

第 3 章 Chapter 3

SDDCN 概述

SDN 起源于校园网，不过商用的第一枪却在数据中心打响，究其原因是云计算和 SDN 在设计思想上的深度契合。从这一章开始，将对 SDN 在数据中心的应用进行介绍。本章将对 SDDCN 的设计原则、架构以及关键技术进行介绍，争取为后面章节的展开起到提纲挈领的作用。

3.1 需求

云计算时代的数据中心对网络提出了很多新的需求，首先要解决的是大规模网络的自动化和集中式控制，另外一些新型的 IT 应用架构也要求网络能够变得更加智能。长期以来，传统网络难以满足上述需求，而 SDN 架构的几大特征完美地契合了这些需求。

3.1.1 自动化与集中式控制

对于数据中心的网络管理员来说，新业务的开通通常意味着地址规划加上配置 VLAN、ACL、路由、防火墙等诸多规则，业务终止后还需要回收相关的资源和规则。这些都需要网管去手动操作，然而手动配置的效率是极为低下的，而且很容易出错。不过传统数据中心中，同一业务的资源分布较为集中，因此这些操作不会涉及过多的设备，手动配置网络在成本上是可以接受的。不过，手动配置还意味着较长的业务交付周期，开通一个新的业务往往需要等待数周的时间。

云计算提出了资源池化的思想，虚拟化技术的发展打破了基础设施的物理边界，管理员人均需要维护的虚拟机数量有了数量级的提升，而且为同一业务分配的虚拟机可能分散在不同的机柜、机房甚至不同的数据中心，管理员很难准确地知道虚拟机某一时刻所在的

位置。如果再考虑到虚拟机不断地进行着迁移，那么此时手动配置网络将变成一件不可接受的事情。同时，云计算还深刻地变革了传统数据中心的业务交付模式，用户通过在 Web 界面上点选一些按钮、输入一些参数就可以自助地开通或者变更业务，其时间要求达到分钟级甚至秒级。另外，随着防火墙、负载均衡、IDS/IPS 这些 L4 ~ L7 设备的虚拟化，业务的组网方式将变得更为复杂多变。“提交变更申请——分解操作流程——手动配置网络”，这样的传统流程不可能实现如此的敏捷性。

上述自动化需求是如此强烈，使得 SDN 成了云数据中心网络的不二之选。SDN 从架构上就原生地具备自动化和集中控制的能力，通过开放转发设备的控制接口，SDN 控制器可以自动地探测网络拓扑，并下发 VLAN、ACL、路由等诸多规则，做到转发设备的即插即用。SDN 控制器还可以向用户暴露业务接口，用户在此接口基础之上进行二次开发，使得网络可以随用户业务动态地进行调整。SDN 对于运维自动化能力的提升同样显而易见，通过对流量和网络状态进行综合分析，SDN 将有可能对云中的网络故障自动地完成隔离、排查甚至预警。

理论上来说，控制接口、业务接口的开放即提供了网络自动化的基础，不过真正实现一套能够在云数据中心商用的 SDN 系统并非一件易事。控制器内部的层次设计，同一层次不同接口的统一抽象（或者说不同层次接口间的适配），都是好说不好做的事情，其中涉及太多的技术流派之争和复杂的市场利益纠缠。不过，随着 OpenFlow/OVSDB 和 OpenStack Neutron 日渐成熟，SDN 在云数据中心中的落地显现出了标准化的前景，OpenFlow/OVSDB 作为设备接口所体现的灵活性和 Neutron 作为业务接口所体现的包容性，都极大地推动了 SDN 技术在云数据中心网络中的发展，也使得业界深刻体会到了网络开放对于 IT 系统自动化能力的整体提升。SDN 也因而成为云数据中心网络演进的必然趋势。

除了自动化外，云计算所提倡的集约式运营还对集中式调度有着明确的需要。虚拟机作为工作负载经常需要动态地进行迁移，这就需要有一个集中式的控制器来将虚拟机在服务器中进行分布。近几年兴起的软件定义存储（Software Defined Storage, SDS）也提出集中地调度闪存、整列的资源，以提高存储资源利用率和存储节点命中效率。

同样，SDN 所具备的对网络进行集中式调度的能力也是云计算所需要的。通过 SDN 控制器集中地向网络边缘分发网络策略已经广泛地应用在云数据中心网络中，当虚拟机发生迁移时，SDN 控制器可以及时地得知虚拟机迁移后的位置，能够将相关的网络策略重新进行部署，以避免迁移过程中由老旧的网络策略导致的业务中断。另外，有一些论文中提出，通过对网络当前状态，如拥塞、丢包率等指标来优化承载关键业务的虚拟机的位置布放，从而达到优化通信效率、节能等目的。

尽管 VxLAN 等隧道技术可以结合 SDN 控制器的集中调度能力，但由于 SDN 的控制权仅存在于网络的边缘，仍无法对网络核心的传输进行调度。云数据中心网络希望能够对东西向流量进行无阻塞传输，虽然 OSPF/BGP/ECMP 等路由技术已经间接为二层提供了多路径传输的能力，然而它们只能基于流的静态特征对路径进行选择，一来无法感知流的动态特征，二来无法结合网络的实时状态。因此，虽然网络中不存在闲置链路，但各条链路上

的负载通常会很不均衡，网络整体的业务吞吐率仍然难以得到有效保障。

若能够使用 SDN 同时控制网络边缘和网络核心，结合测量、流量调度、QoS 等技术，SDN 控制器将拥有流量和网络的动态全局视图，可以有效地对网络核心的传输进行调度，提高云数据中心的业务吞吐率。如此一来，网络将能够真正地做灵活智能，与计算、存储资源三维一体，全面迈向软件定义的数据中心，实现 IT 基础架构的大融合。

3.1.2 应用感知

数据中心的建设是一个大型的系统工程，不考虑电力制冷以及人力上的开销，以每机架为单元，ToR 的成本大概只能占到 5% ~ 8%，服务器的成本占比大约在 25% ~ 35%，而在软件方面所投入的成本，包括操作系统、虚拟化、应用以及管理软件，大概可以占据 70% 甚至更多。因此，与在电信运营商中的基础性地位不同，网络在数据中心内部应该被定位于业务与应用的辅助，如果让应用围着网络来转，显然就是本末倒置了。

虽然数据中心和电信运营商在网络设备厂商中通常是两条独立的产品线，不过数据中心网络架构的设计至今仍然没有摆脱“以网络为中心”的思路，虽然 SDN 为网络提供了可编程的能力，但就目前来看，SDN 的 API 基本上都是为了业务的自动化而设计的，应用仍然只能 Over the Top。大数据、人工智能等新型应用架构的逐步普及，对未来的数据中心网络将提出更高的要求，网络需要能够为这些应用保障一些严苛的 SLA 需求，如传输带宽、端到端时延和传输抖动，等等。因此，网络必须重新思考未来在数据中心的定位，SDN 需要能够更为深刻地理解应用、更为细致地感知应用以及更为灵活地适应应用，使得应用真正能够实现“play with the network”，而非“play around the network”。

下面以 Hadoop 为例，看一看 SDN 为其能够为大数据提供哪些价值。Hadoop 的架构是 Multi-Stage 的，先把数据分布式地存下来，一个 Compute Job 提出后会被拆分为多个 Compute Task 并分配给不同的节点，每个节点在本地完成一小部分 Task 的计算，最后将这些 Task 的计算结果汇总在一起，作为 Job 的最终结果进行返回。在 Hadoop 集群的实现中，分为 Client、NameNode、DataNode、JobTracker 和 TaskTracker 五种主要的角色，主要的工作流如下。

- 1) 数据读取进来之后，Client 将其分解为不同的 Block，然后向 NameNode 发出请求以获得 Block 的存储位置。
- 2) NameNode 为每个 Block 返回一组 DataNode，并对此进行记录。
- 3) Client 根据 NameNode 的返回结果，将 Block 写入相应的 DataNode。
- 4) 当 Job 被提出后，Client 将 Job 通知给 JobTracker，JobTracker 会从 NameNode 处获取存储了相关 Block 的 DataNode。
- 5) JobTrack 将 Job 拆分为不同的 Task（可分为 MapTask 和 ReduceTask），将其发布给不同的 TaskTracker 进行执行，并对此进行记录。
- 6) MapTask 对本地的数据进行处理，处理完成后将结果存在本地。
- 7) ReduceTask 从 JobTracker 询问 MapTask 的执行状态，从已完成的 MapTask 所在的

节点拉取 Map 的结果。

8) 当所有的 MapTask 都完成后, ReduceTask 得到所有的 Map 结果, 并将它们合并为最终结果。

9) 此时 JobTracker 将 Job 状态置为成功, Client 读取最终结果。

以网络的视角来看上述过程: ①将数据写入 DataNode 的过程需要高带宽; ②NameNode、JobTracker 和其他角色间交互的控制信令需要低时延; ③Mapper 和 Reducer 的数量通常是 M:N 的 ($M>N$), 在 Shuffle 的过程中, 可能会出现大量 Many-to-Few 的流量, 需要能够处理 TCP Incast 的问题; ④只有在所有 Task 都处理结束后 Job 才会完成, 需要尽可能地缩短 Reducer 和 Mapper 间 Shuffle 流量的传输时间; ⑤集群会同时通过处理多个 Task/Job, 网络中会存在大量的 Mice Flow, 需要能够处理 MicroBurst 的问题; ⑥为了实现高可用, DataNode 间会进行数据的备份, 需要放置这种次要流量抢占上述工作中流量的带宽。

针对上述过程进行优化, 整体的目标是降低 Job 的 Completion Time, 网络需要能够将大数据流量与其他应用的流量区分开, 并识别出大数据的不同流量, 保障高带宽型和低延时型流量的 QoS, 降低次要流量的优先级。SDN 控制器需要:

1) 将上述理解嵌入到控制算法中;

2) 与 Hadoop 建立接口, 获取到集群中不同角色的位置分布信息;

3) 获取网络的实时状态, 如物理拓扑、链路带宽、拥塞率, 等等。通过将角色的分布信息和网络实时状态输入到算法中, 计算出优化后的路径并下发到网络当中, 或者对当前路径中大数据流量的 QoS 进行保障。

解决 Incast 和 MicroBurst 的问题, 只能从流量的源头来想办法, 这需要 Hadoop 自身能够根据网络的状态去优化 Block 和 Task 的分布。因此, SDN 控制器不仅需要从 Hadoop 处获得信息, 也需要将网络的实时状态提供给 Hadoop, 比如 Server-to-Server 流量统计、Rack-to-Rack 流量统计、拥塞发生点、端到端时延等, 使得 JobTracker 和 NameNode 在进行调度时能够结合网络的实际状态来优化 Block 和 Task 的分布, 优化 Job Completion Time。

可以看到的是, 针对应用来进行网络优化并不是一件很好做的事, 因为不同应用的架构、流量模式和通信需求都不一样, 目前仍然只能是 case by case 地去考虑。关于大数据优化的更多介绍, 可以参考本书 9.10 节中的内容。另一个常见的场景是 VDI, 可以通过 SDN 来监测 / 防止启动风暴, 以及保证 IP Storage 的传输带宽, 来优化虚拟桌面的 QoE。

不过, 通过对流量进行采集, 再结合大数据和机器学习等技术, 便能分析出应用的行为模式。将这些行为模式告知 SDN 控制器, SDN 控制器即可采取有效的优化措施, 从而实现网络对于应用的动态感知与自适应。这种与其他工程领域间的交互能力, 完全得益于 SDN 为网络所带来的开放性, 是传统网络所不可能具备的。

另外, 抛开以上的技术原因, SDN 适合在云数据中心落地的基础还得益于数据中心网络的以下特征:

□ 静态, 网络拓扑和地址规划较为固定;

- 统一，设备类型相对单一，网络管理权限较为集中；
- 独立，通常不会用于传输过路流量，发生故障不会影响外部网络；
- 负担轻，设备和链路成本较低，数据中心新建时无需考虑存量设备。

经过多年的实践，目前业界已经广泛地认可了 SDN 对于云数据中心竞争力的提升，云数据中心已经成为 SDN 落地的主战场。因此，云数据中心的管理人员们没有必要再怀疑 SDN 是否具有价值，而应该考虑的是如何量体裁衣，使 SDN 更为有效地为自身的业务和运维提供价值。

3.2 整体架构

从这里开始切入 SDDCN 正题，首先，本节将对 SDDCN 的整体架构及其实现形态和功能设计进行抽象性的介绍。

3.2.1 实现形态

在 SDN 理想的实现形态中，转发设备中的控制逻辑被完全剥离出来，上移到集中式的控制器中实现，控制器能够直接控制设备的转发表，控制器对应用 / 编排暴露 API，如图 3-1 所示。

在上述理想形态中，集中式的控制器对网络有着完整的控制权，SDN 的优势得到最大程度的体现。但是在 SDN 实际落地的时候，考虑到扩展性与可用性，其实现形态通常与理想形态有所区别。下面针对 SDDCN 介绍几种常见的实现形态。

为了解决纯集中式的缺点，可采用集中式和分布式相结合的方式，控制功能在集中式的控制器和转发设备间进行切割，控制器统一对应用 / 编排暴露 API。在图 3-2a 中，控制器将某些可在本地执行的控制功能卸载到转发设备中实现，转发设备间不会运行分布式协议。而在图 3-2b 中，转发设备间会运行一些分布式协议，实现基本的转发或者状态监测，而集中式的控制器主要负责对流量进行优化。这种架构的优点是控制器的压力减小，网络对于控制器的依赖性降低，缺点是控制平面实现较为复杂，控制功能在控制器和转发设备间尚无明确的切割原则。

集中式和分布式相结合，在实际的工程实现中还有一种常见的变形，是以集中式的数据库来代替集中式的控制器，由数据库对应用 / 编排暴露 API。在图 3-3a 中，转发设备上保留本地控制逻辑，但彼此之间不运行分布式协议，转发设备间的状态同步是以集中式的数据库作为中间件来实现。在图 3-3b 中，转发设备上保留完整的控制逻辑，彼此之间运行分布式协议进行状态的同步，集中式的数据库负责把配置自动地推给各个转发设备，但是数据库本身不对转发进行控制。这种架构的优点是数据库的集群机制成熟，具备良好的扩展性与可用性，缺点是一些较为复杂的网络优化逻辑难以由数据库来实现。

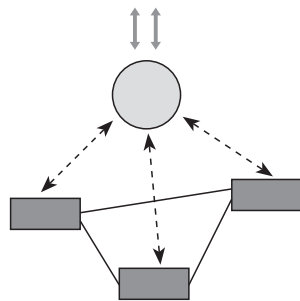


图 3-1 纯集中式的实现形态

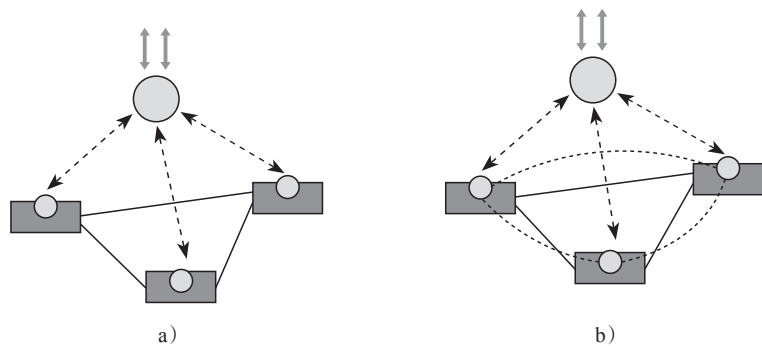


图 3-2 集中式和分布式相结合的实现形态

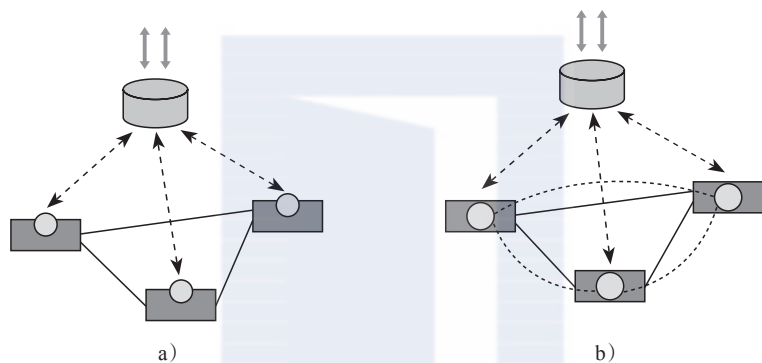


图 3-3 以数据库为逻辑集中点的实现形态

还有一种较为“另类”的 SDN 架构，是转发设备上保留完整的控制逻辑，设备间运行分布式协议，网络不依赖于任何集中式的控制器或者是数据库。与传统网络不同的是，转发设备会直接向应用 / 编排暴露 API，以实现可视化、自动化或者转发优化，当然通过设备暴露的接口也可以对接第三方控制器。另外在一些实现中，如图 3-4b 所示，通过一台转发设备暴露的 API 即可获得全网的状态与数据，实现网络与应用 / 编排 / 控制的单点对接。这种架构的优点是彻底消除了集中点，扩展性和可用性最强，缺点是难于对网络的控制逻辑进行优化。

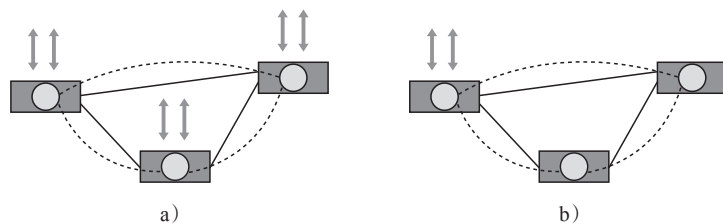


图 3-4 以分布式为主的实现形态

目前，多数 SDDCN 解决方案采用的是图 3-2 或图 3-3 中的某种形态，少数解决方案会

做混合的实现，而图 3-4 中的架构多见于白盒交换机厂商的设计中。在本书的 4 ~ 7 章中，将对商用和开源 SDDCN 解决方案进行深入的介绍，看过这些章节后，读者可以再翻回来重新理解上述的内容，想必会有更好的效果。

3.2.2 功能设计

图 3-5 是 SDDCN 的功能架构。数据平面上，L2 ~ L3 设备实现基础转发，L4 ~ L7 设备提供增值服务。控制平面上，有一块大的逻辑负责对设备上的路由与资源进行控制与配置，另一块大的逻辑是可视化，把设备上的数据采集上来并进行处理与分析，然后反馈给其他模块以实现闭环的自动化。编排与应用层负责实现不同的业务逻辑。安全与高可用贯穿着 SDN 的三层，为系统的落地提供保障。

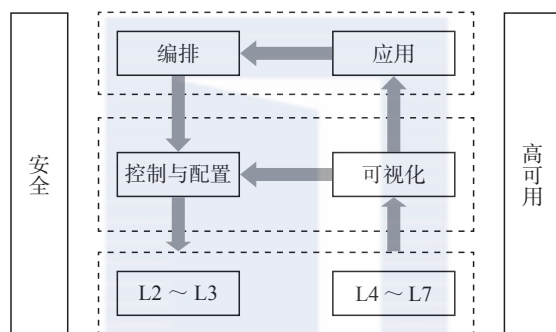


图 3-5 SDDCN 的功能架构

图 3-5 中的模块和控制流已经很清晰了，这里不再做过多的解释。实际上，图 3-5 中的架构不仅局限于 SDDCN，其他场景中的 SDN 系统也正在向这种架构进行收敛。在下一节的内容中，将围绕着该架构，对 SDDCN 中的关键技术进行介绍。

3.3 关键技术

本节将对 SDDCN 的关键技术，包括网络边缘、网络核心、服务链、可视化、安全和高可用进行概要性的介绍。这些关键技术并不是独立的，彼此之间会有很多的交叉，希望读者能够将它们结合起来进行理解。

3.3.1 网络边缘

网络边缘把控着流量的入口，负责在流量开始传输前对其进行一些预处理。相比传统的数据中心，云数据中心的负载形态和流量特征都发生了深刻的变革，而传统网络中接入层的设计，从各个方面来看都难以满足新的要求，对于网络边缘的改造势在必行。

传统数据的中心里，主要工作负载为服务器，与 Internet 互通的南北向流量占主导地

位，因此通常采用 3-Tier 的 Hierarchy 网络设计。接入层是网络的边缘，服务器直接物理上联接入交换机，在接入的布线方式上通常采用 EOR (End of Row) 或者 TOR (Top of Rack)。EOR 方式中，接入交换机位于独立的网络机柜，服务器的接入需要经过长距离的走线，EOR 的布线管理起来很复杂，但接入设备管理起来较为容易。TOR 方式中，接入交换机位于各个服务器机柜的架顶，简化了服务器接入的布线，但是接入设备数量较多难以管理。除了转发以外，接入交换机主要的功能还包括 VLAN 标记、端口安全和 QoS 等。

1. 虚拟交换机

云数据中心里，虚拟机代替服务器成为主要的工作负载，虚拟机间的东西向流量爆发，传统数据中心的层次化网络设计已经难以有效承载东西向流量，以 Leaf-Spine 模型为代表的扁平化设计开始流行。由于虚拟机网络的第一跳发生在服务器内部，因此 vSwitch 成为云数据中心网络边缘的核心技术。不过，在 vSwitch 中实现过多的网络功能会造成很多的开销，影响服务器中能够容纳虚拟机的数量，因此一些功能需要被卸载到 vSwitch 下一跳的物理交换机中。在 Leaf-Spine 的网络架构下，TOR 取代 EOR 成为主要的布线方式，通常由 vSwitch 和 TOR 共同来完成网络边缘的接入工作。

随着 SDN 在云数据中心的广泛流行，vSwitch 和 TOR 上能够执行的功能也变得更为灵活。OpenFlow 的使用极大地拓展了 ACL 的语意，带来了更为多样的匹配条件和处理动作。VxLAN 逐渐取代 VLAN 成为云数据中心网络虚拟化的主流技术，带来了更多的租户数量和不受物理限制的大二层。另一个常见的场景是，虚拟机经常会发生迁移，网络边缘的策略也需要随之进行自动地变更，如 VLAN 和 QoS 等，而 SDN 控制器上的全局视图也有助于实现策略跟随的自动化。此外，如果流量对于 L4 ~ L7 层的服务有要求，那么在网络边缘上还要支持服务链功能。关于服务链，后面会有一个小节的内容对其进行单独的介绍。

vSwitch 的代表性技术是由 Nicira 发起的，目前承接于 Linux 基金会的 OpenvSwitch (OVS)。OVS 同时支持各版本的 OpenFlow 和丰富的传统网络协议，以 OVSDB 作为 OVS 的管理协议基本上已经成为业界的事实标准。当然，要支持灵活的控制必然会带来的是实现的复杂性，因此 OVS 的虚拟转发通道在效率上是要低于传统的 Linux Bridge 的，不过随着 OVS 对于 DPDK 技术的深度整合，目前 OVS 在转发性能上也得到了长足的提升。OVS 目前已经在各个领域得到了广泛的应用，大多数 SDDCN 解决方案在网络边缘都采用了 OVS 作为 Hypervisor Switch，以获得对网络边缘的灵活控制能力。不过即使开启了 DPDK，由 OVS 来实现 VxLAN 等隧道的封装/解封装，开销还是很大的，因此在对于吞吐量有要求的生产环境中，可以通过 TOR 来卸载隧道的封装/解封装。另外，SmartNIC 也可以用于卸载 vSwitch 的部分功能，但目前仍不够成熟。相比 vSwitch-vSwitch VxLAN，TOR-to-TOR 的 VxLAN 还有另外一个好处，那就是可以有效地减少 VxLAN 隧道的数量，降低 Overlay 的维护复杂度。对于一些其他的功能，则可以视情况在 OVS 和 TOR 间进行拆解，若使用 TOR 实现，性能要好一些，但是要考虑到 TOR 所支持的功能集合以及转发容量。

2. 基于 Overlay+SDN 的虚拟网络

SDN 能够为 VxLAN 隧道的封装提供全局的视图，包括如何在 VTEP 间建立隧道、虚

拟机的接入位置, VLAN ID 和 VNI 的映射关系, 等等。全局视图的来源有两种: 一种是由 OpenStack 这种云管理平台直接将虚拟机的接入位置告知 SDN 控制器。这种方式下, SDN 控制器通过 Proactive 方式将转发表预置在数据平面。另一种是 SDN 控制器不通过与 CMS 交互来学习虚拟机位置, 而是当虚拟机上线时触发 Hypervisor 告知控制器, 这种方式下 SDN 控制器只能通过 Reactive 的方式动态生成转发表。两种方式互有优缺点: Proactive 方式中, 流量到来时可以直接在数据平面本地完成匹配转发, 控制信道的实时压力较小, 但是这种方式会将很多不常用到的转发表推送下去, 在使用 TOR 进行隧道处理时会造成硬件资源的浪费。Reactive 方式中, 首包会产生延迟或者被丢弃, 而且控制信道的实时压力较大, 但是数据平面上只会存在需要用到的转发表。实际情况中, 有可能需要在两种方式间进行平衡。

SDN 还有助于网络边缘对流量进行优化。二层流量的优化体现在对于 BUM 流量的处理上, 借助于控制器中的全局视图, ARP Request 得以在本地进行代理, 一些情况下 DHCP 也可以由 SDN 控制器在本地直接处理。SDN (尤其是 OpenFlow) 的灵活性对网络的边界造成了极大的冲击, 人们发现三层流量也不一定要通过物理路由器来处理了, 任播思想的流行和分布式路由技术的发展, 使得三层流量能够在 vSwitch/TOR 本地进行路由, 这样不仅可以简化网络路径, 还能够避免路由的单点故障。不过, 分布式路由技术也可能导致其他的一些问题, 比如网络边缘功能过于复杂、故障定位困难等。

通过 SDN 对流量进行优化时, 使用 OpenFlow 可以获得相当的灵活性。由于 VM 间的通信仍然不得不依赖于大量的 ARP, 因此 ARP 泛洪的抑制是大多数控制器必须要解决的问题。使用标准的 OpenFlow 需要由控制器来代理回复 ARP Reply, 当网络规模比较大的时候, 控制器处理 ARP 的压力会急剧增大, 因此更好的方式是使用 OpenFlow Nicira Extension 将 ARP Reply 的内容预置在 OVS 中, 由 OVS 本地数据平面对 ARP 进行代理回复。DHCP 同样依赖于广播, 不过其数量远不如 ARP 频繁, 因此大多数 SDDCN 方案中 DHCP 仍然通过广播交给 DHCP 服务器去处理。不过, OpenFlow 控制器同样可以代理 DHCP 的回复, 但是由于 DHCP 的构造是基于 UDP 之上的, 因此 OVS 无法在数据平面本地处理 DHCP, 必须由控制器进行回复。分布式路由既可以完全由 OpenFlow 来实现, 也可以使用 OpenFlow 结合 vRouter 来实现, 不同的 SDDCN 解决方案的设计有所不同。目前业界更接受 OpenFlow 结合 vRouter 的实现方式, 不过直接使用 OpenFlow 实现路由, 整体性能一般来说会稍好一些。

3. 虚拟网络与物理网络的对接

由于很多的原因(如数据库要跑在物理机上、P-V 的工作负载迁移、物理防火墙的部署等), 虚拟化环境需要对接物理环境实现二层互通, 这个对接工作通常也需要由网络边缘来实现。二层的对接, 需要 vSwitch/TOR 来作为 L2 Gateway 完成异构二层网络间的封装转换, 同时 L2 Gateway 往往还需要运行 MAC 自学习和 xSTP 等传统二层网络的控制逻辑。由于二层的对接处会承载大量的东西向流量, 因此需要为 L2 Gateway 设计有效的高可用机制。对接了二层后, 仍需要考虑的一个问题是: 该二层的 IP Gateway 是部署在虚拟环境中,

还是物理环境中？这要根据企业实际的情况进行择优处理。值得读者注意的是，即使使用了SDN来解决L2的对接，仍然难以避免泛洪的使用，这是因为物理工作负载的MAC地址SDN控制器最初是不得而知的。

与外界三层对接，需要在网关节点上（可以为ToR，也可以为服务器）运行传统的路由协议，如OSPF/BGP等。TOR通常能够支持这些协议的运行，而服务器中的vSwitch并不具备三层的功能，因此通常需要将vSwitch和协议栈做集成部署，或者将vSwitch直接替换成vRouter来完成路由的工作。SDN控制器在该场景下的功能一个是通过某种方式（一般是在控制器上运行OSPF和BGP）获取到协议栈/vRouter所学习到的外界路由，另一个是将外界路由和DC内部路由进行重分布。如果采用的是vSwitch+协议栈的方案，由于协议栈只有控制平面，转发仍然要依赖于vSwitch的datapath，那么SDN控制器就需要向vSwitch下发流表进行转发。如果采用的是vRouter的方案，则不需要这么做，因为vRouter通常是转控一体的。另外，控制器也可以不依赖于网关节点上的协议栈/vRouter，而是直接与下一跳的物理路由器交互OSPF/BGP，然后通过合适的方式指导vSwitch/TOR进行转发的处理，不过这种做法并不太常见。

当使用分布式路由时，与外界三层对接情况要复杂一些。通常情况下，分布式路由只处理东西向流量，南北向流量仍然需用集中地进行路由和NAT，此时控制器需要将外界的路由反射给各个Hypervisor，以完成南北向流量的引导。还有一种可能的部署，是将所有的服务器都与外界相连，南北向流量的处理也完全分布式地进行。这种部署的优点是消除了南北向流量的处理瓶颈，缺点是难以对流量进行安全的控制，而且会造成额外的布线成本。另外，实现SNAT时往往需要带状态，而OpenFlow/vSwitch目前还没有形成有效的规范去保证带状态处理的性能。

相比OpenFlow，EVPN使用传统的分布式思路解决了网络边缘的上述问题，同样支持VTEP自动发现、ARP抑制、分布式网关、虚拟机迁移等。EVPN多为传统的设备厂商所支持，其结合SDN的方式主要有两种：一种是在SDN控制器上运行BGP，作为RR反射Overlay的路由；另一种是控制器通过NETCONF或其他配置协议或接口，实现EVPN的自动化开通。混合部署EVPN和OpenFlow也是可行的，此时控制器需要完成两种控制方式间的路由重分布，在EVPN路由和OpenFlow流表间进行转换。

3.3.2 网络传输

云数据中心里，Fabric指的是能够为虚拟机间的通信提供高带宽、无阻塞传输的网络。传统数据中心的3-Tier网络，对于Fabric没有明确的需求。云计算兴起后，数据中心的通信模型发生了巨变，前面章节中反复提到过的一些原因，如虚拟机规模的扩张、东西向流量的激增等，使得Fabric成为支撑云数据中心网络的核心技术。高带宽体现在Leaf-Spine设备的线卡由1GE/10GE向40GE/100GE的快速演进，而无阻塞体现在由xSTP转向更为高效的控制逻辑上。

xSTP 是一种有阻塞的协议，网络中的链路利用率极低。为了解决这一问题，以 TRILL 为代表的二层 Fabric 技术最先兴起，TRILL 为二层网络引入了路由的智能，带来了多路径转发（ECMP）的能力。不过，TRILL 的运行需要对传输设备进行升级或者更换，而且协议本身仍不够成熟，跨厂商难于互通，维护起来成本也很高。相比之下，OSPF/BGP 等传统的路由协议则要成熟得多，它们天生地就支持 ECMP，而且部署维护起来也更为方便。不过，三层网络的问题在于不够灵活，虚拟机无法大范围进行二层的迁移。考虑到虚拟机迁移是云数据中心的刚需，而 OSPF/BGP 无法独立地支持大二层，因此云数据中心的第一代 Fabric 技术以 TRILL 为主。

1. Underlay 与 Overlay

不过部署 TRILL 可能会带来产商锁定的问题，人们还是希望可以用更为通用的路由协议来运行 Fabric。于是，以 VxLAN 为代表的隧道技术提出在虚拟机二层流量的外面再封装一层 IP 包头，使得二层虚拟网络得以与 Fabric 解耦，这样一来，只要是支持路由的交换机，就都可以用来传输大二层的流量。VxLAN 在提出后，迅速得到了业界的认可，隧道 +OSPF/BGP 代替 TRILL 成为 Fabric 的新一代技术。不过，隧道技术更多地关注于封装，控制平面往往比较简单，因此 SDN 和隧道技术的结合成为新的趋势。如标准的 VxLAN 只能通过二层的自学习来进行转发，二层的泛洪依赖于 Fabric 对于组播的支持。通过 SDN 控制平面的全局视图指导 vSwitch/TOR 上的 VxLAN 封装，能够有效地抑制泛洪，从而消除 VxLAN 对于 Fabric 组播能力的依赖。

从理论上讲，隧道里面的内容应该对于 Fabric 的传输来说是透明的。不过出于对虚拟网络流量更好的支持，Fabric 需要感知内层包头的一些特征，以方便进行 ECMP 和 QoS 的处理，这就需要提供内层包头字段向外层包头字段映射的机制。VxLAN 使用外层 UDP 源端口号和外层 IP DSCP 字段实现 ECMP 和 QoS，其他隧道协议也各有各的映射规则。不过，这种映射的控制是发生在 vSwitch/TOR 中的，Fabric 使用这些字段也只是服务于传统的路由协议和 DiffServ。ECMP 的处理依赖于 Fabric 设备中的算法，往往是通过 <源 IP，源端口号，目的 IP，目的端口号、协议类型> 的五元组进行哈希。而 QoS 的实现依赖于管理员对 Fabric 设备进行手工的配置。

2. Underlay 的自动化

从上述描述可见，OSPF/BGP+ECMP Fabric 只是作为 VxLAN 等隧道技术的 Underlay，SDN 的控制逻辑（隧道外层包头的封装、标记策略）主要仍发生在网络边缘。也就是说，转发作为 Fabric 的主要逻辑是不依赖于 SDN 控制器的，而依然通过传统的路由协议来支撑。从提供连接的角度来说，传统的路由协议已经做得足够好了，用 SDN 控制器来定义 Fabric 的转发，目前还没有看到明确的场景和需求。但是为了实现网络边缘和 Fabric 的统一管理，简化 Fabric 上的手工配置，很多厂家开始考虑将 Fabric 的管理功能从网管软件转移到 SDN 控制器中，作为云数据中心网络提供单一的管控入口，虽然有别于通过 OpenFlow 对 FIB 直接进行编程，但这仍然可以算得上是广义上的 SDN。例如，在 ZTP

(Zero Touch Provisioning) 技术中，管理员可以预先为 Fabric 编写好脚本，为设备分配好角色、ID 以及 OSPF/BGP 参数，设备加电之后会通过 DHCP Option 获得控制器的地址，从控制器上同步这些配置脚本，并在本地自动加载这些配置，省去了手动对 Fabric 进行初始化的工作。对于一些更为复杂的 Fabric 管理场景，比如涉及事务性的操作，可以通过或者 NETCONF 作为南向的配置协议，或者使用 Puppet、Ansible 等自动化部署工具来完成。

3. Underlay SDN

当然，用 SDN 来定义 Fabric 的转发完全是可以实现的。相比 Internet，数据中心 Fabric 的网络结构和地址规划都是相对简单和固定的，OSPF/BGP 这类分布式路由协议并不会带来明显的优势。相比分布式路由协议，基于 SDN 的 Fabric 可以提供更为精细的流量策略 (QoS)，能实现对拥塞的动态调优 (大象流)，当网络发生故障时有利于提供精确定位和自动修复，当网络进行割接时也能够更加有效地进行路径切换。当然，少数厂商的转发设备上实现了拥塞自适应、隧道 OAM 等特性，不过这些特性需要深度定制化的芯片来提供支持，无法做到跨厂家互通。

如果使用 SDN 来控制 Fabric 的转发，那么网络边缘的隧道技术就变得不是那么必要了，从学术角度来讲，通过在边缘对现有字段的重新规划，足以满足云数据中心对于网络的需求，还可以消除由隧道封装所带来的开销。当然，使用 SDN 来控制 Fabric 并不意味着网络边缘就不能使用隧道。

另外，随着光端口成本的降低和光交换技术的成熟，未来的数据中心网络有可能会向 IP + 光的架构进行演进，那么流量的特征感知、光路的自动开通、IP 光的智能协同，这些都需要 SDN 对网络的传输进行全局的管理与控制才有可能得到实现。

不过，如果从现实的角度来考虑，基于 SDN 的 Fabric 需要对数据中心网络进行较大的改造，推动起来存在着很多的阻碍因素。尽管如此，国外还是有一些 SDN 创业公司选择走上了这条技术路线，希望通过 SDN + 白盒彻底地改造云数据中心网络。相比之下，国内的环境要保守得多，无论是互联网企业还是运营商，都鲜有魄力承担“对网络动大手术”的风险。

3.3.3 服务链

服务链是指，业务流量需要按照特定的业务逻辑有次序地经过特定的服务节点，如从 Internet 进入数据中心访问云中 Web 服务的流量，需要先经过 FW 进行过滤，然后再经过 LB 在多个 Web 服务器实例间进行分流，最后才会发送给某一 Web 服务器实例。传统的数据中心网络中，FW/LB/IPS 均使用专业硬件设备来实现，这些硬件设备的价格十分昂贵，一两个节点就需要负责处理全部的流量，设备的体积通常也很庞大，部署的灵活性较差。在实际部署中，受限传统的路由机制，这些服务节点或者被旁挂在汇聚设备上，或者被串在汇聚设备和核心设备之间。在旁挂的方式中，需要在汇聚设备上手动配置 PBR 去实现引流，使得流量能够按照特定的顺序经过这些设备；在串联的方式中，不需要做额外的配

置，但是服务节点处理流量的压力太大，很可能会成为性能方面的瓶颈。

随着 NFV 的发展，服务节点的形态正逐步从专用硬件向 VNF 转变，形态上的虚拟化使得服务节点的部署不再受物理位置所局限，VNF 可以插入到网络边缘的任意位置，有时还会在不同的位置间进行迁移。因此，传统的路由机制无法满足虚拟化环境中服务链的实现需求，而 SDN 凭借着高度的自动化以及对路由的灵活修饰能力，已经成为数据中心服务链的支撑性技术，而其中流量的分类和引导是 SDN 实现服务链的基础。

1. 流量的分类

流量分类的目的，是识别出来哪些是服务链流量，哪些不属于服务链流量。从概念上来说，流量的分类需要通过专用的流量分类设备来完成，由 SDN 控制器向流量分类设备下发流量分类策略。但在实际中，分类的功能往往集成在转发设备中进行实现，此时分类策略所能达到的粒度也由转发设备来决定。粗放一点的粒度可以是以端口为单位的，以租户为单位的精细一点的粒度往往需要对 L2 ~ L4 中的字段进行组合来识别“流”。当然，也可以在转发的设备上集成 DPI 对流量进行应用层面的分类。

通常来说，流量的分类结果往往是确定的，分类也只需要在服务链的入口设备上进行一次就可以了，但是考虑到某些 VNF 会对流量的字段进行改写，如 NAT 会改写源 IP 地址、LB 可能会改写源/目的 IP 地址，那么经过这些服务节点的改写后，针对原始流量特征的分类结果可能就会失效了，因此就需要控制器向转发设备下发规则对流量重新进行分类。某些 VNF 会对流量进行更为细致的分类，比如 IPS 会根据流量的动态特征分析可能的恶意流量源，据此将黑名单上主机所发出的流量引导到堡垒主机上。这时，也需要 IPS 将分析出来的黑名单告知控制器，控制器向转发设备下发高优先级的分类策略以识别恶意流量。当 IPS 发现攻击已经结束时，黑名单解除，相应的分类策略控制器要及时地清除。

当然，上面所举的 NAT、LB 和 IPS 的例子，都可以看作针对 Per Session 的动态处理。要求控制器能够动态地获取 VNF 的处理结果，并维护 Session 的状态，实现难度较高。相比之下，静态地下发 Per Flow 的分类策略的实现则要简单得多。

2. 流量的引导

有了流量分类作为基础，即可对流量进行引导，使之按需特定的顺序经过 VNF。要实现流量的引导，最为核心的就是对数据进行重新封装。封装的思路无外乎有下面几种：

- 1) 在数据包原有字段间插入新的垫层；
- 2) 使用数据包的保留字段或已有的但不会用到的字段；
- 3) 为数据包封装外层包头；
- 4) 改写现有字段。

封装的字段需要有效地携带服务链转发所需的信息，主要包括以下三类：流量所属的服务链 ID、流量当前在服务链上所处的位置以及下一跳的目的地址。

除了上述三类基本信息外，某些场景下封装还要携带 VNF 的处理结果，以便后续的 VNF 或者转发设备能够根据处理结果调整流量处理或者转发的策略。这些处理结果的信息通常被称为 Context，Context 的定义与 VNF 的处理机制密切相关，因此不同的 VNF 使用

的 Context 的语义可能会非常灵活，这就需要控制器来协调 Context 的分配，同时要求数据平面的封装可以灵活地进行扩展。

流量的引导通常还需要保证上下行流量所经过的路径是对称的，这不仅要求双向经过的 VNF 顺序是对称的，还要求必须经过相同的 VNF 实例，以防某些 VNF 上出现半开连接而将流量丢弃。当 VNF 进行在线迁移时，需要动态地调整流量的引导规则，此时应尽量保证数据包不乱序，并且尽可能地减少由于路径调整造成的业务中断时间。

基于 SDN 的服务链有很多种实现方式。得益于 OVS 在数据中心的普及，基于 OpenFlow 来实现服务链是最为常见的，OpenFlow 在网络边缘可以精确地识别流量，然后通过流表引导流量在不同的 VNF 间进行跳转。其他的实现方式还包括 VLAN Stitching、PBR、BGP VPN、NSH、SR 等。PBR 和 OpenFlow 的原理类似，本质上都属于 ACL，相比于传统方式下在汇聚设备上手配策略路由，控制器可以自动化地配置 PBR。VLAN Stitching 方式中，控制器会为服务链上的每一跳配置专用的 VLAN，通过这些 VLAN 中进行的泛洪，流量就能够自然地流经各个 VNF。BGP VPN 方式中，控制器会为服务链分配专用的 VPN 标识，并根据 VNF 的分布信息集中地修改 BGP 的下跳来实现引流。NSH 为服务链设计了新的包头，引入了新字段 SPI 和 SI 来转发流量 NSH 作为一种数据面的封装，可以配合 OpenFlow 来使用。SR 原生地支持源路由，控制器会根据 VNF 的分布生成 label stack，直接在 header 中指定要经过的各个 VNF。

3. 其他方面的考虑

为了保证 VNF 工作的性能，同时考虑到安全性的问题，不同的服务链可能会被分配不同的 VNF 实例，当某种类型的 VNF 的性能要求较高时，可以分配该类 VNF 的多个实例。为了实现多个 VNF 间的负载均衡，SDN 控制器需要通过一些监测技术来维护这些 VNF 实例的负载状态，同时需要运行一些负载均衡算法来保证多个 VNF 实例的高可用。

在服务链中引入 OAM，主要是对服务链路径的状态进行监测，这一块业界目前还没有形成标准，参考 IETF 相关的 Draft，可以看到对于服务链路径状态的监测包括主要以下几个方面。

- **连通性**。包括可达性、Path MTU、包的乱序和损坏等。
- **连续性**。包括链路故障、路径故障、VNF 故障等，可以通过 BFD 来实现。
- **路径跟踪**。包括路径上逐跳的跟踪，当存在多链路时需要跟踪 ECMP 路径，需要中继设备能够支持对 OAM 的探针进行回复。
- **性能**。包括时延、丢包，需要支持时间戳以及时间同步机制（如 NTP、GPS）。

基于 SDN 实现服务链是一个新兴的技术领域，无论是在工程实现还是学术研究中，目前都存在着很多的空白，这里无法逐一而述。另外，服务链的实现不仅依赖于 SDN 对流量的识别和引导，还依赖于 NFV 对 VNF 的管理与监测。SDN 与 NFV 系统间需要进行接口交互和功能切割，这超出了本书的讨论范围，有兴趣的读者请自行了解。

3.3.4 可视化

传统数据中心里，网络运维一直是件“很痛”的事情。网络是数据中心正常运转的基础，网络状态正常时对业务是透明的，一旦网络出了问题，很多技术团队就会找上门来。小至服务器分不到 IP、Web 访问迟缓，大至数据库连接不稳定，甚至业务的大面积瘫痪，网络运维人员不得不独自对着大量的网络设备进行故障的排查。尽管网络设计已经想尽了一切办法去缩小故障域，但是一个小问题的排查很可能需要花上几个小时的时间，稍有不慎，一条配置出了错，还有可能引发更为严重的问题。有时候花了几个小时去排查网络，但最后发现问题根本就不是出在网络上。在人们的印象中，网络总是“最不靠谱”的，出了问题首先归因于网络也是人之常情。

云数据中心里，虚拟机的数量和网络的规模数倍于传统数据中心，再加上业务对于敏捷性和高可用的要求，网络运维如果不能变得更为智能、自动化，那么一旦发生网络故障，云中的业务可能会受到毁灭性的打击。而且，虚拟交换机的引入会使得网络故障的排查变得更加困难，一方面管理的边界从物理网络延伸进了服务器，另一方面也对网络运维人员的技能提出了新的要求。

传统的网管其实一直就定位于简化网络的运维，不过由于网络一直不够开放，各大厂商设备网管接口区别巨大，网管能采集上来的数据很多，但是难以进行有效的整合，基本上就是出了问题进行告警，并不具备故障的分析、定位和自动纠错的能力。SDN 提出后，网络变得更加标准、开放，控制器通过全局的网络视图就能将部分运维的功能自动化。全局视图的形成依赖于控制器的可视化能力，而可视化能力主要包括网络可视化和流量可视化两个方面。

1. 网络可视化

网络可视化又可分为网元可视化和组网可视化。网元可视化要求控制器能够与设备进行交互，采集设备的运行时数据，如电源、背板、CPU、内存等基础硬件的状态，Up/Down、收发包速率、丢包率等端口状态，以及 MAC 表、BGP 表等转发表项的状态。另外，控制器作为 SDN 中最为关键的网元，其自身的状态、性能、异常记录和管理日志也需要进行全面的维护。

和传统网管做网元可视化的目的不同，SDN 控制器采集到设备运行时的数据，会将其用于控制的反馈和优化，比如通过 buffer 当前的载荷情况来优化传输路径，因此对数据采集接口的实时性要求较高。SNMP 的通用性较强，但是 Polling 的效率低，Push 的功能弱，无法实现实时数据的采集。NETCONF 普遍被看作 SNMP 的代替者，但实际上其优势在于事务性的配置，在监测和可视化方面相比 SNMP 并没有明显的优势，尽管通过 Subscribe/Notification 对 Push 模型进行了增强，但是由于采用了 XML 的数据格式，因此在序列化 / 反序列化性能上存在着明显的瓶颈，同样无法胜任对实时性要求较高的场景。至于 OpenFlow/OVSDB，由于二者对于设备的资源建模侧重于转发方面，对设备本身缺乏足够的描述能力，因此并不适合做网元的可视化。目前，业界正在推动采用 gRPC 来传输实时

数据。gRPC 采用 ProtoBuffer 作为数据格式，序列化 / 反序列化的性能很高，对于 Push 模型的支持也更好。因此，gRPC 已被普遍视为实现 Telemetry 数据采集的标准传输技术，各个主流厂商的设备和一些 SDN 控制器都已经对 gRPC 提供了支持。

组网可视化要求 SDN 控制器能够全面地了解网络的物理结构和逻辑结构。网络的物理结构主要就是指拓扑。拓扑发现有两种思路：一种是由控制器集中式地进行探测和计算，在这种方式下，控制器能够主动地参与到拓扑发现中，但是控制器上的处理压力较大。另一种是由设备运行分布式的协议获取拓扑，然后再以某种方式将拓扑告知控制器，在这种方式下，控制器只能被动学习拓扑，但是控制器上的处理压力得以减小。网络的逻辑结构主要是指 IP 编址和路由，对于虚拟化环境而言，还包括租户标识的分配、虚拟机的位置分布、虚拟机间的通信策略、虚拟机间通信所走的路径等。对于网络中任何一个虚拟机或者任何一个 IP 地址，SDN 控制器应该可以明确地指出其所在的服务器；对于网络中任意两个虚拟机或者任何两个 IP 地址，SDN 控制器应该明确知道二者是否可以通信，如果可以通信，应该明确两者之间通信的逐跳的路径。这不仅要求控制器能够衔接起南北向接口提供的信息，还需要将 Overlay 和 Underlay 进行关联。当底层网络中存在 ECMP 时，控制器需要对哈希算法的结果进行预判，或者通过探针来探测 Underlay 中真实的转发路径。

2. 流量可视化

网络可视化关注的是网络本身，而流量可视化关注的是用户使用网络的行为，如大象流统计、分类流量统计、流量分时走势等。理论上来说，上述功能都可以通过 SDN 控制器集中式地实现，但是考虑到流量的数据量级，往往需要通过部署独立的流量分析软件来实现。有很多的商用 / 开源的产品可供选择，这些软件将处理的结果反馈给 SDN 控制器，有助于实现更为精准的网络控制与优化。

“流”层面的分析可使用传统的 NetFlow、sFlow 来实现，设备会对流量进行采样和统计，然后将结果返回给相应的分析软件，这种实现方式中交换机 / 路由器直接参与到了流量的分析中。gRPC 同样可以用于流量的可视化，可以根据五元组或者其他特征来对流量进行实现采集与推送，实现实时的 Telemetry。不过“流”仅关注的是 L2 ~ L4 的特征，如果要看到应用层面的特征，就需要通过部署深度包检测（Deep Packet Inspection, DPI）来实现了。DPI 的部署思路有两种：一种是在每个服务器上部署一个专门用于 DPI 的虚拟机，通过 vSwitch 将本地的一些流量镜像到这个虚拟机上。这种方式对 vSwitch 要求较低，但是需要为每个服务器额外维护一个虚拟机。另一种思路是直接集成在 vSwitch 中集成 DPI 的引擎，这种方式不需要额外的虚拟机，但是对 vSwitch 的要求非常高。在一般情况下，会采用第一种思路来部署 DPI，但 DPI 的使用需要占用服务器上大量的计算资源，会在一定程度上影响服务器承载虚拟机的能力。

如果想要对流量进行更深层次的分析，如安全威胁诊断或用户行为跟踪，就需要借助专业的分析软件来进行处理了，此时交换机 / 路由器主要负责将流量镜像并导入专业的分析软件中，而并不直接参与流量的分析。导入的方式分为带内和带外两种，带内方式镜像流量与业务流量共存于一张网络中，这就需要 SDN 控制器对它们进行区分，并对镜像流量进

行适当的引导。带内方式的优点是组网简单，缺点是镜像流量会占用业务流量的带宽，需要配合 QoS 来实现业务流量与镜像流量的隔离。相反，带外方式中镜像流量会有独立的网络进行传输，带外方式的优点是镜像流量不会占用业务流量的带宽，缺点是独立组网会带来额外的成本。

对于政府、电信和金融几类行业的数据中心来说，流量可视化的需求非常强烈，能够接受独立为镜像流量组网的成本，因此可以选择通过带外方式进行流量的导出。此类网络的设计既可以选择传统的方式，也可以选择 SDN。使用 SDN 的好处在于可以方便地模拟 NPB (Network Packet Broker) 设备的功能，通过对镜像流量进行汇聚、分类、过滤、负载均衡等操作，将流量有针对性地导出到合适的流量分析软件上，能够有效地提高流量分析软件的工作效率。继网络虚拟化之后，NPB 很有可能成为 SDN 的另一个杀手级应用。

做好了上述的可视化工作，就能够得到网络 / 流量的当前状态和历史数据，这些数据是海量的、细碎的，但是其中蕴含着巨大的价值，为网络的排障和优化提供了基础。网络常见的故障包括网络拥塞、设备 / 链路故障、路由黑洞 / 环路 / 震荡等。通过将这些故障的定位和成因分析返回给控制器的控制逻辑，即可对部分故障进行自动的修复，以实现闭环的自动化运维。进一步地，可以结合大数据和机器学习对原始的数据进行处理，以呈现出更为有价值的信息，比如通过大数据分析出攻击的发生，并反馈给 SDN 控制器丢弃嫌疑流量，或者通过机器学习分析出用户应用的行为模式，增强网络对于所承载的应用的理解，有针对性地优化资源在应用间的分配。

3.3.5 安全

安全从来是数据中心网络的核心问题。传统的数据中心网络内部往往采用分区部署的方式，或者直接进行物理上的隔离，或者通过 VLAN+ACL 进行逻辑上的隔离。数据中心与外界网络的互通则由专业的防火墙设备来完成，这些防火墙通常旁挂在汇聚层，集中地把控着数据中心流量的出入。

云数据中心在负载类型和流量模型上有别于传统数据中心网络，对上述的网络安全模型提出了巨大的挑战。

- 1) 虚拟化打破了数据中心网络的物理边界，物理分区的方式再难适应大二层网络。
- 2) 租户数量和虚拟机规模有了数量级的提升，而且虚拟机还经常发生迁移，因此手动配置 VLAN 和 ACL 将是不可接受的，网络安全同样需要自动化。
- 3) 东西向流量的爆炸使得数据中心内部的安全问题更为明显，集中式部署的防火墙通常会导致东西向流量的路径迂回。
- 4) 硬件防火墙虚拟化能力较弱，难以适应云数据中心对安全的弹性需求。

1. 多维安全模型

云数据中心的网络安全设计需要变得更加立体，从数据中心级别，到租户级别，到子网级别，再到端口级别，为云中的业务提供更为可靠的保障。数据中心级别的安全仍然需要在数据中心入口集中式地设置防火墙，由于硬件防火墙不能满足数据中心对安全的

弹性需求，因此需要结合虚拟化技术来实现防火墙资源的池化。租户级别的安全可以通过 VxLAN 等 Overlay 技术对不同租户进行隔离，防止流量在租户间泄露。子网级别的安全可以通过 vRouter 实现，也可以通过虚拟化的防火墙（vFW）来实现，不同的租户需要分配不同的 vRouter/vFW 实例，以获得灵活性和通信隔离。端口级别的安全通常被称为安全组或者微分段（Micro-Segment），需要与虚拟机进行绑定，可在 vSwitch 上实现，或者将分布式的 vFW 以 inline 的方式串在 vNIC 和 vSwitch 之间，同时要保证端口安全策略随虚拟机进行迁移。对应于微分段，有的厂商还提出了宏分段（Macro-Segment），以实现虚拟机和裸机间东西向流量的安全。

上述安全模型通常需要结合 SDN/NFV 进行实现。vRouter/vFW/vIDS/vIPS 等 VNF 实例本身的实现属于 NFV 技术，并且需要依赖于 DPDK 等加速技术以保证性能。SDN 控制器负责通过 NETCONF 或者 RESTful API 对 VNF 进行自动化的管理配置，包括对安全组规则的下发、安全策略的配置和 Session 的管理，以及将不同租户的流量引导到不同的 VNF 实例中。当然，从 NFV 的视角来看，SDN 控制器所做的这些工作大部分也可以通过 MANO 来实现，这里不去探讨两种视角的区别。

安全组作用于网络边缘，能够实现端口级别的防护，以 Iptables 为主要的实现机制。SDN 控制器可以集中式地分发或配置 Iptables 规则，不过这通常需要在虚拟机和 vSwitch 间额外地引入一跳，导致增加了一个潜在的故障点，而且会对性能造成很大的影响，因此并不是理想的选择。如果 vSwitch 设备支持 OpenFlow（如 OVS），那么可以通过流表来匹配 L2 ~ L4 的字段以实现安全组策略，即相当于传统网络中的 ACL。不过 OpenFlow 流表是无状态的，如果希望通过 OpenFlow 实现有状态的策略，可以将首包 PacketIn 给控制器去维护状态，但是这种方式会对性能造成很大的影响，还可以使用 ovs conntrack 等扩展功能，在 OVS 本地完成状态的维护，但是目前这种方式的实现还不是很成熟。

如果考虑到为 OpenFlow 做硬件卸载，那么 ACL 规则就会被从 vSwitch 上转移到 TOR 上，限于 TCAM 的容量，此时 SDN 控制器需要对 ACL 规则进行适当的聚合，在获得足够灵活性的同时要防止 ACL 规则的溢出。同时，SDN 控制器还需要实时地感知虚拟机的位置，当虚拟机发生迁移时能够及时地将 ACL 规则进行迁移。

VNF 上策略的配置对于 SDN 控制器的问题不大，其粒度也往往在于租户和数据中心级别，因此策略的变化不会非常频繁。Session 的管理则会相对复杂一些，可能需要面临如下的两个挑战。

1) **Session 的同步方式**。如果 SDN 控制器需要实时同步 Session 的状态，那么当 Session 数量很多时，控制信道上的开销会很大。如果 SDN 控制器定期同步 Session 状态，那么 SDN 控制器中全局视图的实时性就会受到影响。OpenFlow 控制器还支持对于 Session 的直接控制，此时 Session 建立的速率以及 OpenFlow 控制器的压力的处理可能就会成为瓶颈。因此，Session 的同步方式要根据实际场景和需求来进行选择。

2) **Session 的备份**。高可用是防火墙以及其他网络高级服务的基本要求，实现高可用是有代价的，不同层面的高可用代价不同。在路由层面解决防火墙高可用性是最为基本的，

其代价较低，但是对于 Session 的控制能力不强，在切换时很可能会导致业务的中断。如果同时和 Session 层面进行防火墙状态的热备份，其可用性会很高，但是实现的代价较高。实现热备份有两种方式：第一种是通过 SDN 控制器在 VNF 实例间进行同步，控制信道压力大；第二种是直接 VNF 实例间进行同步，数据平面带宽的开销会很大。

SDN 只能实现网络层面的安全，对于数据安全问题，只能将流量引入专业的数据分析软件进行处理。传统的数据中心里，需要部署 SPAN 或者 TAP 来实现端口镜像，SPAN 和 TAP 的实现受限于资源和成本，只能部署在少数的关键位置。SDN 可以动态地将镜像点插入网络中的任意位置，为虚拟化环境提供更好的可视性。当数据分析软件完成分析后，可以将分析的结果（如恶意流量的特征）反馈给 SDN 控制器，然后由 SDN 控制器下发安全规则将恶意流量丢弃，实现“数据+应用+网络”的闭环安全防护。另外，随着近几年大数据的发展与普及，SDN 控制器上也得以直接集成一些数据分析的功能，能更好地服务于数据中心安全。

2. 服务链与安全

多租户环境中，不同的租户会获得不同的 VNF 实例。SDN 控制器需要完成流量的引导，即所谓的服务技术。服务链技术通常在 vSwitch 上实现，当 VNF 实例采用集中式部署时，服务链的引流策略只需要部署在少数 vSwitch 上。当 VNF 实例采用分布式部署，服务链的实现将变得非常复杂。

一般而言，通过服务链将一些安全相关的 VNF 串在一起，就可以满足绝大部分的安全需求。如果需要实现更为专业的流量安全策略，数据中心运营商可以与流量清洗中心进行合作，将那些难以区分的流量牵引到专业的清洗中心进行处理。攻击流量被清洗中心过滤掉，安全的流量被回注到数据中心进行服务。流量的牵引可以在数据中心入口通过 OpenFlow 来完成，也可以在城域网入口通过 PBR 直接对可疑流量进行牵引，清洗过后的流量再通过专线或者隧道回注到数据中心。

3. SDN 本身的安全问题

前面概括地说明了 SDN 对于数据中心安全性的提升。不过在引入 SDN 后，SDN 本身的安全就成了其他网络安全的基础。控制器是 SDN 的大脑，其安全问题首当其冲，最基本的要求是在控制器的南向和北向接口上做认证、加密和数据校验，防止非法接入、窃听、重放等攻击形式。控制器上某些操作的执行需要进行鉴权，这要求控制器本身能够支持多租户或者 RBAC (Role-Based Access Control)，同时对一些性关键的操作需要进行审计，以确保其合规性。在网络部署中，如果 SDN 控制器需要暴露在公网上，就需要对其采取更为全面的安全措施，如 DOS 和 DDoS 防护等。

相比 IT 层面的安全问题，SDN 自身的安全性更加值得深入的探讨和研究，尤其是一些新生的网络控制协议。它们在增强了传统网络能力的同时，也会引入很多潜在的安全问题，对于这些问题的研究还仍然停留在学术讨论阶段，目前还没有形成特别成熟的安全防范体系。鉴于 SDN 仍没有进入大规模的商用，因此 SDN 自身的安全问题还并不突出，不过这些问题如果不能及时得到解决，那么 SDN 的大规模落地恐怕就只能纸上谈兵了。

3.3.6 高可用

和安全一样，高可用对于云数据中心网络来说同样是至关重要的。传统网络中广泛使用了各种高可用技术，如实现多网卡绑定的 LACP、实现链路冗余的 xSTP、实现网关冗余的 VRRP、实现多路径的 ECMP、实现数据中心互联的 VPLS、实现故障检测的 BFD 等。总结起来，主要可归为以下几个方面：设备关键组件要有冗余、物理拓扑要有环路保护、路由要快收敛防震荡以及多数据中心间要做双活 / 主备。

1. 网络冗余与监测

传统网络中的冗余机制同样也适用于 SDDCN 的设计。对于一个 SDDCN 的产品来说，应尽可能地多提供高可用的功能，这是在生产环境中落地最为重要的加分项。不过需要明确的是，这些冗余机制并不一定都要通过 SDN 的方式来实现。

- 设备关键组件要有冗余，比如多背板、多主控、多电源、多风扇等，SDN 白盒交换机在设计中需要考虑上述要求。
- 物理拓扑要有环路保护，要求 SDN 控制器能够处理多网卡绑定 (LACP)，并在 Leaf-Spine 间实现负载均衡。在对接传统网络时，还需要能够正确处理 STP 的 BDPU 以防止环路，以及出口 IGP/BGP 以实现多路径。
- 路由要快收敛防震荡，要求 SDN 控制器能够快速发现拓扑变化，计算并开通倒换路径，或者由控制器预先下发逃生路径给设备，发生故障时在本地完成路径切换。
- 多数据中心间要做双活 / 主备，要求 SDN 控制器提供数据中心间的 2 层连接通路，该通路上应该支持 QoS 以区别对待不同的业务，少数场景下还需要在多数据中心间做流量调度。

除了冗余机制的设计外，网络还需要提供实时、全面、准确的监测机制。传统网络中有监测通断 BFD、监测性能的 NQA 等手段，这些手段可以继续云数据中心中使用，设备间交互不同的探针通过网络中不同的状态进行监测，当发生状态异常或者指标波动的时候，设备可以主动上报给 SDN 控制器，然后由 SDN 控制器根据一些网络管理的策略执行对应的动作，以保证网络的高可用。当然，SDN 控制器上也可以集成探针的生成、接收和参数调整等功能。

2. 消除控制器的单点

高可用的天敌是单点，网络中应该尽可能地消除单点。传统数据中心网络中，路由器和防火墙是流量集中经过的地方，最容易成为单点。在 SDDCN 产品的设计中，应尽可能提供分布式路由、分布式 NAT 以及分布式防火墙，消除流量的集中点，避免网络中出现单点故障。

由于 SDN 比传统网络多引入了控制器这一角色，因此还需要考虑控制器的单点问题。对于数据中心而言，控制器通常是部署在 intra-net 的，受到直接攻击的可能性比较小，只要集群部署得当，控制器的可用性是可以有所保障的。集群是个纯软件架构的话题，可以说是老生常谈了，当然，不同控制器对于集群本身的实现是有好有坏的。如果控制器上的

业务允许控制器进行无状态的主备或者负载均衡，那么实现起来较为容易，不过大部分情况下控制器间都需要进行状态同步，此时就需要考虑数据持久化、数据分片、数据一致性、设备控制权主从、脑裂检测等诸多方面的问题。一般来说，控制器可以通过一些成熟的开源分布式产品（如 Zookeeper、ETCD、Consul 等）来实现集群，一揽子地解决集群的问题，但是考虑到集成后的性能以及功能上的定制化，一些控制器需要分解出更为复杂的技术堆栈，甚至另起炉灶自己来“造轮子”，这就需要根据业务需求以及企业自身的技术水平具体地来考虑了。

不过，即使控制器的集群做得再好，也不可能提供 100% 的可用性，因此要考虑在集群整体瘫痪的时候，保证数据平面上的通路仍然能够正常完成流量的转发。实现这个目标有几种主要的思路。

1) 设备立即切换到传统的分布式控制。这种方式实现复杂而且网络的状态不可控。

2) 设备仍然能够按照控制器之前下发的策略继续转发，直到控制器被修复后重新接受新的控制策略。这种方式实现简单但是中间会存在控制的真空期。

3) 将基础类的控制策略保留在设备本地，将优化类的控制策略放在控制器上，控制器出现问题时也不会影响基础的转发。这种方式取了前两种方式的优点，代价是控制器失去了一部分控制的权限。

控制器的高可用还会涉及的另外一个问题，那就是升级。理想情况下，控制器应该能够提供做热补丁的能力，不用重启控制器就可以补掉程序中的 Bug。在实际情况下，控制器也至少应该提供集群的不间断升级能力，即在升级控制器 A 时可以将其控制权临时转移给控制器 B，控制器 A 完成升级后切换控制权，然后再升级控制器 B，A 和 B 都升级完成后再做负载均衡。

实际上，控制平面的高可用也并不是只有 SDN 才会面临的问题，传统网络中对设备主控同样提供了很多高可用机制，比如主备控制引擎的倒换技术，数据平面的不中断转发技术、主控的热升级技术等。这些机制和上述提到的 SDN 控制器的高可用，都是可以一一对应起来的，SDN 控制器在做架构设计时不妨向传统网络多多“取经”。

3. 负载均衡

应用的高可用通常通过负载均衡器（简称 LB）来实现，LB 可以看作 L4 ~ L7 层的转发器，它和防火墙一样属于网络的一部分。对于传统的 Hardware LB 或者 VNF vLB，SDN 控制器可以通过 NETCONF、RESTful API 等方式自动地向下发配置。对于 OpenFlow 交换机，SDN 控制器也可以向其下发流表来模拟一些简单的负载均衡行为。为了避免单点，LB 本身同样也得做高可用，多个 LB 的前端需要有一个 Distributor 来分散 VIP 流量。Distributor 的实现既可以使用传统的 BGP+ECMP，通过路由收敛来保证高可用，也可以使用多台 OpenFlow 交换机组成 Distributor Fabric 来实现。

高可用是个很大的话题，从网络到应用到数据，ICT 的任何层面都对高可用有着无止境的追求。但是，越高的可用性通常也意味着越高的成本，而可用性和成本并不是线性的关系，因此在生产环境中到底需要多少的高可用，还是需要数据中心的管理者在需求和成本

间进行综合考虑。

3.4 本章小结

经过多年的发展与验证，目前 SDN 在数据中心的应用已经形成了明确的市场需求，技术上也已经显现了初步的骨架。本章对 SDDCN 的需求、整体架构和关键技术进行了提炼，其中的一些内容未免有些抽象，读者可以先继续阅读接下来的几章，然后再查看本章所述内容，应该会达到更好的理解效果。

