基于开源 SDN 控制器的下一代金融云网络的研究与实践

中国银联电子支付研究院（电子商务与电子支付国家工程实验室）：

袁航、周雍恺、祖立军

中国银联信息总中心：任明、吴一娜

中国银行数据中心：许泓

一、行业发展历程与技术发展趋势

（一）行业发展历程

金融数据中心网络技术架构与行业安全、合规等特色性要求紧密结合，是金融数据中心显著区别与其他行业数据中心的关键领域，是金融数据中心建设中的核心。中国金融数据中心网络建设的历程与金融行业近三十年信息化过程密不可分，总结归纳金融数据中心网络的发展主要经历了三个阶段：

第一阶段：专网专用阶段。该阶段是采用特有的网络协议来对专用设备进行连通，如 IBM 的专有网络协议 SNA 对其大型机与中型机的支持；

第二阶段：基于 IP 协议，分层分区。在本阶段采用了更为开放通用的 IP 技术协议，不再受制于某个厂商，组网更加灵活。此外，本阶段中虽然各金融机构的网络架构跟随应用系统的发展而变化，但一般会出于应用分层及安全的要求，遵循“垂直分层、水平分区”的

理念。

第三阶段：大规模共享接入。技术上更为开放，网络虚拟化与SDN等技术开始在金融行业应用。在继承安全区域保护机制下，采用“总线型、模块化”架构，中国的金融数据中心网络结构趋于一致，并且普遍采用网络虚拟化共享接入的技术方案，在新的云计算环境下能够对网络的灵活、弹性等要求进行有效应对。

（二）技术发展趋势

近年来，随着金融数据中心云化的加速，金融云作为最新的基础设施形态开始被行业认同并接纳。但在金融云环境下，传统网络技术架构受到了挑战：一方面虚拟化思想的出现，颠覆了原有的数据中心网络模型，使得传统网络技术已不足以适配云环境下产生的新场景，如虚拟机的出现要求网络颗粒度从物理机细化到了虚拟机级别；另一方面面向互联网的金融创新业务的快速发展，也会对网络的性能、弹性等特性提出更高的要求。所以未来金融数据中心网络技术必将进行变革式的创新发展，我们认为未来发展趋势主要包括以下三点：

1、面向互联网新兴业务的敏捷网络，即未来金融云网络能够高效满足互联网方式下金融创新应用的多样化需求。一是网络资源的快速提供与开通，以支撑应用的快速投产；二是强化细粒度的网络策略管控能力，在应用需求的频繁变化的情况下，网络能够进行灵活地变更调整；三是网络可兼容多样化的资源类型接入，以融合网络方式实现虚拟机、容器、物理机不同资源的统一接入。

2、面向数据中心资源动态变化的弹性网络，即在大流量挑战下保证网络的平稳运行。一是金融云规模巨大，承载业务系统众多，要求网络必须具备足够的容量与健壮性，比如如何解决网络规模快速增长情况下存在广播风暴的风险；二、在营销活动等访问量暴增的情况下，网络能够根据应用重要性与链路情况实现对流量的智能调度，保证核心业务平稳运行；三则是在计算资源不充足的情况下，网络能够连通分布在不同物理位置的计算资源池，打破由于物理区域隔离所造成的资源容量限制。

3、面向数字化智能化运维模式的网络，即在网络运维压力暴增的情况下，能够做到先于业务发现网络问题。一方面，金融行业数据中心规模不断扩张，网络终端数量与网络模型复杂度都呈几何式增长，必须采用高效、出错率低自动化运维代替传统的手工方式；另一方面，在金融云的新常态下，网络运维模式需要形成闭环来提升自身价值，通过对流量数据的采集分析，实现对网络的问题预测、排障、优化，甚至做到对网络攻击的规避，提升整体网络的稳定性。

软件定义网络（SDN）技术通过分布式架构理念，将网络中数据平面与控制平面相分离，从而实现了网络流量的灵活控制，为核心网络及应用的创新提供了良好的平台，其与金融云网络发展趋势相契合，是实现金融云网络服务的有效支撑技术。

二、以金融云为载体的创新网络需求

（一）金融云对网络的创新需求

基于上述金融云网络的发展趋势，结合金融业务面向互联网的挑战，我们认为未来金融云网络需求可总结为高安全、高敏捷、高性能、高可用、高弹性与高可管理：

高安全，金融业务的特殊性对承载网络的第一要求即为保证数据的安全性，因此金融云网络必须具备能够抵御系统外部攻击，保证数据完备性与私密性的能力；

高敏捷，实现业务快速上线，面对应用的变化达到资源的按需变更，通过新技术应用打破因重安全而舍效率的困局，在云计算新环境下安全与高效并重；

高性能，面对秒杀等新业务场景等的极限服务能力，实现时延和带宽等关键指标的跨越式提升，同时注重资源的高效利用，用尽可能少的资源实现最大的性能服务。

高可用，网络架构持续稳定影响金融数据中心全局服务能力，网络架构需要基于稳定可靠的技术构建，使网络服务具备 7*24 小时业务连续性服务的能力；

高弹性，一是内部弹性强化，打破竖井式架构中网络区域成为限制资源共享的壁垒，实现网络资源池整合与灵活共享与隔离，二是外部弹性兼容，支持新老架构并存，从而使原有网络可以平滑过渡到新架构；

高可管理，一是实现管理的体系的简化，支持多品牌的融合管理，二是实现管理自动化与智能化，提供端到端的业务可视和故障快速定位、排查能力，使日常运维从大量人工维护的高工作量解放出来；

（二）金融私有云与行业云对网络需求的异同分析

虽然金融私有云与行业云本质上都承载金融业务，但是由于应用场景与服务模式上的不同，也使得金融私有云与行业云对网络的需求有所差异。在表 1 中，我们基于上面提出网络需求的 6 个维度，对金融私有云与行业云网络需求的异同进行分析。

表 1 金融私有云与行业云对网络需求的异同

	私有云	行业云
高安全性	保证金融数据不泄露，并具备操作可审计、可回溯能力。	
	1.主要是外部攻击防范。	1.行业云中承载不同机构的业务系统，除了外部攻击防范外，平台内部不同租户的数据安全隔离以及内部攻击防范也十分重要。
高敏捷性	做到网络根据业务需求进行灵活、快速变动。	
高性能	网络能够保证业务系统在访问量暴增的情况下仍可以做到快速响应。	

	1.对突发流量可采用错分互补的方式满足性能要求，可将流量调度到。	1.可实现针对不同用户之间的网络性能分配。
高可靠性	网络层面必须要保证 7*24 小时的业务连续服务能力； 具备金融特色的两地三中心高可用架构。	
	1.二至七层网络能力全部高可用实现，并且进行整体提供。	1.高可用服务分层化，二至三层网络高可用作为基础能力，四至七层网络能力高可用作为服务提供。
高弹性	网络应满足业务系统对资源量的弹性需求。	
	1.可支持新老平台的互相对接； 2.应具备接入到行业云的能力，应对暴涨的资源需求； 3.能够兼容物理机、虚拟机、容器的共同接入。	1.能够打破资源分区造成的共享壁垒，实现跨区域资源共用； 2.应支持私有云的接入； 3.能够兼容物理机、虚拟机的共同接入（不包括容器）。
高可管理	实现自动化管理、智能化运维，缓解超大规模	

	网络带来的运管压力。	
	1.面向应用颗粒度的网络运维。	1.具备面向租户的运维能力。

三、下一代金融云 SDN 网络的设计原则与架构规划

（一）网络设计原则

SDN 技术的应用颠覆式地改变了金融数据中心网络架构，因此基于对网络发展趋势与具体需求的分析，在云环境下构建新一代的 SDN 网络需进行针对性的设计。据此我们提出了以下三条设计原则：

1.根据不同网络边界分层构建网络资源池

从能力层面来看，网络作为一种基础设施资源，应构建统一的云网络资源池，打破传统网络竖井式架构，提升计算、存储资源调用的灵活性；从管理与安全层面看的话，因为不同网络区域能力不同，在数据中心网络中的角色不同，所以应根据对不同网络区域分别构建资源池。

2.网络能力全部服务化实现

面向服务理念，对每层网络功能以服务、标准 API 接口的形式对外提供，网络系统内部以服务的形式进行自组织，从而提升对外服务能力，简化外部调用网络能力的复杂性；

3.网络资源统一编排管理

数据中心内网络二/三层连通、四/七层功能的管理界面统一视图，不同网络资源池的管理采用二级管理编排方式，即底层适配不同网络资源池管理操作、上层异构协调编排。

（二）网络架构

根据上述网络设计原则，我们规划了金融数据中心的整体网络架构如下图 1 所示。

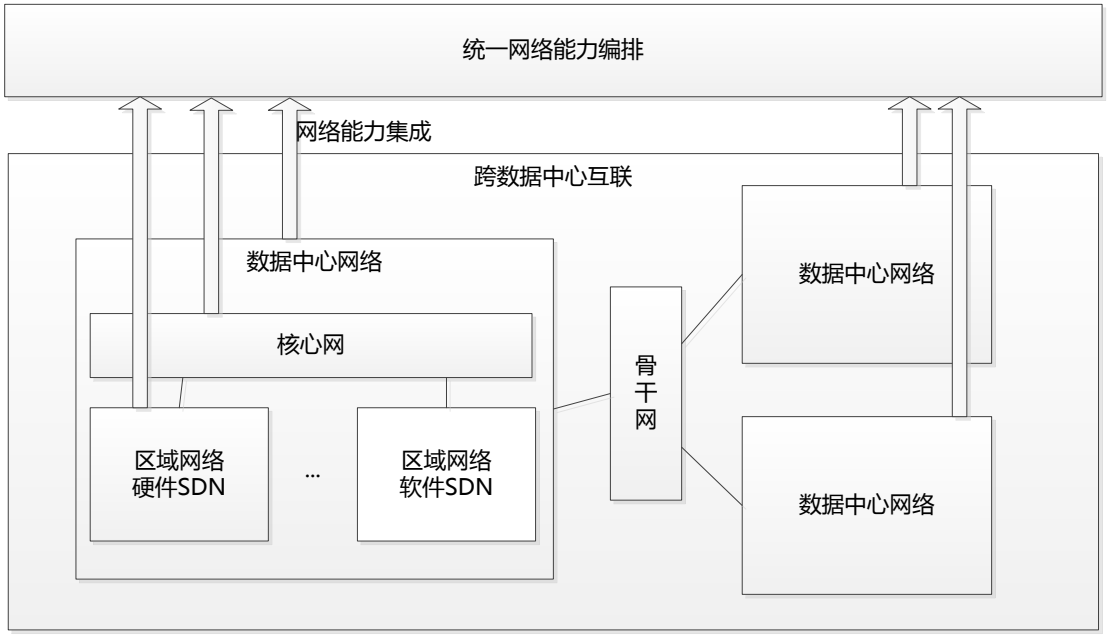

图 1 金融数据中心整体网络架构

首先，我们根据数据中心网络构成，将整体网络分成了三个部分，具体如下：

1.区域网络，也就是我们常说的一个云平台 Region 的网络，业务系统就运行在该区域内。其网络方案可分为硬件方案和软件方案。网络设备包括区域交换机、区域内控制器、负载均衡；

2.核心网，核心网就是连通各个区域的网络，主要设备包括核心

交换机；

3.数据中心网络，其实称为数据中心外联网络可能更准确一些，负责数据中心与外部网络的连通，其与外部骨干网连接，主要设备包括边缘路由器。

网络分区确定后，随后就根据各个分区的能力边界构建各自的网络资源池，并对各个资源池能力进行标准的 API 接口化实现。

最后，在顶层设计一个统一的网络能力编排系统，将各个资源池的能力通过 API 对接的方式进行上收，随后根据权限配置将不同网络区域的能力下放至相应的管理员与应用系统。

四、基于 ODL 开源控制器的数据中心内 SDN 网络方案研究与实现

SDN 方案分为硬件和软件两大类。硬件 SDN 是采用专用的硬件交换设备与控制器来实现相关的网络功能，控制器对硬件设备进行策略以及流表的下发，来实现网络相关的功能。它的优点是性能强，比较稳定，缺点是不灵活且组网成本很高。业界常见的硬件方案包括思科的 ACI，华为 AC、华三 VCF 等。

而在软件 SDN 的解决方案中，网络的功能是通过软件层面的 Linux 协议栈以及相关的虚拟交换机技术实现的。它的优点可以避免对硬件网络设备的过度依赖，同时降低了组网的成本，缺点是稳定性、性能和可扩展性不如硬件方案。常用的软件方案包括 Neutron+OpenvSwitch、OpenDayLight+OpenvSwitch 等。

下面对银联当前对 SDN 应用研究的现状进行介绍。

（一）银联 SDN 应用研究现状

中国银联自 2014 年启动软件定义网络（SDN）技术在金融云环境下的应用研究，长期跟踪 SDN 技术在国内外金融行业的研究进展，并积极推动 SDN 技术在银联生产环境的应用以及与银行金融机构的合作研究。目前，银联对 SDN 软硬方案的研究测试工作均已完成，两套方案全部落地生产。

银联私有生产云采用华为 SDN 整体硬件方案，银联生产托管云则采用 Neutron+OpenvSwitch 的软件 SDN 方案。当前实现了网络二/三层、负载均衡、防火墙等多网络资源服务，承载了近期银联与相关合作金融机构的关键应用，有效支撑了银联业务创新。

当前，银联结合当前生产现状与行业技术发展趋势，开展下一代金融 SDN 相关技术研究工作。目前主要针对金融数据中心区域内的软件 SDN 方案进行进一步研究优化，从而满足下一代金融云的网络要求。下图 2 中的红色范围即为本次研究工作的定位。

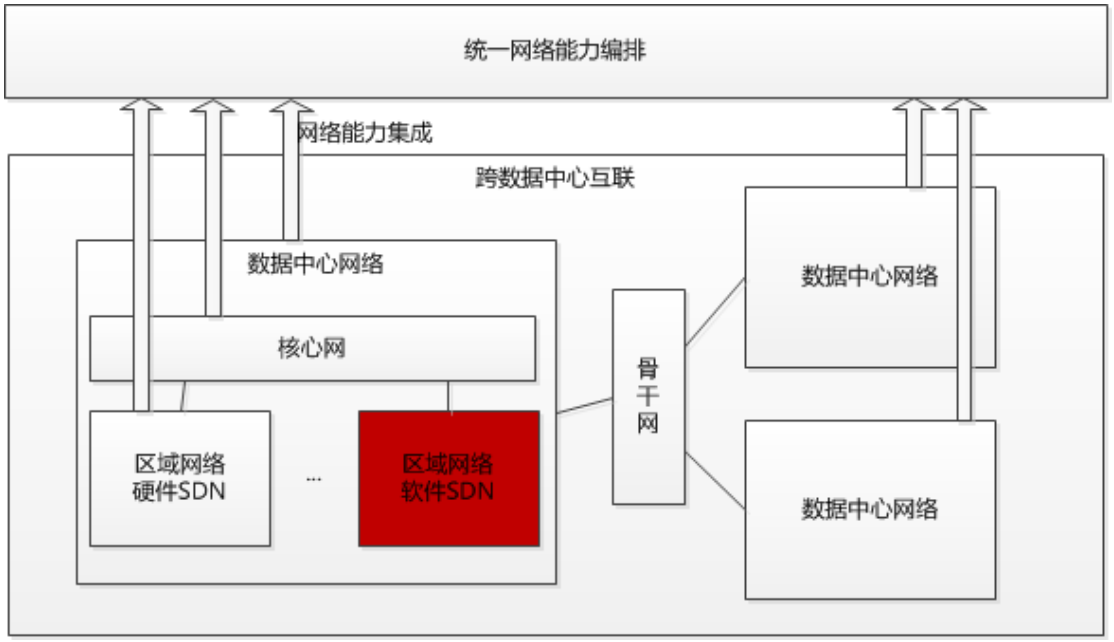

图 2 本次研究工作定位示意图

（二）下一代金融云网络软件 SDN 网络方案

1.方案设计与实现思路

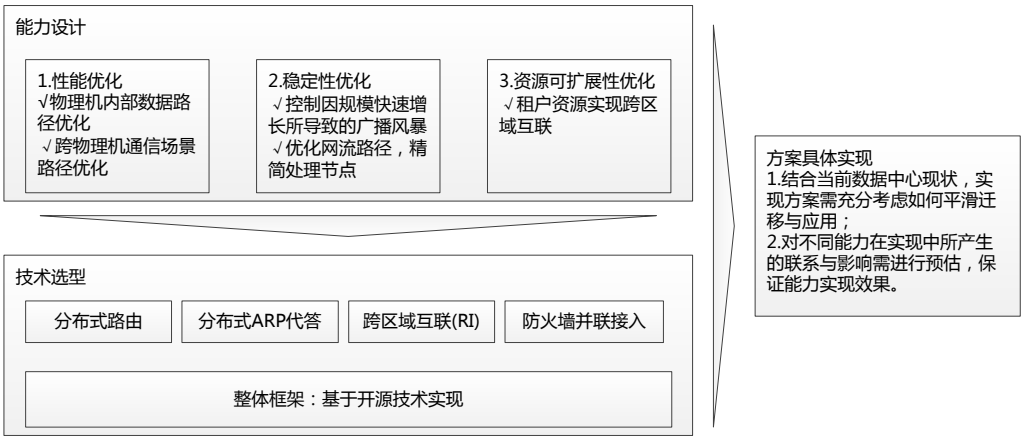

图 3 整体方案能力设计与实现思路图

（1）首先在对方案的能力设计上，我们结合了当前金融数据中心针对软件 SDN 方案的需求来进行规划，主要围绕三点：

✓ 首先性能是软件 SDN 方案较硬件方案来说比较明显的短板，

在性能上我们从两个层面上进行优化。**一是**要简化物理机内部的网流转发路径，如 Neutron 方案下物理机内部的网桥有三层，过多的网桥数量势必减缓对网络数据的处理速度，所以要简化；**二是**要优化物理机外部的网流转发路径，如 Neutron 方案下所有跨网段通信的流量全部要绕至专门的网络节点进行路由转发，给性能带来较大的影响。

- ✓ 然后是金融行业着重关注的稳定性上，我们也有相关的能力设计考虑。**一是**由于节点数量的规模快速增长所导致的广播风暴会对网络造成极大的损伤，因此本方案中将会针对该问题进行解决;**二是**优化网流路径,精简网流数据的处理节点，进一步减少网流转发中存在的风险点，并且打破集中式的网络瓶颈，采用分布式架构实现。
- ✓ 最后是为应对业务流量的突发式增长，方案在支撑资源的可扩展性上也有相关考虑。主要是网络能够打通跨区域的计算资源，并且做到在多租户环境下实现跨区域资源的互通。

（2）其次根据方案的能力设计，我们对方案的技术选型也进行了思考与规划，具体分为两个层次：

- ✓ 首先整体方案的技术框架我们依然选择采用基于开源技术实现。**一是**因为金融行业的一些特殊需求需要对相关能力进行定制化开发，而且足够快速和灵活，这就要求我们对方案具备自主可控的能力，采用商业软件是做不到的；**二是**如果从头进行开发将会消耗大量的时间经历，基于开源技术则会达

到快速实现的目的；

- ✓ 从具体的能力技术设计上，我们会进行分布式路由、分布式ARP、跨区域互联、防火墙并联接入等具体的技术方案以满足最初的能力设计。分布式路由打破了 Neutron 网络集中式节点处理方式，会对网络的性能、稳定性进行优化；分布式 ARP 将会很大程度上抑制网络中存在的广播报文，提高网络稳定性；跨区域互联通过对接 RI 系统实现；防火墙并联的实现也避免了防火墙成为网络瓶颈。具体设计思路式会在下文中详细阐述。

（3）最后根据方案的能力设计与技术选型，我们整理了两点具体实现的思路。**一是**能力的实现方案应充分考虑当前数据中心现状，应选取可以平滑迁移和应用的方案进行实现；**二是**不同能力在实现的过程中相互之间会有联系或影响，比如要实现高级的跨区域通信能力，就必须对底层的分布式路由、ARP 代答等能力进行修善。因此在能力实现中要对这种情况进行充分预估与判断，防止由于忽视相互之间的影响而导致能力不足或出现相关隐患。

2.整体技术框架

本次方案研究中，我们依然采用开源技术框架来进行实现。核心控制层采用开源控制器 OpenDayLight（下文简称 ODL），上层编排仍使用 OpenStack 的 Neutron，Neutron 与 ODL 之间使用 ML2 plugin 进行对接；底层数据层使用 OpenvSwitch（下文简称 OVS）进行网流转

发交换，并通过 OVSDB 与 Openflow 协议与控制器对接，其中 OVSDB 负责对 OVS 进行配置，Openflow 则负责实现所有的数据转发功能现。整体框架图如下。

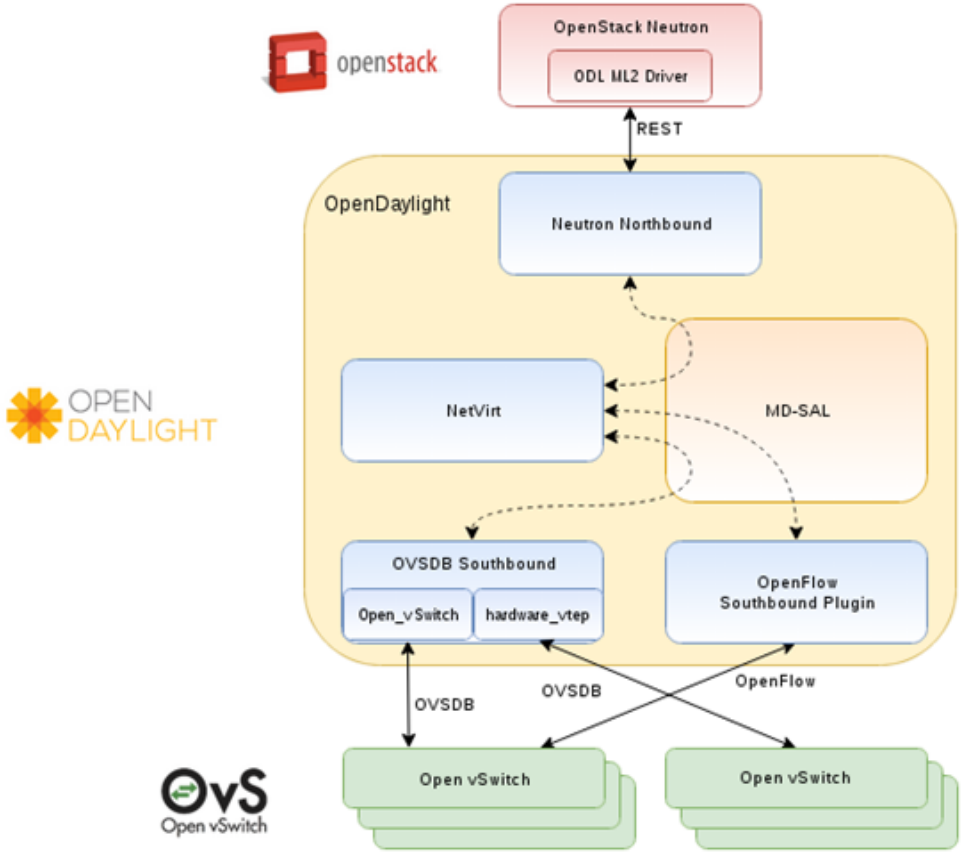

图 4 方案整体框架图

3.能力设计

（1）基于虚拟化的多租户支持

多租户虚拟化网络解决方案通过 Overlay 网络和 SDN 控制器的相互配合，可以使得逻辑网络与物理网络解耦、控制平面和转发平面分离，进而实现消除网络限制、虚机任意迁移、IP 地址灵活分配的目的，从而充分满足用户随时随地接入、业务快速上线、虚机迁移及策略自

动跟随的需求。

当前多租户已成为行业云的基本能力要求，但是在私有云中则会根据管理方式选择性部署。但是我们认为随着私有云规模的不断扩大，多租户也必将成为私有云的必需。一方面多租户技术允许网络资源重叠，能够缓解整体网络资源紧张的局面；另一方面多租户概念的引入将会使得不同应用之间的网络逻辑边界更加明显，在方便管理的同时也使得抽象的访问策略更加具象化，提升运维效率。

本方案中我们采用比较通用的 Vxlan 技术来实现多租户的能力。

（2）跨区域互联能力

传统交换网络稳定有余但灵活、高效不足。各网络分区之间计算、存储、网络、机房物理环境等资源均为独享模式，不同分区之间计算宿主机无法共享资源，虚拟机不允许在不同分区宿主机间漂移，计算资源利用率下降。为打破传统分区，本方案将会对基于多租户模式下的跨区域资源互联进行实现，提高资源利用率与应用部署灵活性。

在具体能力实现中，我们采用与银联自研的区域互联系统（以下简称 RI，RI 具体实现方式请见文章《中国银联与上海银行基于 SDN 的下一代金融云网络联合研究与应用实践》）进行对接的方案，在数据平面通过 Vlan 的方式与防火墙进行连通。

（3）分布式网络功能设计

云的本质是分布式计算的一种形式，采用虚拟化技术将集中的物理资源进行切割，并通过网络将资源分散给不同用户。因此，为了更好的契合云计算分布式的本质，避免集中式的网络功能成为云的瓶颈，在进行下一代金融云网络能力设计中，我们将分布式的网络功能作为必备能力。

常用的云网络功能包括网关、DHCP、ARP 响应、防火墙在本方案中，防火墙的能力通过硬件实现，所以在此不对其分布式实现进行讨论；DHCP 主要作用只是在虚拟机网络发生变化时，向虚拟机下发主机名和 IP 地址，应用场景少、涉及流量小，并非云网络瓶颈，对其进行分布式实现意义也较小。

而相反，在软件云网络方案中，网关与 ARP 响应两组功能也全为软件实现，属于网络基础能力，应用频繁。网关是三层通信的流量转发点，不同网络之间的流量通信都必须经过网关进行路由；而 ARP 响应则是获取目的 MAC 地址的唯一途径，是二层通信中不可或缺的流程与手段，同时也是区域内正常通信下广播流量的主要来源。因此，对网关与 ARP 响应进行分布式实现将会较大幅度地提升云网络的效率与稳定性。

综上所述，本方案会对网关与 ARP 响应能力进行分布式实现。

（4）防火墙并联方式接入

防火墙用于提供四到七层网络安全服务，实现逻辑区域之间的安

全隔离。

金融云网架构模型中，可将硬件防火墙资源进行池化部署，并按需进行调度，通过云控制平台实现防火墙统一管理。除此之外，金融数据中在防火墙接入形态上采用物理并联、逻辑串联的方式，在防火墙故障的情况下仍能保证业务的正常运行，提升了业务的稳定性。在本方案中也将实现该效果。

（5）最终能力实现效果

以上能力全部实现后，最终的效果图如下所示。

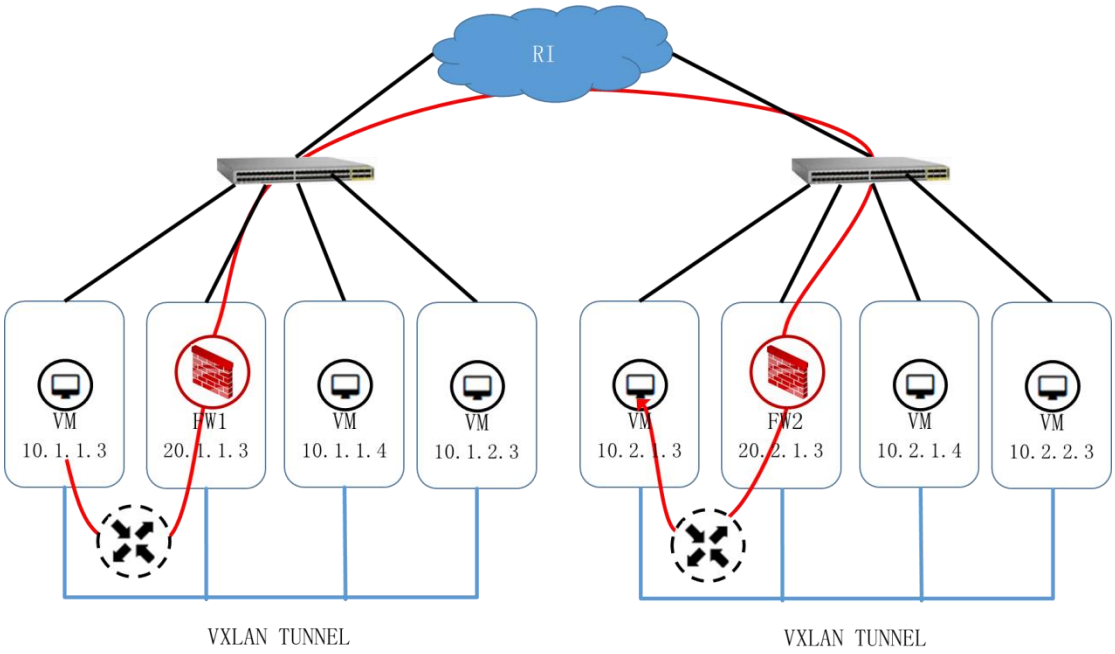

图 5 网络能力效果图

（三）基于 OpenFlow 流表的能力实现

从数据平面来看，本方案中所有的数据转发功能全部通过 Openflow 流表进行实现，即区域中所有的流量都由 OVS 依照 Openflow

流表来进行转发动作；而从控制平面上看，控制器只是根据方案预先制订的 Openflow 流表框架来实现到 OVS 的自动配置与下发的能力。所以，方案能力实现的关键仍在对 Openflow 流表的设计。

1.整体流表设计框架

OVS 的网络功能主要由网桥，端口与流表等实现。一个网桥中可以包含多级流表（Table0,Table1,Table2,…），流量在转发的过程中可以在不同的 Table 上进行跳转，以实现不同的功能；同时一个 Table 可以包含多条流表（flow entry），对流表可进行优先级的控制，但是只有一条流表会对进入 Table 的流量起作用。

原生 ODL 会在平台的每台物理机上创建 br-int、br-ex 两个 OVS 网桥，其中 br-ex 主要负责南北向通信，连接外部网络和 br-int 网桥，且只有一个 Table 0，功能比较简单；而 br-int 则负责虚拟机的接入，并实现大部分的网络能力，包含了 Table0、10、20 到 110 共 12 个 Table，功能较为复杂。各个 Table 的具体功能如下所示。

```
CLASSIFIER    Table0    "流量分类"
DIRECTOR    Table10    "Director"
ARP_RESPONDER    Table20    "分布式 ARP 应答"
INBOUND_NAT    Table30    "入站流量浮动 IP 流量 DNAT"
EGRESS_ACL    Table40    "出口访问控制"
LOAD_BALANCER    Table50    "分布式负载均衡"
ROUTING    Table60    "分布式路由"
```

```
L3_FORWARDING    Table70    "3 层转发"

L2_REWRITE    Table80    "2 层重写服务"

INGRESS_ACL    Table90    "入口访问控制"

OUTBOUND_NAT    Table100    "访问外部网络流量的 SNAT"

L2_FORWARDING    Table110    "二层转发"
```

为方便开发，本方案在流表设计中继续沿用原生 ODL 的流表框架与各个流表的功能设计。同时为了实现方案的设计能力，对相关 Table 进行能力补足与优化，具体修改的流表如下图 6 所示。

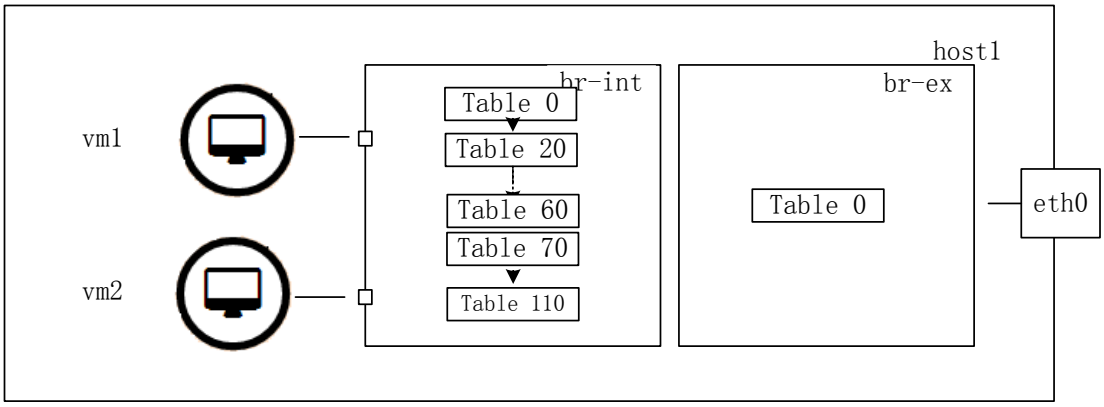

图 6 主要流表框架图

Table 0：租户在云网分区内部与外部之间的标签转换；

Table60：防火墙物理并联逻辑串联接入实现；

Table 20、70、110：支持去 Floating IP 的分布式网关实现。

下文中会对每项功能具体实现的技术方案、详细流表与代码架构进行详细说明。

2.能力优化与实现

能力实现 1：多租户环境下跨区域互联

做到多租户环境下的跨区域互联主要难点在与如何对存在于不同云网分区的租户流量进行标记与识别，从而保证通过核心交换网络后，云网分区可以正确将 IP 地址重用的多租户流量转发至正确的租户资源。

在对接 RI 后，所有跨区域通信流量在出区域防火墙后，即通过 RI 在核心交换区架起的隧道到达另一区域的防火墙。不同的租户在核心交换区对应不同的隧道，从而实现了不同区域不同租户的流量隔离。因此，我们只需关心跨区域互联时在区域内部的一些功能操作，而无需关心外部核心交换区域如何实现。具体如下图 7 所示。

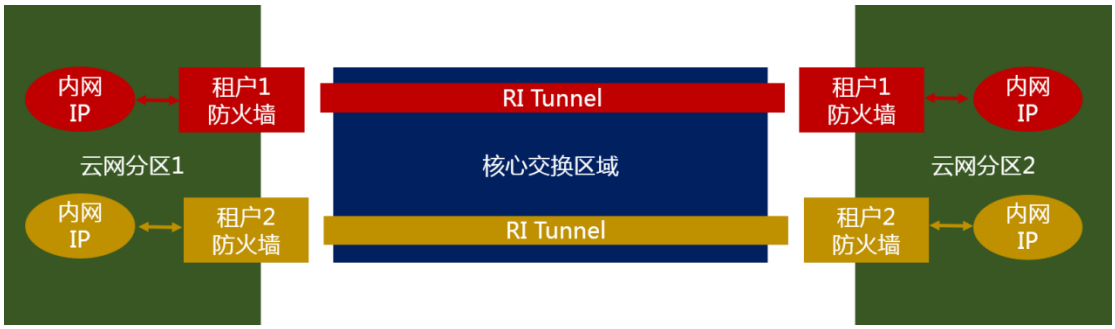

图 7 RI 跨区域通信示意图

在进行跨区域通信时我们考虑的问题有两个：一是跨区域的流量通过什么方式送到防火墙；二是防火墙接收到外部区域发来的跨区域访问流量的时候，如何将该流量发送到区域内。下面我们对这两个问题进行逐一分析。

问题一：跨区域流量通过什么方式发送到防火墙

在考虑该问题的时候，又会衍生出新的子问题：1.火墙支持的接入方式是什么，是否支持隧道接入；2.防火墙的物理连接方式是什么，并联还是串联。

先看子问题 1。在金融行业，对防火墙的性能和可用性有着比较高的要求，因此金融数据中心内部绝大部分仍使用硬件防火墙。而硬件防火墙往往不支持如 Vxlan、GRE 等一些隧道功能，所以一般还是采用 Vlan 的方式与防火墙连接。

子问题 2 提出了防火墙的物理连接方式。在能力设计的第四点已经提出，为保证业务运行的稳定性，降低网络故障瓶颈与影响范围，在金融数据中心防火墙采用并联方式接入。且为保证防火墙的性能，降低故障率，区域的外部网关不会建立在防火墙上。

既然防火墙已并联方式接入且外部网关不在防火墙上，那么区域内流量要发送到防火墙必须经过引流才能实现。具体引流方案我们会放在后面关于防火墙并联接入实现的内容中，在此不做赘述。

问题二：防火墙如何将流量发送到区域内

该问题也会产生两个子问题：

1. 防火墙如何将流量发送到区域的外部网关。该问题则是由于外部网关的分布式实现导致的，具体方案会在分布式路由实现中进行详细描述，在此不做赘述；

2. 同样是由于连接防火墙与区域内部网络方案的不同需要进行报文转换，只不过这次转换是有 Vlan 报文转换为 Vxlan 报文。

综上所述，要实现跨区域通信的影响面较广，分布式网关、防火

墙接入都会有所涉及。为使功能实现更加清晰，我们在这里只对 Vlan 到 Vxlan 的报文转换的实现方式进行描述。

流表设计

br-ex Table0：

```
table=0,priority=2048,in_port=3,dl_Vlan=310,nw_dst=10.2.1.0/24 actions= output:1
```

以上流表位于 br-ex Table0，接收到 Vlan 标签为 310、目的 IP 地址为 10.2.1.0/24 的报文并转发到 br-int。Vlan 310 是该租户在外部网络的 Vlan ID，output:1 则表示从 br-ex 的标号为 1 的端口发出，该端口即是 br-ex 与 br-int 的连接端口。

br-int Table0：

```
table=0,priority=2048,in_port=4,IP,dl_Vlan=310,nw_dst=10.2.1.0/24 actions=strIP_Vlan, set_field:0x28->tun_id,goto_table:20
```

当检测到其它区域发来的流量时，检测 Vlan 号和网段是否属于本区域并且对应关系一致，如果该流量的目的终端确实在本区域，卸载 Vlan 号，并进行 Vlan 到 Vxlan 的映射操作，并将该流量发送到下一流表中继续处理。

代码架构

添加接口：

文件：net-virt/src/main/java/org/opendaylight/netvirt/openstack/netvirt/api/ClassifierProvider.java

函数：programVlanToVxlanFlowEntry

文件：net-virt/src/main/java/org/opendaylight/netvirt/openstack/netvirt/api/L3ForwardingProvider.java

函数：programBrexFlowEntry

流表逻辑实现：

文件：net-virt/src/main/java/org/opendaylight/netvirt/openstack/netvirt/impl/NeutronL3Adapter.java

函数：handleNeutronRouterInterfaceEvent

流表下发：

文件：net-virt-providers/src/main/java/org/opendaylight/netvirt/openstack/netvirt/providers/openflow13/services/ClassifierService.java

函数：programVlanToVxlanFlowEntry

文件：net-virt-providers/src/main/java/org/opendaylight/netvirt/openstack/netvirt/providers/openflow13/services/L3ForwardingService.java

函数：programBrexFlowEntry

能力实现 2：防火墙物理并联逻辑串联接入实现

在上文的问题分析中，已经提到为什么防火墙要采用物理并联逻辑串联的接入方式，并且也提到通过引流方式进行实现。在本节中对实现具体方案和步骤进行详细描述。

首先，从引流的场景看，都有哪些南北向流量需要通过引导才能发送到防火墙。在本方案中，南北向流量可分为两种，一种是通过 Floating IP 被外部访问的流量，另一种是通过内网网段信息对外通信

的跨区域流量。具体如下图 8 所示。

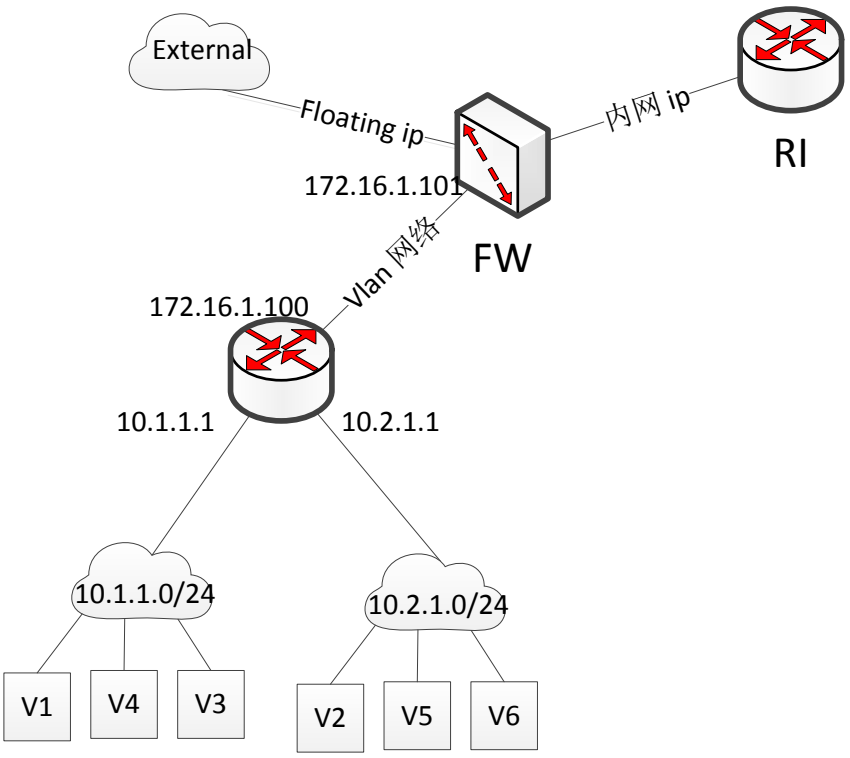

图 8 云网区域南北向通信示意图

对第一种南北向流量，因为带有 Floating IP 地址，因此其默认下一跳就会被送至外部接口网关，因此不需要引流就会被传送至防火墙并发出。对第二种南北向流量，其源 IP 和目的 IP 都为内网地址，其传送的防火墙属于跨网段通信，因此需要设置路由表对其进行引流，将去往另一个区域网段的下一跳设置在防火墙与路由器的接口上，从而实现了防火墙的引流功能。

流表实现

确定需要引流的流量后，我们就要进行引流功能的流表实现。在这里需要考虑两点：第一路由器与防火墙之间是 Vlan 模式的网络，因此流量在通过路由器的时候应打上 Vlan 标签；第二每个租户有各自的防火墙接口，接口的 MAC 地址要进行获取。最终流表实现如下：

table=60,priority=4096,IP,tun_id=0x1e,nw_src=10.1.1.0/24,nw_dst=10.2.1.0/24,actions=set_field:f8:4a:bf:5a:2b:ea ->eth_dst,dec_ttl,mod_Vlan_vid:211,output:3

以上流表是静态路由的实现，报文目标 IP 地址是另一个租户的网段时，将目标 mac 地址改成外部网络上租户防火墙接口的 mac 地址，根据源 IP 和 tun_id 确认租户，设置租户对应的的 Vlan id，将报文发出。

代码架构

添加接口：

文件：net-virt/src/main/java/org/opendaylight/netvirt/openstack/netvirt/api/RoutingProvider.java

函数：programStaticRoutesFlowEntry

文件：net-virt/src/main/java/org/opendaylight/netvirt/openstack/netvirt/api/GatewayMacResolver.java

函数：resolveMacAddressWithVlanTag

流表逻辑实现：

文件：net-virt/src/main/java/org/opendaylight/netvirt/openstack/netvirt/impl/NeutronL3Adapter.java

函数：handleNeutronRouterEvent

流表下发：

文件：net-virt-providers/src/main/java/org/opendaylight/netvirt/openstack/netvirt/providers/openflow13/services/RoutingService.java

函数：programStaticRoutesFlowEntry

文件：net-virt-providers/src/main/java/org/opendaylight/netvirt/openstack/netvirt/providers/openflow13/services/arp/ArpSender.java

函数：sendVlanTaggedArp

createVlanTaggedArpFrame

能力实现 3：支持去 Floating IP 的分布式路由

原生 ODL 实现方式

在能力设计中，我们提出采用分布式路由的实现方式。在 ODL 原生方案中，也对分布式路由进行了实现，具体实现方式如下图 9 所示。

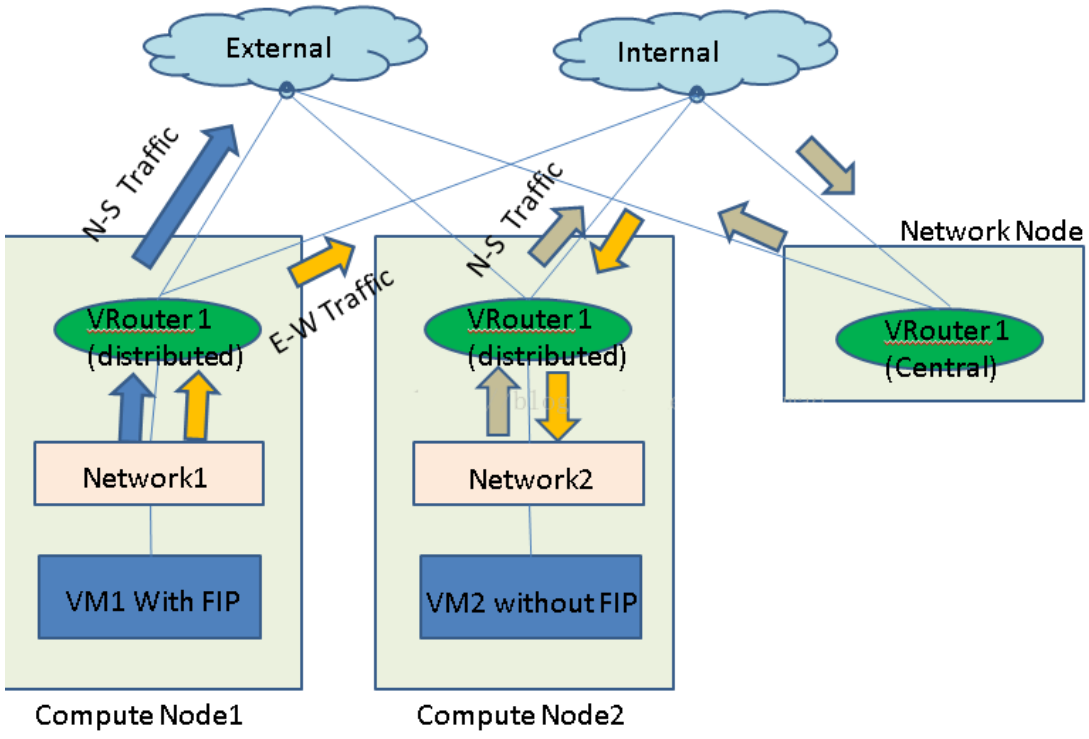

图 9 原生 ODL 分布式路由实现示意图

从图 9 可以看出，原生 ODL 的分布式路由机制则在每个节点上都使能一个路由器。对于东西向的流量， 流量会直接在计算节点之

间传递。对于南北向的流量，如果有 Floating IP，流量就直接走计算节点；但是对于没有 Floating IP 的流量，依然要通过集中式的网络节点发送。

在一般场景应用中，区域的南北向流量都要经过 NAT 处理（即使用 Floating IP）才能进行正常通信。如不进行 NAT 处理，区域内部的网段地址无法被外部网络识别，因此无法实现预期的数据转发。

但是在本方案中，由于存在跨区域通信的场景，为了识别租户信息，反而需要携带内部网络地址信息与其它区域进行通信。而该场景恰恰与上文中提到的无 Floating IP 进行南北向通信的方式相吻合。所以在原生的 ODL 设计中，该流量仍需要通过集中式的网络节点发送，这就与本方案的能力设计不符。

原理分析与问题提出

为了实现支持去 Floating IP 的分布式路由能力，我们需要对 ODL 原生分布式路由的设计方式进行进一步分析：为什么无 Floating IP 的南北向流量不能使用分布式网关方式实现？为了阐述起来更直观，我们通过下面的一个具体场景来寻找原因。

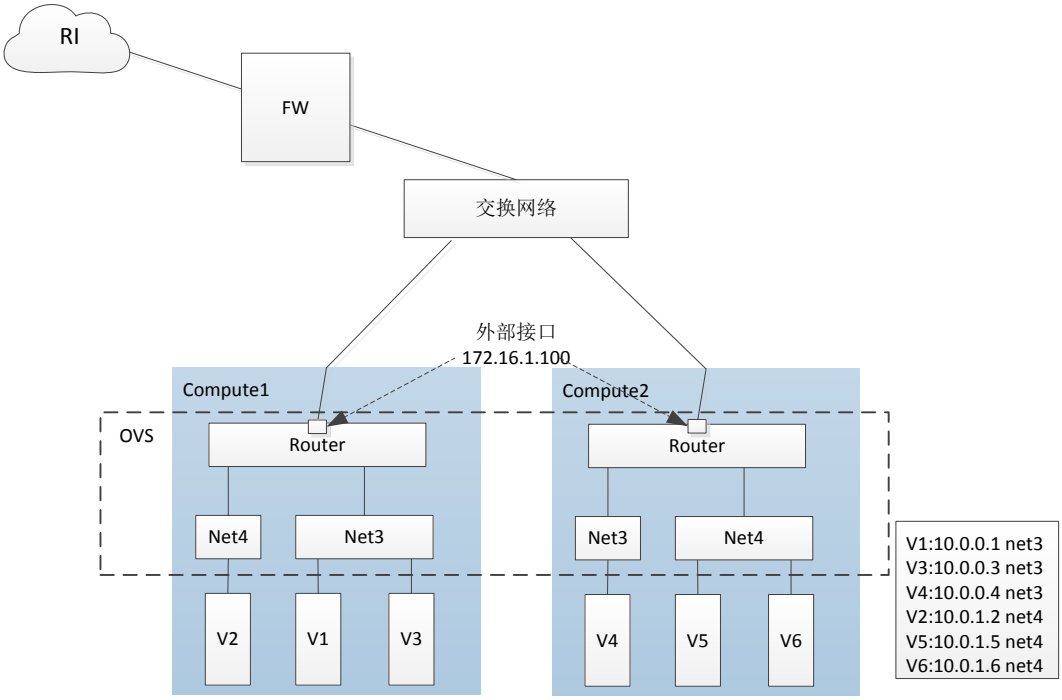

图 10 分布式网关物理结构图

上图是一张分布式网关的物理结构图，由图可看出每台物理节点的 OVS 都具备路由的功能。图中六台虚拟机同属同一租户且分布在两个网段中，租户与外部网络的接口地址为 172.16.1.100，同时为每台虚拟机都分配了相应的 Floating IP。

在该环境下，当虚拟机在与 external 网络通信时，流量到达 OVS 上时，OVS 中的相关表项会将数据包的源 IP 地址转换为唯一与该虚拟机对应的 Floating IP。如 v1 在与外部网络通信时，从 v1 中出来的数据包的源 IP 地址还是 v1 的 IP 地址，即 10.0.0.1，那么数据包到了 OVS 上之后，OVS 根据该数据包的目的 IP 地址判断出这是 v1 与外部网络通信的流量，这时 OVS 中就会有相应的流表对该数据包的源 IP 地址字段进行转换，即将 10.0.0.1 转换为 172.16.1.1，也就是 v1 的 Floating IP。那么对于外部网络来说，v1 的 IP 地址也就变为了

172.16.1.1。

因为 Floating IP 与虚拟机之间是一一对应的，所以外部网络在进行回包的时候，就可以直接通过 Floating IP 找到 v1 所在的位置，从直接而将数据送回至 v1。

如果 v1 没有 Floating IP ，虽然它主动向外部网络发送的数据是能够送至目的端的，但是目的端的返回包是无法送至 v1 的。因为 v1 的数据包是其内网地址 10.0.0.1 作为源 IP 地址的，而其内网地址是不为外部网络所认知的，这是存在的第一个问题。不过回到我们的跨区域通信场景中，由于对接 RI 系统，带有内网地址的数据可通过 RI 建立的隧道跨过核心交换区域到达另一个云网分区的防火墙上。所以第一个问题在跨区域通信的场景中不存在，我们继续往下分析。

当数据流到达云网分区的防火墙上时，我们需要解决上文中已经提及的问题：在分布式的网关场景下，流量如何通过防火墙发送到云网区域的外部接口上？

在原生的 ODL 方案中，区域的外部接口并没有实现接收外部网络数据的能力，所以我们需要对此功能进行实现。

实现了数据接收能力后，仍存在另外一个问题。如图 10 所示，云网分区的外部接口分布在区域内的每台物理节点上，而跨区域通信的数据包的目的虚拟机只存在于一台物理节点中。当防火墙向云网区域的外部接口发送数据包时，应该将数据包发送到哪一台物理节点上呢？换句换说就是如何定位目的虚拟机的具体位置。

综合分析下来，我们得知，要实现支持去 Floating IP 分布式网关

实现，就必须解决下面两个问题：

1. 实现区域外部接口对外来流量的数据接收能力；

2. 外部接口接收到数据后，能够将数据送达目的虚拟机。

流表实现

针对问题 1，我们通过设计外部接口的 ARP 响应的 openflow 流表，实现外部接口接收外来数据的能力。具体流表如下

```
table=20,priority=1024,arp,arp_tpa=172.16.1.100,arp_op=1 actions=move:NXM_OF_ETH_SRC[]->NXM_OF_ETH_DST[],set_field:f8:4a:bf:5a:2b:ea->eth_src,load:0x2->NXM_OF_ARP_OP[],move:NXM_NX_ARP_SHA[]->NXM_NX_ARP_THA[],move:NXM_OF_ARP_SPA[]->NXM_OF_ARP_TPA[],load:0xf84abf5a2bea->NXM_NX_ARP_SHA[],load:0xac100164->NXM_OF_ARP_SPA[],IN_PORT
```

上面的流表的主要作用就是为外部接口构造了一个 ARP 的响应包，在接收到对外部接口的 ARP 请求的时候，OVS 会根据该流表生成一个 ARP 响应包，发回给请求方。当请求方接收到该 ARP 响应报文后，就会将数据包发出，送至发出该响应报文的物理节点的 OVS 上。

为了保证路由的分布式架构，我们会在所有的物理节点上下发外部接口的 ARP 响应的流表。所以，在外部网络发出 ARP 请求后，所有物理节点都会对该请求进行响应。收到响应后，外部网络就会将数据包发出，发出后数据包就会按照物理交换机上的 mac 表进行转发，最终发送到平台中的某一个物理节点的 OVS 上。具体如下图 11 所示。

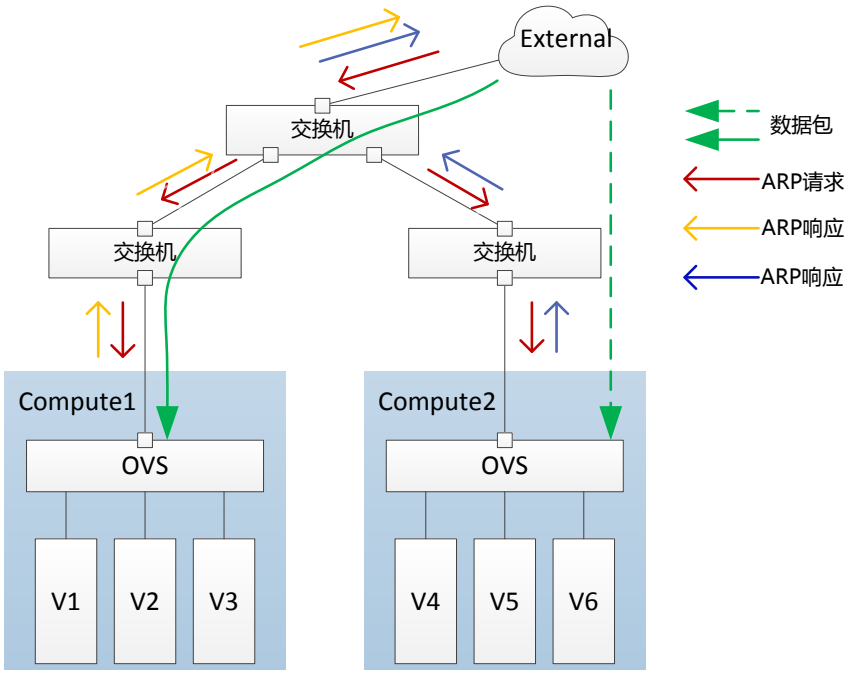

图 11 分布式网关 ARP 响应示意图

针对问题 2，当物理节点收到数据包后，会进一步对数据包进行分析。此时就会有两种情况：一是该虚拟机恰好在该物理节点中，此时就可以直接将数据包送到虚拟机上，相应流表如下；

table=70,priority=1024,IP,tun_id=0x5a,nw_dst=10.0.0.3 actions=set_field:fa:16:3e:99:df:47->eth_dst,goto_table:80（三层转发）

table=110, tun_id=0x5a,dl_dst=fa:16:3e:99:df:47 actions=output:23（二层转发到虚拟机，23 口与是虚拟机连接的 OVS 的端口）

还有一种情况就是虚拟机不在该物理节点中，那么这时候就要用隧道的方式，将数据包通过 Vxlan 隧道发送到虚拟机所处的物理节点，然后再送到虚拟机，如图 12 所示。相应流表如下：

table=70,priority=1024,IP,tun_id=0x5a,nw_dst=10.0.0.3 actions=set_field:fa:16:3e:99:df:47->eth_dst,goto_table:80（三层转发）

table=110, tun_id=0x5a,dl_dst=fa:16:3e:99:df:47 actions=output:3（通过 Tunnel 转发到对应物理机，后面的 output：3 代表从 3 口发出，3 口即为隧道的端口）

到对端物理机的 OVS 后：

table=110, tun_id=0x5a,dl_dst=fa:16:3e:99:df:47 actions=output:23（二层转发到虚拟机）

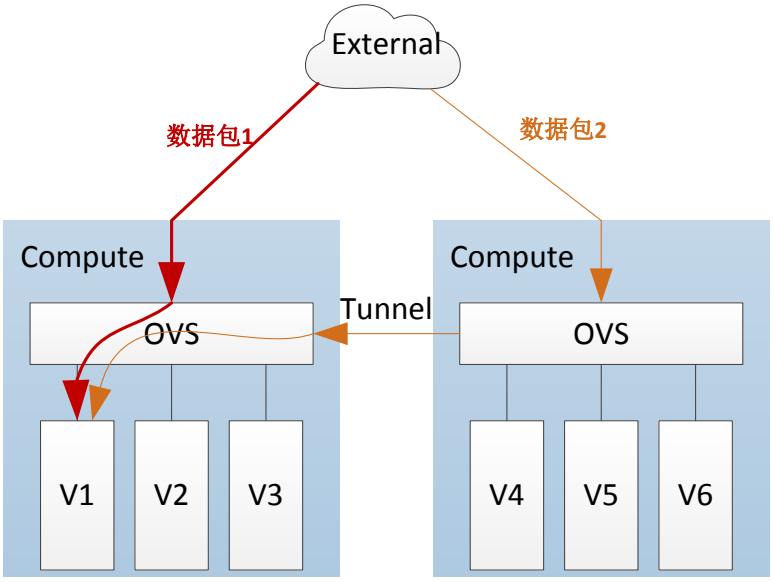

图 12 虚拟机定位示意图

整个流程统一起来，步骤如图 13 所示。

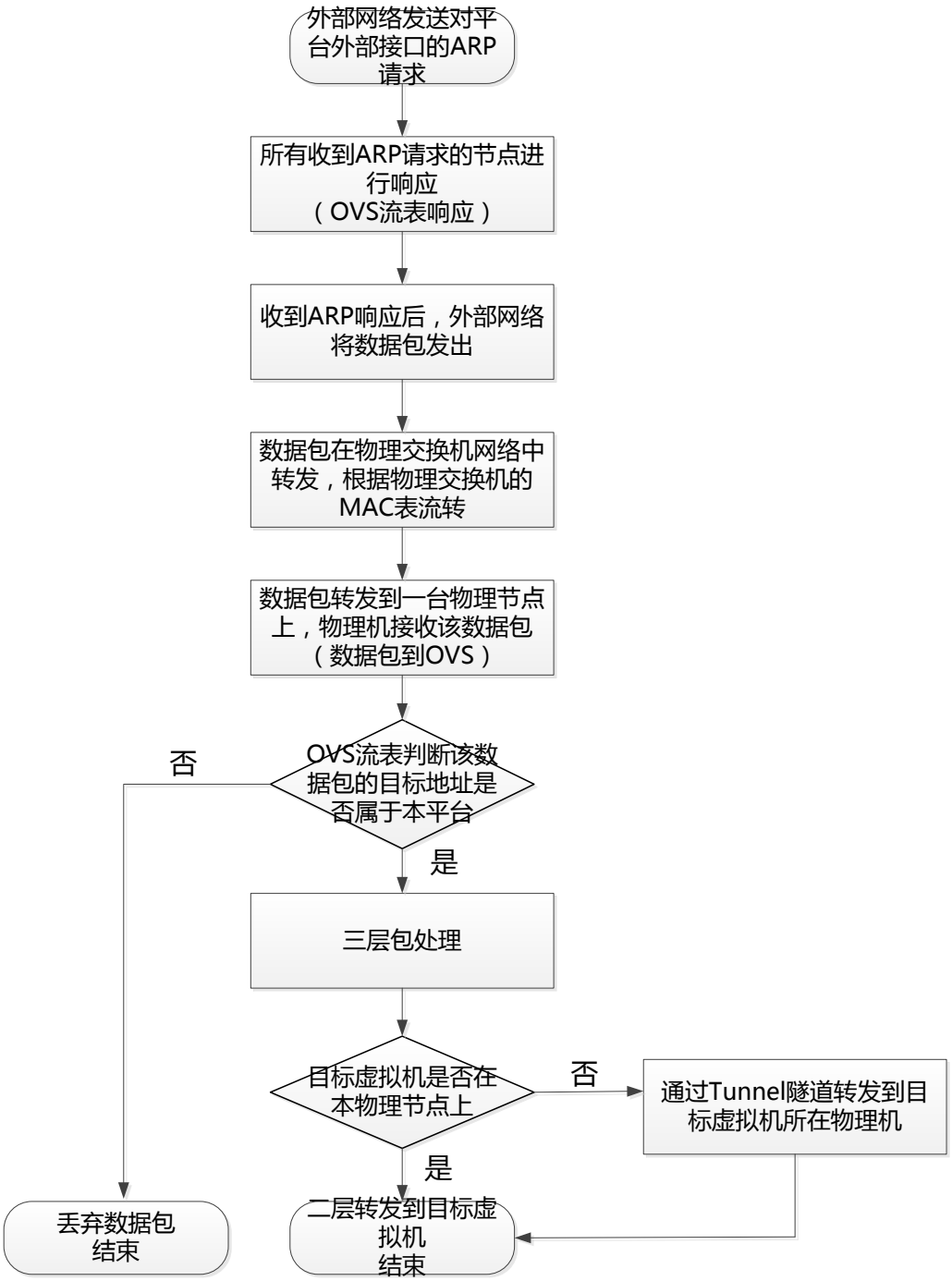

图 13 支持去 Floating IP 分布式网关数据处理流程图

代码架构

添加接口:

文件：net-virt/src/main/java/org/opendaylight/netvirt/openstack/netvirt/api/ArpProvider.java

函数：programPFWProviderArpEntry

文件：net-virt/src/main/java/org/opendaylight/netvirt/openstack/netvirt/PortHandler.java

函数：doNeutronPortCreated

流表逻辑实现

文件：net-virt/src/main/java/org/opendaylight/netvirt/openstack/netvirt/impl/DistributedArpService.java

函数：handleNeutronPortForArp

流表下发

文件：net-virt-providers/src/main/java/org/opendaylight/netvirt/openstack/netvirt/providers/openflow13/services/ArpResponderService.java

函数：programPFWProviderArpEntry

文件：net-virt-providers/src/main/java/org/opendaylight/netvirt/openstack/netvirt/providers/openflow13/services/L3ForwardingService.java

函数：programForwardingTableEntry

3.跨区域通信实现案例分析

下面我们结合一个跨区域通信的场景，举例分析跨区域虚拟机通信过程中经过的流表以及各个流表的功能。

如图 14 所示，两个虚拟机 VM1 和 VM2 分别在两个区域的两台主机上，VM1 向 VM2 发送 ICMP 请求。

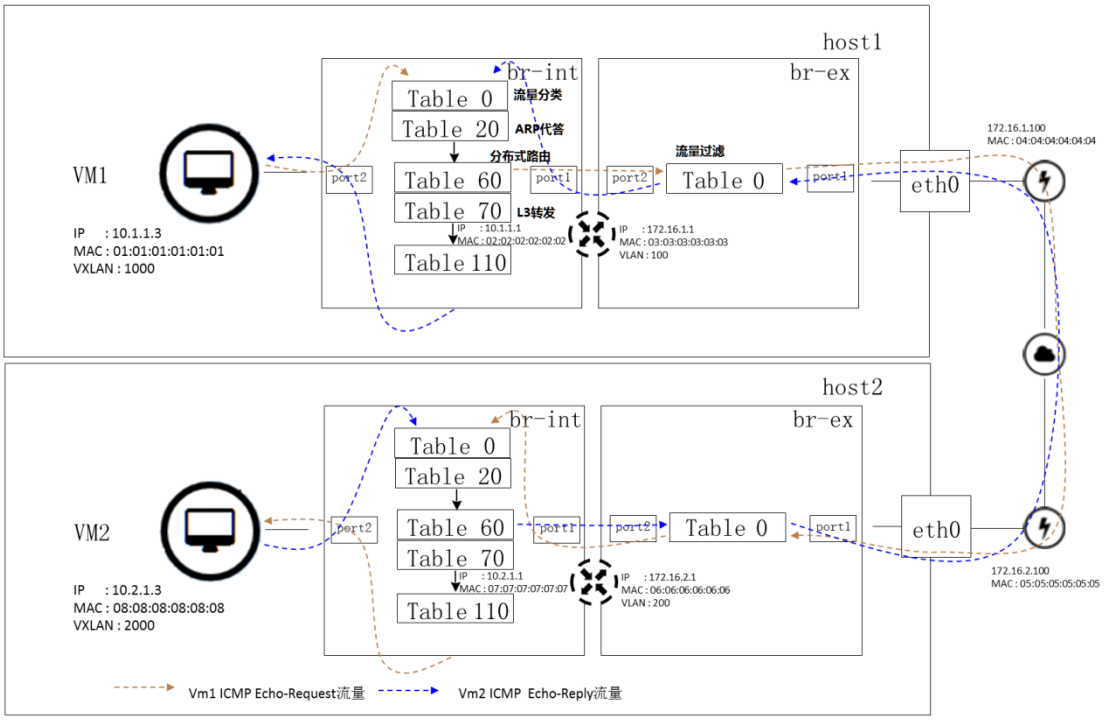

图 14 跨区域通信流表作用示意图

首先 VM1 会发送目标 IP 为网关 IP 10.1.1.1 的 ARP 请求广播包，由 OVS 获取并发送到 Table20 中进行处理，起作用的流表如下：

table=20,priority=1024,arp,arp_tpa=10.1.1.1,tun_id=0x3e8,arp_op=1 actions=move:NXM_OF_ETH_SRC[]->NXM_OF_ETH_DST[],set_field:02:02:02:02:02:02->eth_src,load:0x2->NXM_OF_ARP_OP[],move:NXM_NX_ARP_SHA[]->NXM_NX_ARP_THA[],move:NXM_OF_ARP_SPA[]->NXM_OF_ARP_TPA[],load:0x020202020202->NXM_NX_ARP_SHA[],load:0x0a010101->NXM_OF_ARP_SPA[],IN_PORT

Table20 匹配 tun_id 为 1000，目标 IP 为 10.1.1.1 的 ARP 请求包，将报文的目标 MAC 设为 10.1.1.3 的 MAC 地址，将报文的源 MAC 和 ARP_SHA 改为 10.1.1.1 的 MAC 地址（从 Neutron 获取），将报文类型改为 ARP 响应，并将响应报文原路送回到发送方。

VM1 拿到网关的 MAC 地址后，就会将 ICMP 报文发出，报文的

源 IP 是 10.1.1.3，目标 IP 是 10.2.1.3，并在整个传输过程中保持不变。报文发送到 OVS 后，首先起作用的报文是 Table60 的静态路由流表，本场景的静态路由流表会匹配 tun_id 为 1000，源地址是 10.1.1.0/24，目标地址是 10.2.0.0/16 的流量，将目标 MAC 地址修改为区域防火墙的 MAC 地址 04:04:04:04:04:04，给报文打上 VLAN TAG 100，并将报文转发至 br-ex。具体流表如下：

```
table=60, priority=4096,IP,tun_id=0x3e8,Vlan_tci=0x0000/0x1fff,nw_src=10.1.1.0/24,nw_dst=10.2.0.0/16 actions=push_Vlan:0x8100,set_field:4196->Vlan_vid,set_field:04:04:04:04:04:04 ->eth_dst,dec_ttl,output:1
```

br-ex 接收到报文进行流表匹配后，最终会匹配到优先级最低的 NORMAL 流表，NORMAL action 会以普通交换机的行为转发报文，也就是根据 MAC 地址和端口的对应关系转发，具体流表如下：

```
table=0, priority=0 actions=NORMAL
```

br-ex 的 NORMAL 流表会将报文通过 host1 的 eth0 发送到区域防火墙上，区域防火墙的默认网关在区域核心上，流量会通过路由到达交换核心并最终送到另一个区域的防火墙。防火墙上有区域内部网络的回程路由，由于目标地址是 10.2.1.3，会匹配到区域内 10.2.1.0/24 的回程路由，并送到下一跳 172.16.2.1。防火墙会发送目标 IP 为 172.16.2.1 的 ARP 请求广播报文，ARP 代答流表所在的主机 host2 会响应 ARP 请求，ARP 代答流表如下：

```
table=20,priority=1024,arp,dl_Vlan=200,arp_tpa=172.16.2.1,arp_op=1 actions=move:NXM_OF_ETH_SRC[]->NXM_OF_ETH_DST[],set_field:06:06:06:06:06:06->eth_src,loa
```

d:0x2->NXM_OF_ARP_OP[],move:NXM_NX_ARP_SHA[]->NXM_NX_ARP_THA[],move:NXM_OF_ARP_SPA[]->NXM_OF_ARP_TPA[],load:0x060606060606->NXM_NX_ARP_SHA[],load:0xac100201->NXM_OF_ARP_SPA[],IN_PORT

响应防火墙 ARP 请求后，防火墙会将 IP 报文发送到响应的主机 host2，通过 eth0 网卡进入 OVS，开始流表匹配。首先 br-ex 上的引流流表将外部区域访问本区域的流量转发到 br-int，匹配 VLAN ID 为 200，目标 IP 为 10.2.1.0/24 的报文，转发到 br-int。具体流表如下：

table=0, priority=2048,IP,dl_Vlan=200,nw_dst=10.2.1.0/24 actions=output:2

br-int 收到流量后对报文进行 VLAN 到 VXLAN 的转换，匹配 VLAN ID 为 200，目标 IP 为 10.2.1.0/24，从 br-ex 发来的 IP 报文，去掉 VLAN TAG，打上相应的 VXLAN ID 并交给之后的 table 继续处理。具体流表如下：

table=0,priority=2048,in_port=1,IP,dl_Vlan=200,nw_dst=10.2.1.0/24 actions=strIP_Vlan, set_field:0x7d0->tun_id,goto_table:20

报文不会匹配到 table20 到 table60 的处理流程，在 table70 匹配到三层转发流表，根据报文的 VXLAN ID，目标 IP 地址，将报文的目标 MAC 地址修改为目标 IP 地址对应的 MAC 地址，并交给之后的 table 继续处理，具体流表如下：

table=70,priority=1024,IP,tun_id=0x7d0,nw_dst=10.2.1.3 actions=set_field:08:08:08:08:08:08 ->eth_dst,goto_table:80

报文不会匹配到 table80 到 table100 的处理流程，在 table110 匹配到二层转发流表，根据报文的 VXLAN ID，目标 MAC 地址，转发到

相应的虚拟机端口。如果目标 MAC 对应的虚拟机不在本节点，则转发到目标虚拟机所在主机的 VXLAN 隧道端口。具体流表如下：

```
table=110, tun_id=0x7d0,dl_dst=08:08:08:08:08:08 actions=output:2
```

至此，VM1 的数据包发送至另一个区域的 VM2，VM2 收到数据后，会按照上述步骤将响应数据返回，跨区域通信结束。

五、原型实践与效果

（一）物理架构概述

由于是软件 SDN 方案，实现网络功能的模块分布在每台服务器上，复杂性也卸载到 SDN 控制器和每台服务器上，所以物理架构相对比较简单。下图展示了原型平台的物理架构，整体架构由两个云网区域和一个 RI 区域组成。RI 区域由两台交换核心设备和两个区域各一台的区域核心设备组成。区域核心下联区域防火墙，防火墙下联交换机，交换机接入服务器。

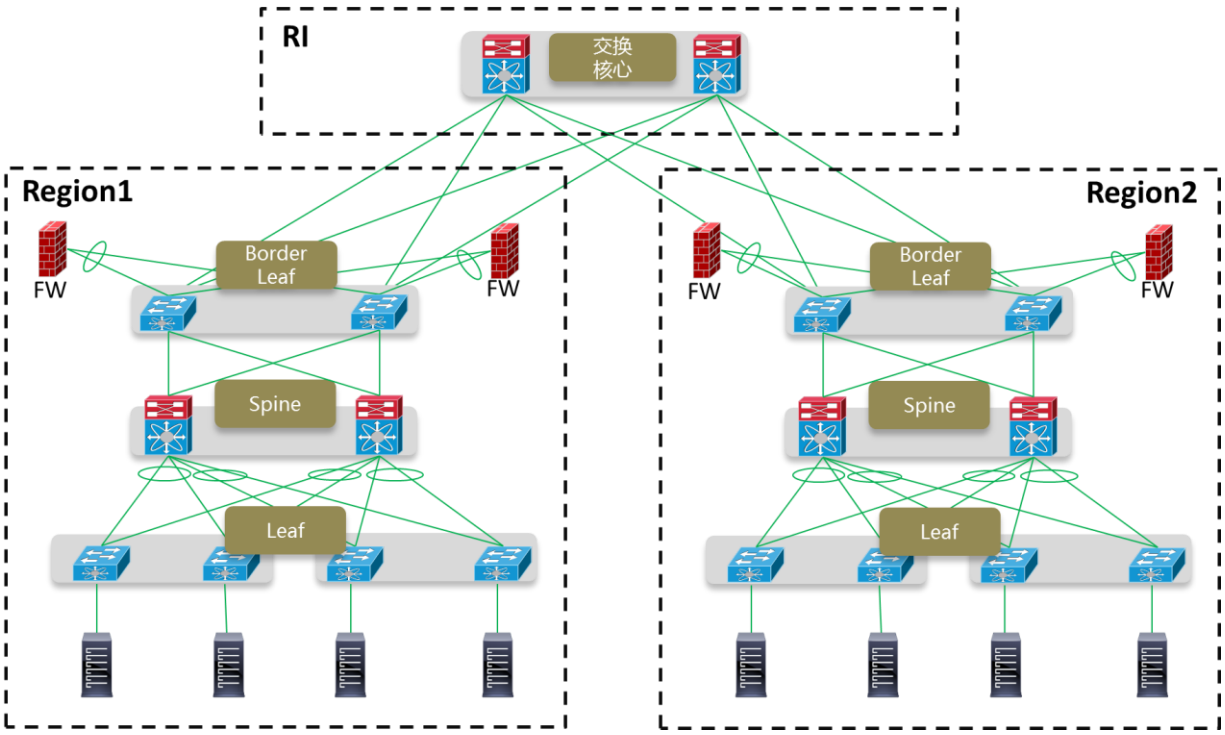

图 15 原型平台物理架构图

我们在中国银联的数据中心实验环境搭建了原型平台，平台由两个 SDN 云网分区组成，云网分区包含接入交换机，服务器，同时配备了防火墙。平台基于 OpenStack、OVS、ODL、Centos 等开源软件进行研发，相应软件版本情况见如表 2 所示。

表 2 软件版本情况表

软件	版本
OpenStack	L 版本
ODL	Beryllium-sr4
OVS	2.4.0
Centos	7.2

（二）管理控制平面概述

本次原型实践使用OpenStack作为云平台来管理虚拟资源的生命周期，向上提供标准API供用户使用，向下通过SDN控制器，防火墙驱动来实现对下层网络资源的抽象、隔离和调度。其中网络的控制平面使用ODL而非原生OpenStack的Neutron网络功能，ODL提供ML2和GBP的方式和OpenStack集成，本次实践使用ML2的方式。

（三）云网与云控平台集成

OpenStack需要管理我们实践环境中的二层虚拟网络，三层虚拟路由，以及防火墙，其中二层和三层的功能由ODL提供，防火墙功能由Neutron直接控制独立的防火墙实现。

ODL与OpenStack的集成需要两个平台的接口来实现。如图16所示，OpenStack的networking-odl项目提供ODL ML2 mechanism driver替代OVS driver，ODL L3 Plugin替代原生L3 agent。

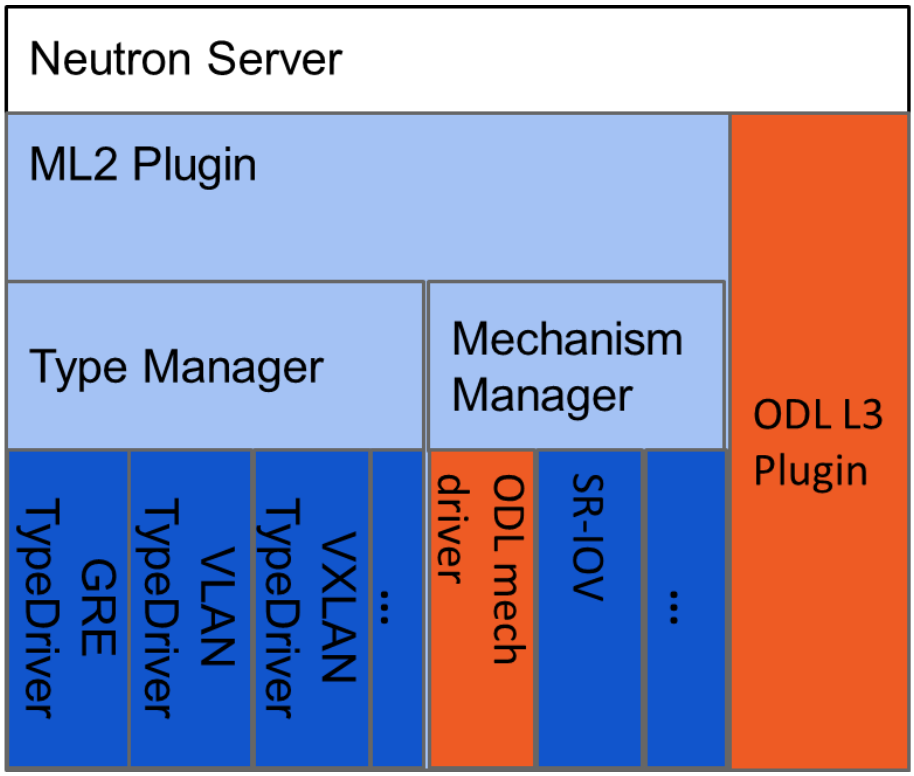

图 16 Neutron 集成模块示意图

OpenStack Neutron 的 ODL ML2 Driver 通过 REST API 将 Neutron 请求发送到 ODL 的北向接口，ODL 的 Neutron Northbound 和 Netvirt 项目分别提供对应 Neutron 的北向接口和业务逻辑，其中 MD-SAL 是 ODL 内部的数据交互模块。南向接口 OVSDB Southbound 和 OpenFlow Southbound Plugin 分别通过 OVSDB 和 OpenFlow 协议操作 OVS。

下面我们详细介绍每个模块的具体集成方式。

1.ODL 与 OpenStack 集成

本次原型实践使用 ODL 的 netvirt 模块与 OpenStack Neutron 集成，需要 OpenStack 和 ODL 两个平台的接口实现。

OpenStack 方面，需要在控制节点，网络节点，计算节点做以下配置的变更：

（1）控制节点

修改配置，使用 ODL ML2 mechanism driver 替代 OVS mechanism driver；

修改配置，使用 ODL L3 plugin 替代原生 OpenStack 的 L3 plugin；

配置 ODL 的访问路径和认证方式。

（2）网络节点

停止原生 OVS agent，OVS 由 ODL 控制；

停止原生 L3 agent，三层服务由 ODL 提供；

DHCP 服务使用 OpenStack 原生的 DHCP agent；

metadata 服务使用 OpenStack 原生的 metadata agent。

（3）计算节点

停止原生 OVS agent，OVS 由 ODL 控制；

ODL 方面，需要安装包含 netvirt 的 feature odl-OVSdb-OpenStack 并修改配置，打开 ODL L3 功能。

2.ODL 与 OVS 集成

如图 17 所示，OVS 主要由两个用户态进程 OVSdb-server、OVS-vswitchd 和 OVS 内核模块组成，其中 OVSdb-server 接收 OVSDB 协议的消息，保存 bridge，port 等信息到数据库，并通知 OVS-vswitchd 创建相应对象。OVS-vswitchd 同时接收 OpenFlow 协议的消息，操作

OVS 中的流表，最终使流表在内核模块中生效。

ODL 提供了 OVSDB southbound 和 OpenFlow Southbound Plugin 两种南向接口来操作 OVS，其中 OVSDB southbound 操作 OVS 上的 bridge，port 等元素，OpenFlow Southbound Plugin 操作 OVS 上的流表。

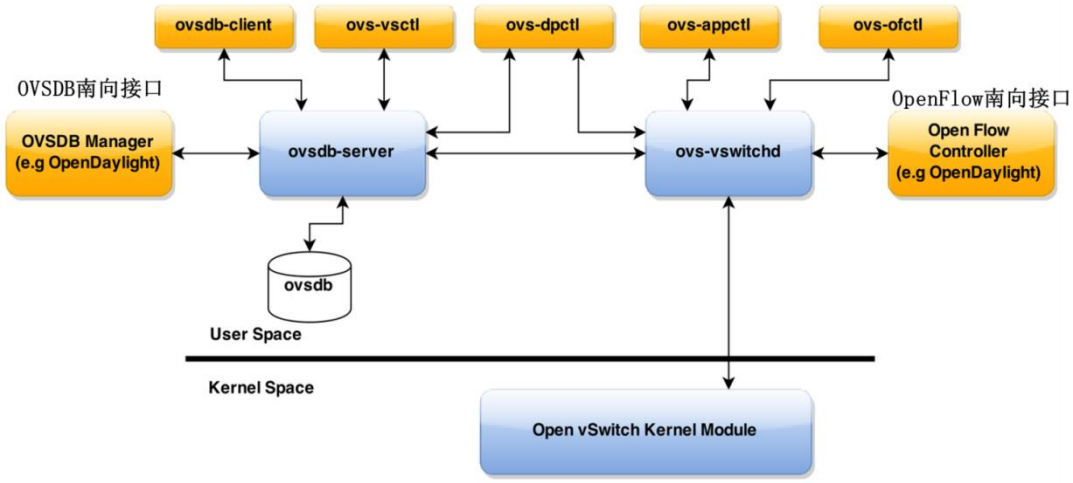

图 17 OVS 集成模块示意图

OVS 集成 ODL 需要在所有计算节点做以下配置：

将 OVS 的 manager 设置为 ODL，ODL 会在 OVS 上创建两个 bridge：br-int 和 br-ex，并会自动将这两个 bridge 的控制器设置为 ODL，之后 ODL 就可以通过 OVSDB 和 OpenFlow 来控制 OVS。

3.防火墙集成

在原型环境中我们使用 Neutron FWaaS 来驱动防火墙。

OpenStack 为防火墙服务提供了 V1.0 和 V2.0 两种 API 模型，FWaaS 2.0 在社区 M 版本中提出，现在仍在开发中，因此我们采用 FWaaS 1.0 模型和防火墙设备进行集成。

防火墙服务被抽象成多种虚拟资源，分别是 firewall，policy 和 rule。

一个 firewall 可以关联应用到多个 router，一个 firewall 使用一个 policy，policy 是 rule 的集合。

在原型环境中，防火墙物理上位于服务器和区域核心中间，逻辑上串联在租户虚拟路由和区域核心中间。在我们的环境中防火墙同时也承担路由器的作用，所以 FWaaS 除了需要驱动防火墙下发安全规则外，还需要在防火墙上配置路由，具体包括：

一条默认路由指向上层的区域核心，用于将目标是其他区域的流量送到区域核心

数条静态路由指向下层的租户虚拟路由，用于将目标是本区域的流量送到虚拟路由。

除此以外，为了支持多租户通信，创建每个防火墙实例时 FWaaS 驱动需要相应地在物理防火墙上创建 context，创建内向流量 internal 和外向流量 external 的 Vlan 子接口并加入 context 中。

（四）整体网络架构

原型环境的整体网络架构由云网分区和 RI 区域组成，其物理组网方式仍为 Vlan 模式，两个云网分区连接到 RI 互联区域。

防火墙和 RI 区域之间通过路由控制，防火墙以下的网络由 ODL 下发的 OpenFlow 流表控制。

服务器分别连接到一个管理网络，一个通往防火墙的 VLAN 网络，以及一个区域内通信的 VXLAN 隧道网络。

虚拟机之间的网络通信大致可以分为以下三种场景：

区域内同网段虚拟机二层通信，ARP 代答流表会替目标虚拟机代答 ARP 请求，并接收通信报文，根据目标 MAC 地址和 VXLAN ID，通过 VXLAN 隧道送到指定的节点，再匹配二层转发流表送到 OVS 的相应端口上。

区域内不同网段虚拟机三层通信，ARP 代答流表会替网关代答 ARP 请求，OVS 接收报文，匹配流表，由路由流表模拟路由功能并将报文的目标 MAC 地址修改为目标虚拟机的 MAC 地址，根据目标 MAC 地址和 VXLAN ID，通过 VXLAN 隧道送到指定的节点，再匹配二层转发流表送到 OVS 的相应端口上。

跨区域通信，发送出区域时，ARP 代答流表会替网关代答 ARP 请求，OVS 接收报文，匹配流表，静态路由流表会将报文的目标 MAC 地址修改为防火墙接口的 MAC 地址，给报文加上 VLAN TAG，并通过 VLAN 网络送到防火墙。两个区域防火墙之间由路由控制。接收进区域时，ARP 代答流表会响应防火墙对虚拟路由器接口地址的 ARP 请求。VLAN 到 VXLAN 转换流表会卸载报文 VLAN TAG 并打上目标网络的 VXLAN ID。如果目标虚拟机位于本节点，由二层转发流表送到虚拟机端口，如果目标虚拟机位于其他节点，由二层转发流表通过 VXLAN 隧道送到相应节点，再由相应节点的 OVS 接收。

原型环境的整体网络部署架构如图 18 所示：

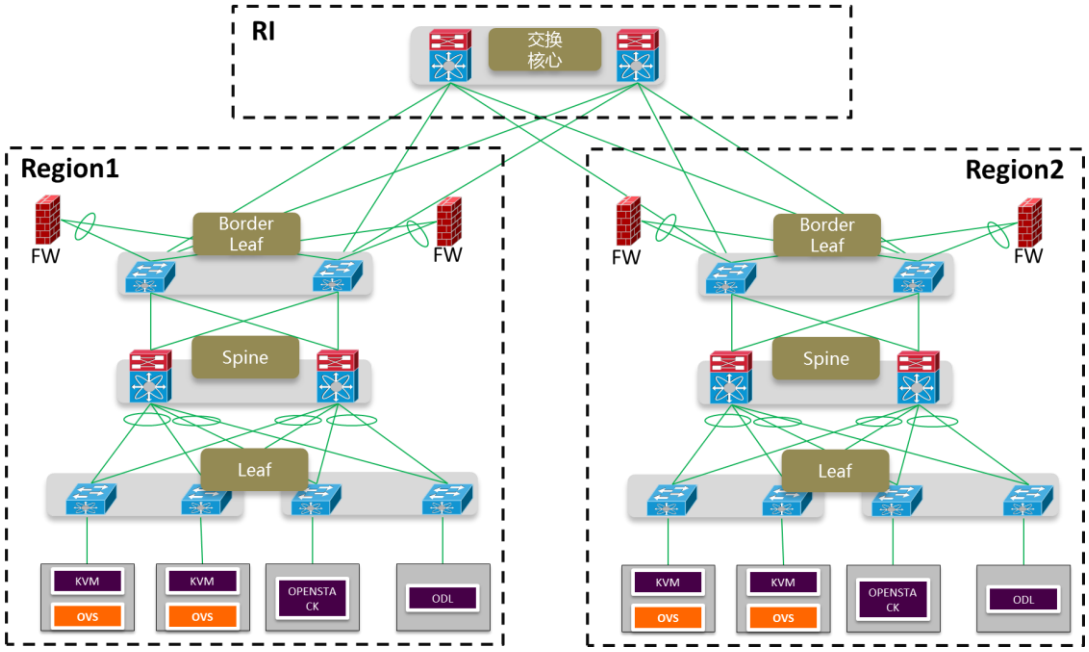

图 18 原型环境整体网络部署架构图

（五）效果展示

1.跨区域网络拓扑

原型平台基于多租户能力，创建了两个金融机构租户，两个租户的网络地址完全隔离复用，每个租户横跨基于 ODL 的软件 SDN 方案的两个云网分区资源，且共同复用所有硬件资源，通过核心交换网络进行数据互通，最终网络拓扑如图 19 所示。

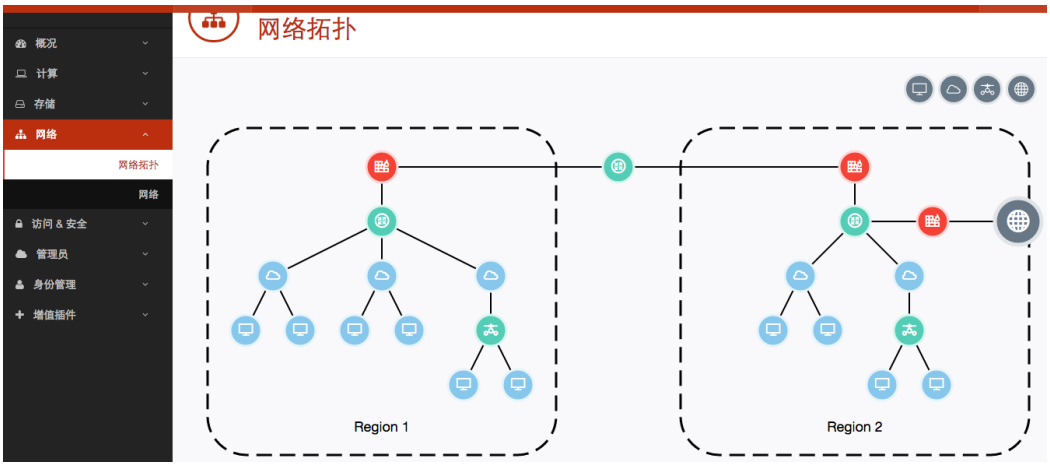

图 19 整体网络拓扑图

在此拓扑中，租户进行跨区域资源访问测试，可以相互通信，如图 20 所示。

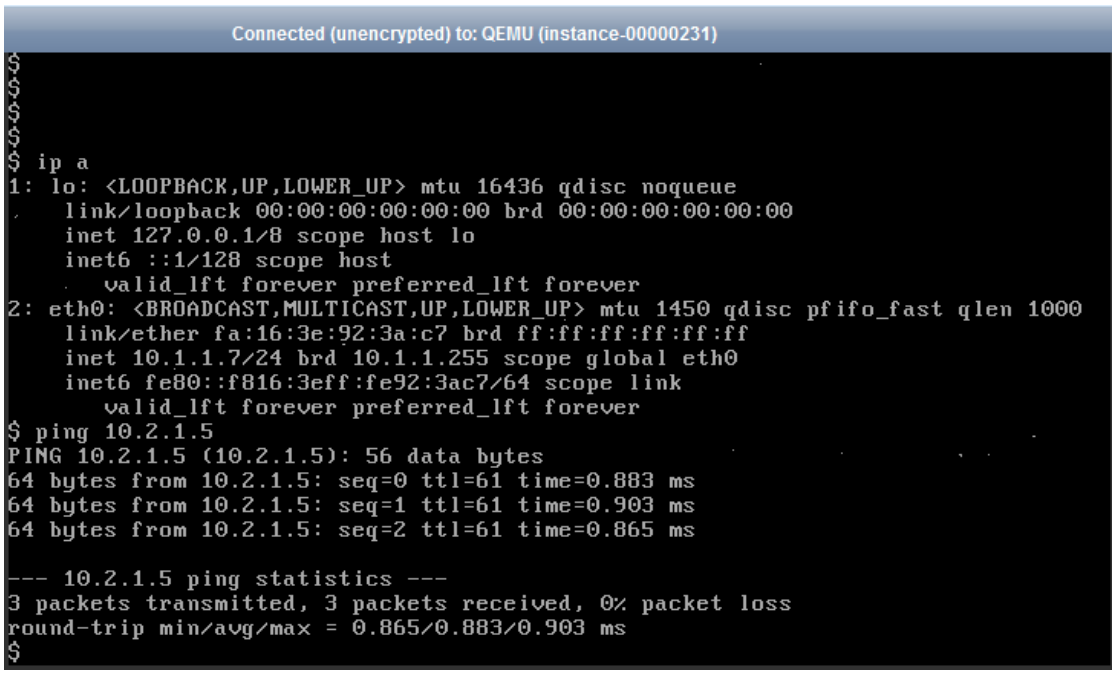

图 20 跨区域通信 ping 测结果图

2.性能测试

方案实现后，我们在万兆环境下对该方案进行了性能测试，并且与之前应用的 Nuetron 方案进行了对比。

（1）测试环境

硬件环境：CPU Intel E5-2630 V3； 网卡 Intel 82599；

软件环境：操作系统 Centos7.2；云平台 OpenStack L 版； 测试工具 IPerf；

测试项时长：5 分钟。经与 10 分钟测试时长比对，5 分钟测试时

长所得数据已足够平稳、精确，所以本次测试时长为 5 分钟；

测试包长：在单对虚机性能测试，测试包长的选取参考了 RFC2544，分别是 134、256、512、1024、1280、1456，其中 134 和 1456 是 IPerf 支持的最小和最大包长；在随后的极限性能测试中，采用最大包长 1456 进行测试。

（2）单对虚拟机性能测试数据（虚拟机配置：8c32g）

在测试中，我们首先测试了单对虚拟机的数据转发性能，在结果中挑选了部分代表性数据如表 3 所示。

表 3 单对虚拟机性能测试数据表

数据包大小（B）		256		512		1024		1456	
场景/测试项目		延时（ms）	带宽（G）	延时（ms）	带宽（G）	延时（ms）	带宽（G）	延时（ms）	带宽（G）
同网段同主机	ODL	0.463	28.9	0.449	28.7	0.599	28.8	0.451	29
	Neutron	1.298	18.5	1.267	19.1	1.016	23.4	1.031	22.9
同网段不同主机	ODL	0.92	0.329	1.376	0.778	2.078	1.75	1.861	2.41
	Neutron	3.765	0.197	4.78	0.501	4.491	1.2	3.613	1.71
跨网段同主机	ODL	0.437	28.3	0.42	29.2	0.677	27.9	0.356	30.1
	Neutron	2.989	0.141	3.962	0.321	4.506	0.689	4.793	1.02
跨网段不同主机	ODL	1.161	0.34	1.725	0.801	1.869	1.7	1.665	2.32
	Neutron	2.898	0.181	2.929	0.422	3.273	0.907	3.353	1.28
跨区域通信	ODL	0.696	0.276	0.851	0.706	1.168	1.57	1.235	2.25
	Neutron	4.246	0.206	4.577	0.551	4.561	1	3.096	1.29

结果分析：从测试数据上看，ODL 方案不管是在延时上还是在带宽上相比 Neutron 来说都有了比较好的提升。经计算，ODL 方案时延

较 Neutron 平均降低 68.8%；带宽（不含跨网段不同主机场景）平均提升 39.3%。

（3）极限性能测试数据

在本项测试中，逐步增加虚拟机数量，在同网段不同主机的场景下，采用最大包长测试跨主机通信的极限性能（带宽），得到数据如下图 21 所示。

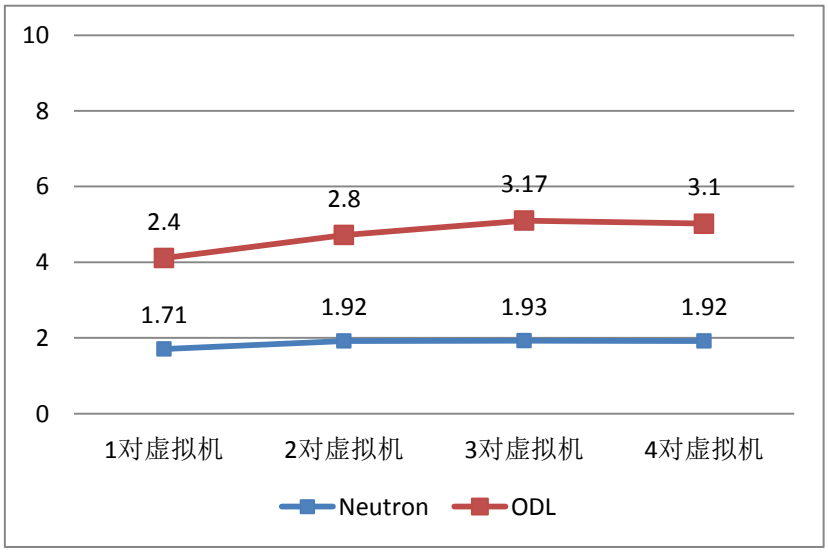

图 21 极限性能测试数据图

测试结果：在跨物理机通信的情况下，ODL 与 Neutron 方案的极限带宽分别为 3.17G 与 1.93G，ODL 高出 Neutron 64.2%。

（4）测试结果分析

通过测试，我们发现虽然 ODL 方案在万兆环境下的性能依然高于 Neutron 方案，但整体测试数据不佳，最高带宽只能达到 3.17G，仅占整体带宽的 30%，不能对网络资源进行较高效率的利用。

经过分析，我们认为产生该问题的主要原因是因为硬件网卡不支持 Vxlan offload 功能造成的。具备 Vxlan offload 功能的网卡，能够识别 Vxlan 数据包并对包头进行相应的处理，将原本在协议栈中进行的分片、TCP 分段、重组、checksum 校验等操作，转移到网卡硬件中进行，降低系统 CPU 的消耗，提高处理性能，能够在使用 Vxlan 通信的情况下较大幅度的提升带宽。而本次测试中使用的 Intel 82599 网卡则不具备该功能，所以造成总体性能较低。

六、总结与展望

（一）工作总结

通过本次研究我们形成了许多技术积淀。首先，我们根据实际需求，设计出一整套的软件 SDN 解决方案，包括整体的网络架构、能力设计与流表框架，相比其它方案来说，优势如下：

1.整体完全自主可控；

2.方案按照金融需求设计，相比其它方案更能贴合金融场景下的应用；

3. 相比商业方案更加灵活、成本更低。

其次我们通过对 ODL 的开发，对原生的能力进行了增强与优化，掌握了软件控制器功能定制化的能力；最后，在性能测试中，对 Linux 环境下的网络接收性能调优也进行了相关的研究，这对于我们来说也是很好的一次技术积累。

虽然当前方案已经实现，但是在具体的实践过程中仍存在一些问题有待解决，具体如下：

1.支持去 Floating IP 的分布式网关功能仍不完善，目前平台内分布式的外部接口进行 ARP 响应会造成交换机的 MAC 表震荡，当平台物理节点的数量较多的时候会影响网络稳定。因此，后续我们会对该实现机制进行进一步的优化，采用定向下发 ARP 响应流表的方式进行功能实现；

2.本方案中只实现了通过 Vlan 连接物理防火墙的流表框架，其实通过 Vxlan 连接虚拟防火墙的流表我们也设计了出来，只不过当前应用场景较少所以没有在控制器上进行能力实现。后续随着 NFV 技术在金融行业的推广应用，我们也会择机对该 Vxlan 的连接方案进行实现；

3.万兆环境下网络性能极限测试效果不佳，具体原因已经在上文中进行了分析，后续将优化测试环境，更换具备相应能力的网卡进行进一步测试；

4.控制器的高可用性仍待加强，针对 ODL 控制器的高可用方案社区中也没有给出较好的方法，所以在后续工作中也希望能得到更多业内专家们的帮助与支持。

（二）展望

中国银联正积极探索软件定义网络技术在下一代金融云环境中的深化研究与应用，并已与中国银行、中国农业银行、中国工商银行、

中国建设银行、中国交通银行、兴业银行、恒丰银行、上海银行等机构就 SDN 技术在金融行业中的应用与联合研究需求进行了紧密沟通。当前结合各单位聚焦研究点，中国银联已发起“跨数据中心多云协同资源管理技术”的联合研究课题，希望更多的银行等金融合作机构能够参与到相关的研究工作中，共同推进软件定义网络相关的金融科技联合研究与应用。