一、背景介绍

1. 起源介绍

早期研究人员提出了转发和控制元件分离（**ForCES）框架**，以允许使用灵活的建模语言对流表和流逻辑实现流级网络可编程性。

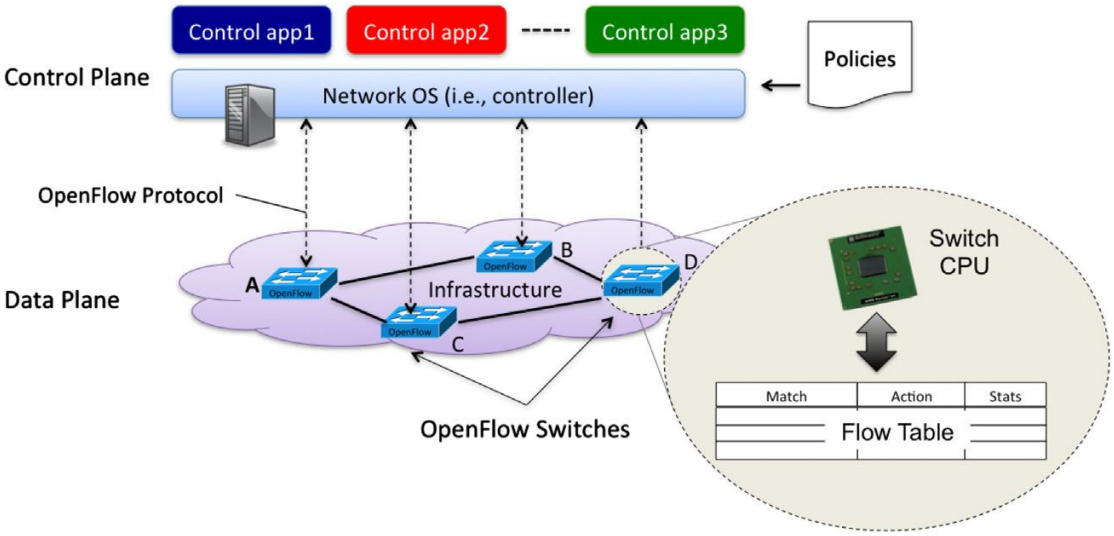
OpenFlow 和 SDN 架构形成于 **2007 年**，继承了 ForCES、SANE、Ethane 和 4D 等项目（主张**数据和控制分离**，以及控制和管理逻辑集中化)。Ethane 在 2006 年使用这种方法被成功应用于需要复杂政策的校园企业网络访问控制，是一次伟大的成功实践。

2. SDN 早期参考设计

在 SDN 早期参考设计中，**数据平面**和**控制平面**被解耦。

• 数据平面：由**简单交换机和控制处理器**组成。交换机被抽象化为一组流表条目，用于决定如何转发匹配一个或多个流表条目的入站数据包；控制处理器提供与控制平面的接口并管理交换机上的流表。

• 控制平面：使用 OpenFlow 协议来编程数据平面并监控数据平面的状态，以网络策略(Policies)作为输入。



2.1 流处理逻辑

当数据包到达交换机时，交换机检查其流表中是否存在与数据包的头字段匹配的流条目：

• 如果存在，则根据该条目进行转发；

• 如果，交换不存在，则：

交换机机生成一个 pauv_in 消息→使用 OpenFlow 协议发送到控制器→控制器根据编程的策略将分组事件传递给适当的控制应用程序。

2.2 控制平面

• 控制平面可以通过被动(新流到达后才响应) 和 主动方式（流到达前插入流条目) 来处理网络流量

• 使用 OpenFlow 协议、**stats_请求算法**和**拓扑发现算法**对网络进行动态控制。

2.3 拓扑发现(discovery)

• 控制平面可以集中式管理大量的交换机集合构建 openFlow 岛(islands),

• 控制器使用 packet_out 消息(message)来指示数据平面从交换机端口发送 LLDP 包到网络链路。

• 控制平面可以根据 LLDP 包_ins 推断岛内的连接性并构建网络拓扑。

2.4 存在的不足

• 控制操作的额外延迟：数据平面和控制平面的分离会导致控制操作的额外延迟的性能损失，特别是对于流设置、拓扑发现和故障恢复。

• 控制平面的（逻辑）集中需要加强对可伸缩性和单点故障的关注。

3. 本文 SDN 部署的主要目标

通过 SDN 的部署，作者主要希望实现以下目标：

• 展示 SDN 架构的通用性及其促进网络创新的能力。

• 在校园生产网络上实现大规模实验。

• 在同一物理 SDN 基础设施上使用切片和虚拟化实现多个并行实验。

• 为 OpenFlow 总体体系的规范过程提供参考意见。

4. SDN 部署的可行性和未来展望

现代服务器可以为整个校园的网络控制和管理提供必要的性能，在本文作者的部署中，2010 年的老式服务器就可以充当 SDN 的控制平面。但是当时大多数商业交换机支持的流表条目数量在短期仍然有局限性，未来的 OpenFlow 交换机必须在硬件中支持大量的流表项，以实现高性能和大规模应用。

使用商用 OpenFlow 交换机和强大的 SDN 控制器是促进 SDN 发展广泛应用的关键。在本文中，作者建议创建一个 Enterprise-GENI 工具包，帮助其他校园在未来可以拷贝 SDN 部署，这有助于推动研究领域和教育界中 SDN 的发展。

二、SDN 部署的四个阶段

第一阶段：概念验证(Proof of concept)

(1) 阶段目标

通过第一阶段，作者希望达到以下三个目标：

1) 使用 openflow 的交换机和控制器的第一个原型来**创建一个小型的 SDN 部署；**

2) 为 OpenFlow 规范过程和提供 OpenFlow 交换机的商业供应商**提供参考意见**；

3) 让更多的研究人员、网络运营商和供应商**参与到 SDN 的部署之中**。

(2) 关键模块

为实现第一阶段的目标并完成实验，作者需要以下两个关键模块作为支撑：

1) **2008 年 OpenFlow** 许可下发布的仅用于研究和实验的 OpenFlow 参考实现;

2) 当时唯一可用的控制器的 **NOX**。NOX0.4 版本提供了一个基本的平台，用于解析 OpenFlow 消息和生成要处理的不同应用程序模块的事件。

(3) 基础设施构建：

作者在斯坦福大学实验室建立了一个**小型 SDN/OpenFlow 网络测试平台**，该网络采用了一种混合模型，其中 OpenFlow VLAN 和传统 VLAN 共存，设置控制器用于管理网络流量。

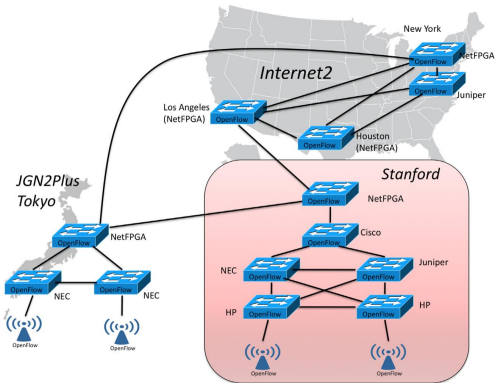
1) 交换机构建

·首先将三个 NetFPGA 机顶盒部署在 Internet2 网络上作为 OpenFlow 交换机，并同时部署了 Juniper 的 OpenFlow-enabled MX 交换机，并创建 SDN 岛；

·在东京，部署了 NEC 的 OpenFlow 交换机和 NetFPGA-based OpenFlow 交换机，并建 SDN 岛；

2) 跨区连接

通过点对点隧道将这些 OpenFlow 岛屿连接起来，创建了一个跨区的 OpenFlow 网络。



(4) 关键实验：

1) 基本思路

利用 SDN 的两个关键特性【**OpenFlow 的层独立性**以及 **OpenFlow 控制器对全球网络地图的可见性**】实现跨 L2 和 L3 网络边界的虚拟机（VM）的无缝移动性：

作者演示了具有**移动虚拟机的多人游戏应用程序的虚拟机移动能力**，以支持移动玩家而不中断游戏。一种允许在斯坦福内部的虚拟机移动，另一种允许跨在斯坦福和日本之间的广域网络（WAN）移动，展示了**由 SDN 支持的 VM 移动性**。

2) 实验结果

能够在游戏客户端和后端服务器之间提供最小的延迟，或者通过最少的跳跃次数路由游戏流量，或者通过将托管游戏服务器的虚拟机移到客户端附近。

(5) 获得的经验：

1. 性能和 SDN 设计

• 短期流时间开销：每个流量需要在开放流网络中等待持续时间，对于短期的流动来说，这可能是一个重大的开销。

• CPU 性能不足：当时交换机的 cpu 控制操作性能低，无法处理大量的 OpenFlow 消息。流设置的 stats_请求和响应模式会对 CPU 造成压力，导致流设置时间增加和控制平面消息的总体响应时间较差，甚至还会导致网络中断。

2. 与旧式网络并存(Living with legacy networks)

• 使用 NetFPGA 封装数据平面通信作为隧道传输的一部分，有时会导致数据包大小超过中间路由器上的 MTU，从而**导致数据包下降**。

--解决方案：创建一个基于软件的隧道机制，使用以太网封装和利用 IP 分片来规避 MTU 问题，但会导致性能降低。

• 通过在 OpenFlow 域中使用一个非标准的 LLDP 目标地址来减轻中间旧式(legacy)交换机可能会删除 LLDP 数据包的问题。

• 默认启用的 TCP Nagle 算法干扰了 OpenFlow switch 和控制器之间的消息交换，使用 **TCP 套接字选项进行禁用**得以解决。

第二阶段： SDN 部署中的切片和缩放(Slicing and scaling SDN deployment)

(1) 阶段目标

通过第二阶段，作者希望达到以下四个目标：

1. **在同一物理基础设施上实现并发实验；**
2. **与生产流量共存，以验证可以在日常网络中进行实验；**
3. **评估实际部署设置中 SDN 架构和组件的性能；**
4. **改进可用于 SDN 部署的软件堆栈和工具。**

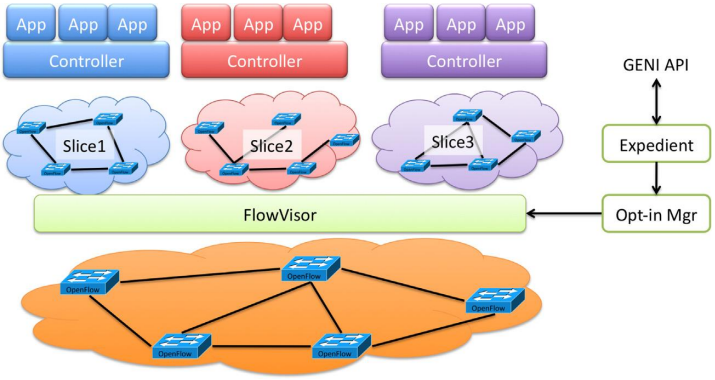
(2) 关键模块

1) 控制平面引入网络虚拟化层

该层使用 FlowVisor 实现了 SDN 网络切片，每个切片通常由不同的 OpenFlow 控制器管理，相互之间互不影响，以支持在同一物理基础设施上的多个并发实验以及生产流量。

2) 引入 NOX0.4 和 SNAC0.4 控制器

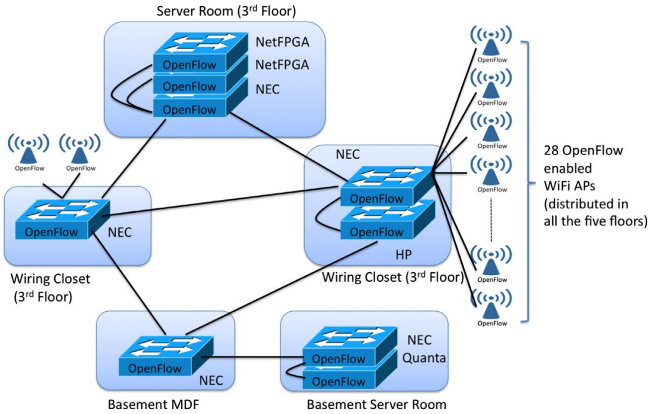
SDN 软件堆栈的其余部分包括用于实验/研究的 NOX0.4 版本和用于生产 OpenFlow 流量的 SNAC0.4 的控制器，使网络管理员能简单地为生产 OpenFlow 流量指定访问控制策略。



(3) 基础设施构建

1) **使用 NEC、HP 和 NetFPGA 交换机**，采用 OpenFlow 版本 0.8.9 的新固件。

2) **使用 VLANs**：NEC 和 HP 交换机能够在一个物理交换机上创建多个 OpenFlow 交换机的实例，模拟更大的拓扑结构。



(4) **关键实验**

实验重点在于探讨 SDN 基础设施中切片技术如何支持多个并发切片，每个切片可以用于研究实验、承载生产 OpenFlow 流量或传统生产流量。通过 FlowVisor 虚拟化基础设施，支持在隔离的切片内进行实验，每个切片由不同的基于 NOX 的控制器管理。

作者主要进行了 Web 负载均衡、移动设备利用多个蜂窝和 Wi-Fi 网络的用户体验优化，以及 OpenFlow 对流定义的灵活动态聚合等实验。所有研究实验都以简单的 SDN 软件应用程序实现，验证了 SDN 的创新能力和轻松创建新网络功能的可行性。

(5) 获得的经验：

• 流空间切片的灵活性：

FlowVisor 版本 0.4 通过为每个切片添加软件速率限制来解决工作负载隔离问题，但当切片数量超过可用数据平面队列时，实现数据平面和控制平面的隔离仍然具有挑战性。

• 重叠流空间的问题：

FlowVisor 版本 0.4 和 0.6 允许流空间重叠，这可能是一种优势，也可能是一个陷阱。

第三阶段：端到端的国家部署(End-to-end deployment with a national Footprint)

(1) 阶段目标

1. 帮助校园、GPO、NLR 和 Internet2 能够**以较小代价部署 SDN**；
2. 协助校园研究人员**在本地和国家级 SDN 基础设施上进行实验**和演示；
3. 创建一个具有切片和 GENI 接口的端到端 SDN 基础设施，在同一物理基础设施上**进行多个并发切片的实验**；
4. 协助**完善 OpenFlow 和 SDN 组件**，促进 SDN 系统的发展；

(2) 关键模块

本阶段使用 OpenFlow 版本 1.0，引入了基于队列的切片等功能。

SDN 堆栈包括 OpenFlow 可用交换机、FlowVisor 用于切片、NOX 作为控制器，以及 Expedient 和 Opt-in Manager 等附加组件。

(3) 基础设施构建

• 采用**可虚拟化的计算节点**，构建了可切片和可编程的计算和网络设施，将全球 11 个独立 SDN 岛屿连接形成整体。

• 初始阶段采用**星形拓扑**将校园的 SDN 部署与 NLR 相互连接；

(4) 关键实验

• **Aster*x 实验：**基于先前的 Plug-N-Serve 系统，旨在研究负载感知服务器和路径选择的不同算法。

• **Pathlet 实验：**通过 OpenFlow 控制器实现，演示了边缘设备如何在面对网络动态时有效地利用多路径路由的灵活性。

• **OpenRoads 实验：**展示了如果设备允许利用其周围的多个无线网络，通过 SDN 实现 AP 之间的无缝切换和在 Wi-Fi 和 WiMAX 网络上进行三路播放视频流，可以实现实时视频的出色用户体验。

• **SmartRE 演示：**展示了 SmartRE 框架如何通过消除传输中的冗余内容使服务和网络容量都能够得到扩展。

(5) 获得的经验：

1. 全国范围 SDN 实验和演示

• 将各个独立的岛屿拼接成一个大规模网络是一项复杂的任务；

• 为每个校园提供 SDN 技术性的指导、能选择不同的方法来更新和维护最新的软件以提高校园 SDN 的灵活性；

• 要使所有岛屿运行相关版本的软件堆栈和关键组件过程十分复杂。

• 在进行大规模实验之前，需要进行 Plastic Slices 项目的调试，以提高稳定性。

2. **网络运营维护**

• 基础设施规模大，稳定性压力大，需控制流表入口限制，避免交换机负载过高。

• 控制器位置重要，需考虑性能和可靠性，避免流空间重叠和环路风险。

• 跨遗留网络挑战大，需处理 MAC 学习和拓扑发现错误。

• 环路风险大，在与遗留网络相互连接的 OpenFlow 网络中引入环路需谨慎。

第四阶段：生产部署(Production deployment)

(1) 阶段目标

1. 将 OpenFlow 和 SDN 接近生产状态(ready)
2. 演示 SDN 切片可同时支持研究和生产。

(2) 实施计划

• 创建 Wi-Fi 网络

• 构建生产 SDN 基础设施

• 实施全面的测量和调试基础设施，并与软硬件供应商合作。

(3) **实验执行过程**

团队在 Gates 大楼使用 SDN 进行日常生产，先在 ofwifi 网络测试，再在有线团队网络中使用 NEC、HP、NetFPGA 和 Pronto 交换机，持续监测并与网络管理员分享经验。

结果：经过两年演进，CIS 大楼网络管理员在生产网络中启用 OpenFlow 功能，标志着 SDN 在 Stanford 的有限生产部署中取得成功。现有的网络结构包括 30 个 OpenFlow 能力的 Wi-Fi 接入点，并在 CIS 大楼的 6 个运行，由一个控制器控制。

(4) 获得的经验：

1. **避免混合使用生成树协议 STP 和 OpenFlow：** 在生产环境中同时使用 Spanning Tree Protocol（STP）和 OpenFlow 可能导致不可预测的结果，因为 OpenFlow 可能无法意识到被 STP 阻塞的端口。

2. **OpenFlow 交换机的硬件限制：**如流表大小和流入速率，可能需要采用一些解决方案，如通配流匹配头字段和扩展流表大小进行解决。

三、SDN 部署后提出的建议

在以上 SDN 实验部署后，作者对 OpenFlow 的说明规范提出了一些中肯的建议，如：

- 允许更快地替换和修改现有流表规则，而无需删除和添加；
- 允许匹配由 ping 生成的 ICMP 数据包，以便控制应用程序可以将 ICMP 请求与 ICMP 回复进行区分；
- 允许 OpenFlow 网络在主控制器失败或不可达时切换到备用控制器；
- 紧急流缓存：在控制连接中断时插入规则的可能性，以代替正常规则.....

此外，作者的实验为 SDN 架构和其组件的演进提供了有益的参考(input)：这包括对 SDN 架构的影响，对 FlowVisor 设计和实现的影响。作者还提出了一些功能请求，如 FlowVisor 规则爆炸问题，与整个垂直栈的一致状态问题以及对控制器开发者的反馈。总之，作者的工作为 SDN 的发展做出了突出贡献。

四、总结

本篇文章是在 SDN 早期发展阶段撰写的，着重介绍了 SDN 在大学校园的早期部署，通过实践展示了 SDN 的潜力，为 SDN 架构的演进提供了重要参考依据。此外，通过本篇文章也可以让我们更好了解 SDN 的早期发展历程，了解其背后的关键技术。