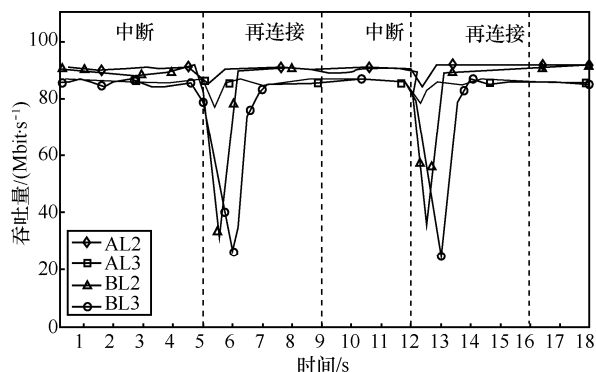


图6 故障场景吞吐量对比

链路影响较大,吞吐量出现波动。而扩展后的 OVS 设备通过检测、灾备等多种保障机制保障通信正常,使得吞吐量相对稳定,故障恢复时间较短。此外从图 6 中可以看出,当控制器重新连接后,扩展后的 OVS 设备能较快收敛并且保障通信正常,快速恢复吞吐量稳定状态,反映信息更新机制起到了故障恢复后的积极效果。全面保障了 SDN 通信的可靠性和稳定性。

5 结束语

SDN 应用场景中,网络控制器故障、拓扑突变等故障,容易使 OVS 设备出现失去 SDN 控制器控制、流表老化无法更新导致网络通信中断等问题。针对以上问题,本文设计了 OVS 的编程扩展技术架构,使传统 OVS 设备具备连接状态检测、表项自学习、灾备路由转发等能力,可实现 SDN 故障场景下不同网段的数据分组正确匹配转发和数据层面的自动化组网,提高了无控制器场景下的数据分组的控制,增强了数据分组转发效率,降低了网络恢复时间,保障了二、三层通信,提升了 SDN 安全性和可控性,并完全兼容 OpenFlow 通道,可拓展与应用于广泛的 SDN 场景。

参考文献:

[1] Open vSwitch[EB/OL]. (2015-11-05)[2017-01-13]. http://baike.baidu.com/link?url=eE6wD_8BWkt3MHsd3LP31KT2Li88ZVEkwmw2AmER85vvZHYErKXNIBVG1f3I8p5J5aQthlZJBQEdANWJ-wNITS_20G_pF-h1uTNsPqtGZm.

[2] ONF. OpenFlow-enabled mobile and wireless networks[EB/OL]. (2013-11-05) [2017-01-13]. http://wenku.baidu.com/link?url=KSCjGIIn79WW4sm9Abm906LqhjpXdVKKtOrDIBZ-9kqSVwTakdGQi-ONZRBYYQt93ZvIaYcn1xYohLexmgMtkLy-rHEIby3cIjScV_iAVre_.

[3] YAP K K, KOBAYASHI M, SHERWOOD R, et al. OpenRoads: empowering research in mobile networks[J]. ACM Sigcomm Computer Communication Review, 2010, 40(1): 125-126.

[4] KU I, LU Y, GERLA M, et al. Towards software-defined VANET: architecture and services[C]//2014 13th Annual Mediterranean Ad Hoc Networking Workshop (MED-HOC-NET), June 2-4, 2014, Ayia Napa, Cyprus. New Jersey: IEEE Press, 2014: 103-110.

[5] BASKETT P, SHANG Y, ZENG W, et al. SDNAN: software-defined networking in Ad Hoc networks of smart-phones[C]//2013 IEEE Consumer Communications and Networking Conference (CCNC), January 11-14, 2013, Las Vegas, NV, USA. New Jersey: IEEE Press, 2013: 861-862.

[6] KIM H, FEAMSTER N. Improving network management with software defined networking[J]. IEEE Communications Magazine, 2013, 51(2): 114-119.

[7] CHANDRASEKARAN B, BENSON T. Tolerating SDN application failures with LegoSDN[C]//The 13th ACM Workshop on Hot Topics in Networks, August 22, 2014, Chicago, Illinois, USA. New York: ACM Press, 2014: 22.

[8] MULEC G, VASIU R, FRIGURA-ILIASA F. Distributed flow controller for mobile Ad Hoc networks[C]//2013 IEEE 8th International Symposium on Applied Computational Intelligence and Informatics (SACI), May 23-25, 2013, Timisoara, Romania. New Jersey: IEEE Press, 2013: 143-146.

[9] MEKKY H, HAO F, MUKHERJEE S, et al. Application-aware data plane processing in SDN[C]//The Third Workshop on Hot Topics in Software Defined Networking, August 22, 2014, Chicago, Illinois, USA. New York: ACM Press, 2014: 13-18.

[10] PFAFF B, PETTIT J, KOPONEN T, et al. The design and implementation of open vSwitch[C]//12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15), May 4-6, 2015, Oakland, CA, USA. [S.l.:s.n.], 2015:117-130.

[11] 韦乐平. SDN 的战略思考[J]. 电信科学, 2015, 31(1): 7-12. WEI L P. Strategic thinking on SDN[J]. Telecommunications Science, 2015, 31(1): 7-12.

[12] 赵辉, 丁鸣, 程青松, 等. SDN 与 NFV 技术在云数据中心的规模应用[J]. 电信科学, 2016, 32(1): 144-151. ZHAO H, DING M, CHENG Q S, et al. Application of SDN and NFV technology in the cloud data center[J]. Telecommunications Science, 2016, 32(1): 144-151.

[13] 刘汉江, 欧亮, 陈文华, 等. 基于 SDN 的跨数据中心承载技术[J]. 电信科学, 2016, 32(3): 28-34. LIU H J, OU L, CHEN W H, et al. Bearing technology across the data center based on SDN[J]. Telecommunications Science, 2016, 32(3): 28-34.



- [14] 唐宏, 朱华虹, 曹维华, 等. 基于 SDN 的大型 IP 网络 BGP 路由优化方案[J]. 电信科学, 2016, 32(3): 14-19.

TANG H, ZHU H H, CAO W H, et al. Route optimization method for BGP based on SDN in large-scale IP network[J]. Telecommunications Science, 2016, 32(3): 14-19.

[作者简介]



杨帆 (1981-), 男, 北京邮电大学讲师, 主要研究方向为软件定义网络、高性能路由交换技术。



晏思宇 (1991-), 男, 北京邮电大学硕士生, 主要研究方向为软件定义网络、高性能路由交换技术。



黄韬 (1980-), 男, 北京邮电大学教授, 主要研究方向为未来网络体系架构、软件定义网络、信息中心网络、网络试验平台。