一. Abstractions for Software-Defined Networks 阅读报告

《Abstractions for Software-Defined Networks》是一篇 2014 年发表于 COMMUNICATIONS OF THE ACM（美国计算机学会通讯）上的文章，Martin Casado 等人回顾了改进 SDN 编程模型和抽象的工作。作者在论文中指出为了充分实现 SDN 的愿景，将需要对网络进行抽象。在 SDN 中，抽象是一种设计工具，它可以简化网络编程，使网络管理员和开发人员能够更容易地实现复杂的网络功能。

接下来我将从**网络范围的结构**、**分布式更新**、**模块化组成**、**虚拟化**、**形式验证**等软件定义网络控制器的关键特性概述该论文。

1. Network-Wide Structures

SDN 的一个主要优点是控制器可以计算**网络范围的结构**，从而提供对网络状态的**全局可见性**，使用分布式算法，为这些结构在控制器之间的一致性提供强有力的保证。在控制分布在大量设备上的传统网络中，维护这些网络范围的结构实际上是不可行的，但是使用它们，许多应用程序的逻辑可以变得简单得多。例如，最短路径路由可以通过在表示拓扑的结构上评估 **Dijkstra** 算法来实现。

考虑维护一棵连接网络中的交换机的生成树的任务：优点是这样的树可以用来转发广播流量，而不会有转发环路的危险，但缺点是设计一个分布式算法来构建和维护生成树是非常困难的——在发生**意外**设备或链路故障等事件时迅速重新收敛到新树。

那么，SDN 是如何利用**网络范围的结构**解决这一问题的呢？大多数 SDN 控制器提供了一套在许多应用中出现的常用功能，如拓扑发现和链路故障检测，并且还维护跟踪网络状态信息的结构，如主机位置、链路容量、流量矩阵等。存储这些信息的数据库通常称为**网络信息库(Network information Base, NIB)**。使用 NIB，生成树的 SDN 实现可以比其分布式实现简单得多:每当拓扑发生变化时，它只需使用 Prim 算法**拓扑计算生成树**，并在**沿树转发**的交换机上安装规则。

2. Distributed updates

首先作者给出了更新的要求：控制器程序指定被推入网络的状态版本，更新子系统保证遍历网络的每个数据包只“看到”状态的一致版本。

也就是说，SDN 控制器管理整个网络，因此它们必须经常在多个交换机上更改规则，也就是**分布式更新**。

两阶段提交策略确保了网络更新的一致性，避免了在更新过程中出现不一致的状态。

（1）准备阶段：控制器修改新配置的转发规则，使其只匹配带有新版本对应标签的数据包，并将其安装在每个交换机上。这样，新的配置就准备好了，但还没有开始处理数据包。

（2）提交阶段：**一旦所有的交换机都准备好了**，控制器就会更新网络外围的规则，用新版本标签标记数据包。这样，新的配置就开始处理数据包了。同时，控制器会从每个交换机上卸载旧的配置。

这种策略可以确保在整个网络更新过程中，所有的数据包都被正确地处理，无论它们是在更新开始前、更新过程中，还是更新完成后发送的。

3.Modular composition:

模块化组成：许多网络程序自然分解为几个模块。提供组合编程接口的控制器使得在模块化组件中轻松指定网络行为的正交方面变得容易。

作者认为：尽管 SDN 控制器被比作“网络操作系统”，但当前的控制器缺乏类似于进程的抽象。相反，大多数控制器允许应用程序不受限制地访问网络中每个交换机上的转发表，这使得以模块化方式编写程序变得困难。

SDN 应用程序通常由路由、广播、监视和访问控制等标准构建块构建而成。然而，大多数 **SDN 控制器缺乏模块化，**迫使程序员在每个新应用程序中从头开始重新实现这些基本服务，而不是简单地从库中获取它们。

因此作者指出了三种**方案**：编程语言抽象、隔离切片、参与式网络。**工具**有 FlowVisor、FortNOX、NetKAT、PANE 控制器等。

4.Virtualization

虚拟化的一个突出例子是 VMware 的网络虚拟化平台（**NSX**）。Pyretic 控制器支持类似的抽象。这些控制器向程序员展示了虚拟和物理级别的相同基本结构——网络拓扑的图，这使得为物理网络编写的程序可以在虚拟级别使用。为了定义一个虚拟网络，程序员需要指定逻辑网络中的元素和物理网络中的元素之间的映射。例如，为了从任意拓扑中创建一个“大交换机”，他们会将物理网络中的所有交换机映射到单个虚拟交换机，并隐藏所有内部链接。

将程序与物理拓扑解耦可以简化应用程序，还可以在不受干扰的情况下在几个不同的程序之间共享网络。

5.Formal verification

形式验证：为了帮助程序员编写正确的程序，一些控制器提供了用于自动检查**形式属性**和在出现意外错误时诊断问题的**工具**。

首先，网络中存在许多关键的不变性，这些需要检测。分为两类，一类是只有在给定网络结构的模型后才能陈述和验证。如连通性、无环性、路由点、带宽，另一类要么完全不依赖于拓扑，要么适用于大类拓扑，如访问控制、主机学习。

通过标准化网络硬件接口，SDN 为开发方法和工具使用静态或动态工具自动检查，这些**工具**正式模拟了网络和控制器的状态。例如，Nelson 等人提出了一种基于 Datalog 的 SDN 编程语言，称为 Flowlog，他们也使用它来编写和验证几个规范属性。因为 Flowlog 被设计为有限状态，所以它适合于自动验证，而不需要复杂的程序员提供的断言。

总结

总的来说， 软件定义网络（SDN）的发展集中在解决架构问题上，以促进网络的演进和丰富的应用程序开发。随着新的软件生态系统的增长，新的基本抽象不断发展。这些抽象利用了在具有较少约束状态分布模型的标准服务器上编写网络控制软件的能力。

二.RFC7426 阅读报告

通过《计算机网络》课程的学习，SDN 具有以下关键特征：基于流的转发，数据层面和控制层面分离、网络控制功能用软件实现、可编程网络。RFC 7426 这

份文件由互联网研究任务组（IRTF）的软件定义网络研究组（SDNRG）于 2015 年发布，旨在解决关于 SDN 的混淆，明确 SDN 架构中的层次结构以及各层如何相互接口。

接下来我将从**SDN 定义**、**SDN 层和架构**、**SDN 模型视图**等方面概述该 RFC。

1.SDN 定义

"软件定义网络（SDN）"是可编程网络范例的一个术语。在 RFC7426 中指出，SDN 是软件应用程序动态地编程单个网络设备，从而控制整个网络的行为。在 RFC7149 中指出，SDN 是一组用于以确定性、动态性和可扩展性的方式设计、交付和操作网络服务的技术。

SDN 的一个关键元素是在转发平面和控制平面之间引入抽象，以便将它们分开，并为应用程序提供必要的手段以编程方式控制网络。目标是利用这种分离以及相关的**可编程性**，以减少复杂性。

2.SDN 层和架构

图 1 以详细的高级示意图的形式总结了 SDN 体系结构抽象。

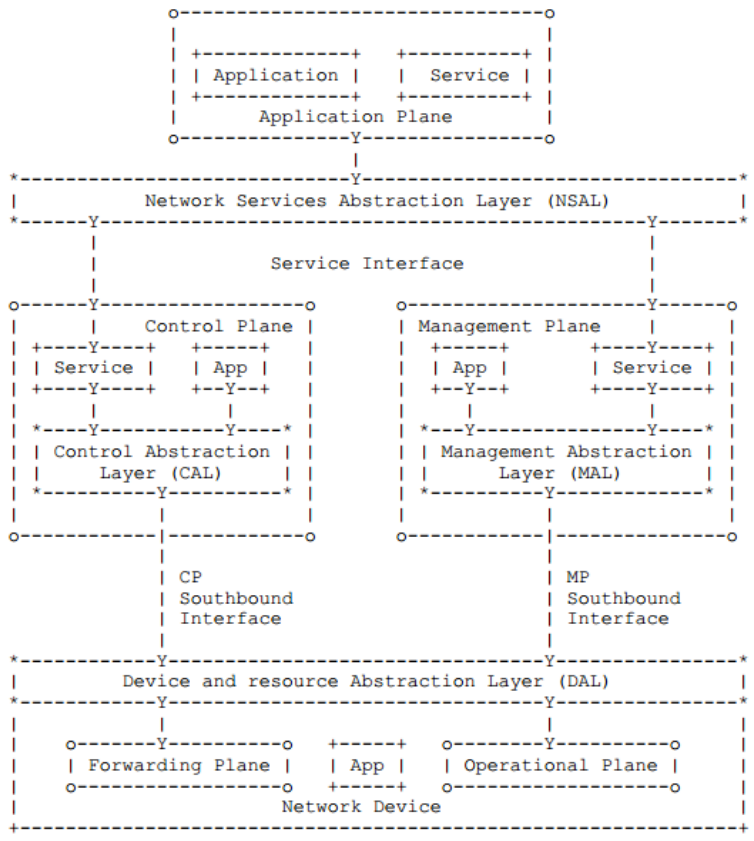

图 1：SDN 层和架构图

自底向上地看，有如下平面

• 转发平面（Forwaring Plane）：负责根据从控制平面接收的指令处理数据路径中的数据包。转发平面的动作包括但不限于转发、丢弃和改变分组。

• 操作平面（Operational Plane）：负责管理网络设备的操作状态，例如，设备是否处于活动状态、可用端口数、每个端口的状态等。

• 控制平面（Control Plane）：负责决定一个或多个网络设备应如何转发数据

包，并将此类决定下推至网络设备执行。

• 管理平面（Management Plane）：负责监控、配置和维护网络设备，例如，就网络设备的状态做出决策。

• 应用程序平面（Application Plane）：定义网络行为的应用程序和服务所在的平面。

在《计算机网络》第四章 PPT 中，SDN 控制器与网络控制应用程序交互的接口称为北向接口，也就是 **NSAL**；通信层与数据层面的接口被称为南向接口。这个接口位于 **DAL**。南向接口允许 SDN 控制器直接与网络设备（如交换机和路由器）进行通信，以便管理和控制网络流量。DAL 提供了一种机制，使得 SDN 控制器可以以一种设备无关的方式与网络设备进行交互。

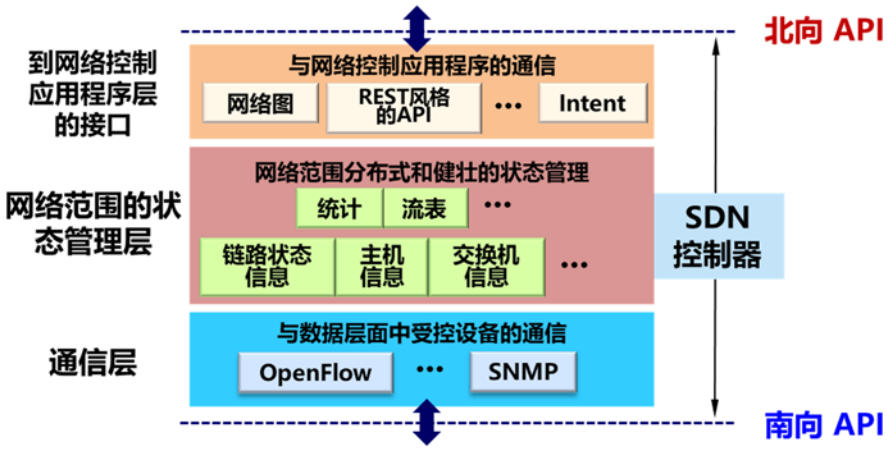

图 2：《计算机网络》第四章 PPT 中 SDN 控制器的结构

3. SDN 模型视图

作者指出，SDN 模型视图有 ForCES、NETCONF/YANG、OpenFlow、Interface to the Routing System、SNMP、PCEP、BFD。下面着重报告 ForCES 和 OpenFlow。

ForCES（Forwarding and Control Element Separation，转发和控制元素分离）是 IETF（Internet Engineering Task Force，互联网工程任务组）的一个框架，由一个模型和两个协议组成。ForCES 通过一个开放的接口，即 ForCES 协议，将转发平面从控制平面中分离出来。该协议操作的是使用 ForCES 模型建模的转发平面的实体。ForCES 使用逻辑功能块（LFBs）来模拟转发平面。

OpenFlow 框架定义了一种协议，通过该协议，逻辑上的集中控制器可以控制 OpenFlow 交换机。每个符合 OpenFlow 的交换机都维护一个或多个流表，这些表用于执行数据包查找。

图 1 中，OpenFlow 交换机规范定义了一个用于**转发平面**以及 **CPSI 的 DAL**。基于 YANG 模型的 OF-CONFIG 协议提供了一个用于 OpenFlow 交换机的转发和操作平面的 DAL，并指定 **NETCONF 作为 MPSI**。从而实现了作者的主张——SDN 南向接口应包括 CPSI 和 MPSI。

总结

总的来说，网络元素视为资源的组合。每个网络元素都有一个负责处理数据路径中的数据包的转发平面（FP），以及一个负责管理设备操作状态的操作平面（OP）。正如图 1 所示，网络元素中的资源由设备和资源抽象层（DAL）抽象出来，由属于控制或管理平面的服务或应用程序进行控制和管理。控制平面（CP）

负责决定如何转发数据包。管理平面（MP）负责监控、配置和维护网络设备。服务接口由网络服务抽象层（NSAL）抽象出来，其他网络应用程序或服务可以使用它们。