Network Virtualization and Software Defined Networking for Cloud Computing: A Survey

Raj Jain and Subharthi Paul, Washington University

ABSTRACT

Network virtualization is the key to the current and future success of cloud computing. In this article, we explain key reasons for virtualization and briefly explain several of the networking technologies that have been developed recently or are being developed in various standards bodies. In particular, we explain software defined networking, which is the key to network programmability. We also illustrate SDN's applicability with our own research on OpenADN — application delivery in a multi-cloud environment.

INTRODUCTION

The Internet has resulted in virtualization of all aspects of our life. Today, our workplaces are virtual, we shop virtually, get virtual education, entertainment is all virtual, and of course, much of our computing is virtual. The key enabler for all virtualizations is the Internet and various computer networking technologies. It turns out that computer networking itself has to be virtualized. Several new standards and technologies have been developed for network virtualization. This article is a survey of these technologies.

WHY VIRTUALIZE?

There are many reasons why we need to virtualize resources. The five most common reasons are:

1 **Sharing:** When a resource is too big for a single user, it is best to divide it into multiple virtual pieces, as is the case with today's multi-core processors. Each processor can run multiple virtual machines (VMs), and each machine can be used by a different user. The same applies to high-speed links and large-capacity disks.

2 **Isolation:** Multiple users sharing a resource may not trust each other, so it is important to provide isolation among users. Users using one virtual component should not be able to monitor the activities or interfere with the activities of other users. This may apply even if different users belong to the same organization since different departments of the organization (e.g., finance and engineering) may have data that is confidential to the department.

3 **Aggregation:** If the resource is too small, it is possible to construct a large virtual resource that behaves like a large resource. This is the case with storage, where a large number of inexpensive unreliable disks can be used to make up large reliable storage.

4 **Dynamics:** Often resource requirements change fast due to user mobility, and a way to reallocate the resource quickly is required. This is easier with virtual resources than with physical resources.

5 **Ease of management:** Last but probably the most important reason for virtualization is the ease of management. Virtual devices are easier to manage because they are software-based and expose a uniform interface through standard abstractions.

VIRTUALIZATION IN COMPUTING

Virtualization is not a new concept to computer scientists. Memory was the first among the computer components to be virtualized. Memory was an expensive part of the original computers,so virtual memory concepts were developed in the 1970s. Study and comparison of various page replacement algorithms was a popular research topic then. Today's computers have very sophisticated and multiple levels of caching for memory. Storage virtualization was a natural next step with virtual disks, virtual compact disk (CD) drives, leading to cloud storage today. Virtualization of desktops resulted in thin clients, which resulted in significant reduction of capital as well as operational expenditure, eventually leading to virtualization of servers and cloud computing.

Computer networking is the plumbing of computing, and like plumbing in all beautiful buildings, networking is the key to many of the features offered by new computing architectures. Virtualization in networking is also not a new concept. Virtual channels in X.25-based telecommunication networks and all subsequent networks allow multiple users to share a large physical channel. Virtual local area networks (VLANs) allow multiple departments of a company to share a physical LAN with isolation. Similarly, virtual private networks (VPNs) allow

This work was supported in part by a grant from Cisco University Research Program and NSF CISE Grant #1249681.

0163-6804/13/$25.00 © 2013 IEEE

companies and employees to use public networks with the same level of security they enjoy in their private networks.

However, there has been significant renewed interest in network virtualization fueled primarily by cloud computing. Several new standards have been developed and are being developed. Software defined networking (SDN) also helps in network virtualization. These recent standards and SDN are the topics of this article.

We discuss several recent network virtualization technologies. Software defined networking is discussed in detail. Our own research on open application delivery using SDN is described. Finally, a summary follows.

Network Virtualization

A computer network starts with a network interface card (NIC) in the host, which is connected to a layer 2 (L2) network (Ethernet, WiFi, etc.) segments. Several L2 network segments may be interconnected via switches (a.k.a. bridges) to form an L2 network, which is one subnet in a layer 3 (L3) network (IPv4 or IPv6). Multiple L3 networks are connected via routers (a.k.a. gateways) to form the Internet. A single data center may have several L2/L3 networks. Several data centers may be interconnected via L2/L3 switches. Each of these network components — NIC, L2 network, L2 switch, L3 networks, L3 routers, data centers, and the Internet — needs to be virtualized. There are multiple, often competing, standards for virtualization of several of these components. Several new ones are being developed.

When a VM moves from one subnet to another, its IP address must change, which complicates routing. It is well known that IP addresses are both locators and system identifiers, so when a system moves, its L3 identifier changes. In spite of all the developments of mobile IP, it is significantly simpler to move systems within one subnet (within one L2 domain) than between subnets. This is because the IEEE 802 addresses used in L2 networks (both Ethernet and WiFi) are system identifiers (not locators) and do not change when a system moves. Therefore, when a network connection spans multiple L2 networks via L3 routers, it is often desirable to create a virtual L2 network that spans the entire network. In a loose sense, several IP networks together appear as one Ethernet network.

Although discussions of providing computing as a utility have been around for quite some time, the real physical implementation of cloud computing came when Amazon announced Elastic Computing 2 (EC2) on August 25, 2006. The (unverified) folklore is that when Amazon's CEO visited the company data center, he was amazed by the number of computers. Since data centers, like most other computing facilities, are designed to avoid crashes when overloaded, the normal utilization of systems is low. The Amazon CEO therefore asked to figure out a way to manage the hardware in a programmatic manner where all the management could be done easily remotely using application programming interfaces (APIs). This allowed them to rent out the unused capacity; so began the computer rental business we now call cloud computing. The concept was immediately successful since it relieved customers of all the headaches of managing equipment that has to be continuously updated to keep up with the latest technologies. Sharing an underutilized resource is good for cloud service customers as well as for the cloud service providers.

Sidebar 1. *Genesis of cloud computing.*

Virtualization of NICs

Each computer system needs at least one L2 NIC (Ethernet card) for communication. Therefore, each physical system has at least one physical NIC. However, if we run multiple VMs on the system, each VM needs its own virtual NIC. As shown in Fig. 1, one way to solve this problem is for the "hypervisor" software that provides processor virtualization also implements as many virtual NICs (vNICs) as there are VMs. These vNICs are interconnected via a virtual switch (vSwitch) which is connected to the physical NIC (pNIC). Multiple pNICs are connected to a physical switch (pSwitch). We use this notation of using p-prefix for physical and v-prefix for virtual objects. In the figures, virtual objects are shown by dotted lines, while physical objects are shown by solid lines.

Virtualization of the NIC may seem straightforward. However, there is significant industry

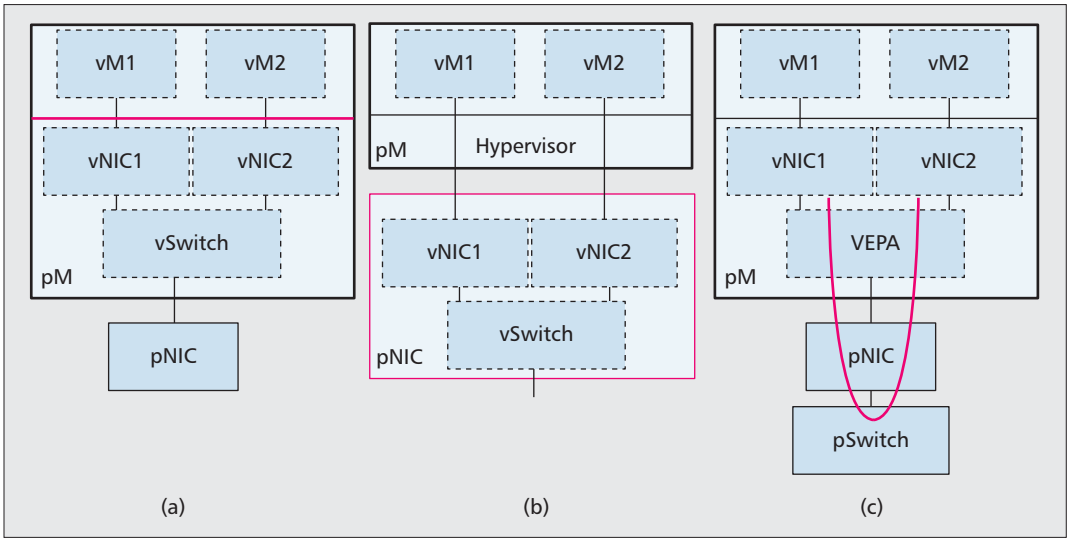

Figure 1. *Three approaches to NIC virtualization.*

The second recent development that is partly responsible for the growth of cloud computing and is fueling a need for networking innovations is smart phone apps. On June 29, 2007, Apple announced the iPhone with the associated app store. Although there were several generations of smart phones before then, the app store was a marketing innovation that changed the landscape for application developers. Today, all businesses including banks, retail stores, and service providers have their own apps, and each of these apps needs to serve a global audience. Cloud computing provides an easy way for these application service providers to obtain computing services worldwide. However, networking features required for application partitioning over multiple clouds owned by multiple cloud service providers are still lacking. Hence, there is a need for virtualization of the Internet, as discussed further in this article.

Sidebar 2. *Growth of cloud computing.*

competition. Different segments of the networking industry have come up with competing standards. Figure 1 shows three different approaches. The first approach, providing a software vNIC via hypervisor, is the one proposed by VM software vendors. This virtual Ethernet bridge (VEB) approach has the virtue of being transparent and straightforward. Its opponents point out that there is significant software overhead, and vNICs may not be easily manageable by external network management software. Also, vNICs may not provide all the features today's pNICs provide. So pNIC vendors (or pNIC chip vendors) have their own solution, which provides virtual NIC ports using single-route I/O virtualization (SR-IOV) on the peripheral-component interconnect (PCI) bus [1]. The switch vendors (or pSwitch chip vendors) have yet another set of solutions that provide virtual channels for inter-VM communication using a virtual Ethernet port aggregator (VEPA), which passes the frames simply to an external switch that implements inter-VM communication policies and reflects some traffic back to other VMs in the same machine. IEEE 802.1Qbg [2] specifies both VEB and VEPA.

Virtualization of Switches

A typical Ethernet switch has 32–128 ports. The number of physical machines that need to be connected on an L2 network is typically much larger than this. Therefore, several layers of switches need to be used to form an L2 network. IEEE Bridge Port Extension standard 802.1BR [3], shown in Fig. 2, allows forming a virtual bridge with a large number of ports using port extenders that are simple relays and may be physical or virtual (like a vSwitch).

Virtual LANs in Clouds

One additional problem in the cloud environment is that multiple VMs in a single physical machine may belong to different clients and thus need to be in different virtual LANs (VLANs). As discussed earlier, each of these VLANs may span several data centers interconnected via L3 networks, as shown in Fig. 3.

Again, there are a number of competing proposals to solve this problem. VMware and several partner companies have proposed virtual extensible LANs (VXLANs) [4]. Network virtualization using generic routing encapsulation (NVGRE) [5] and the Stateless Transport Tunneling (STT) protocol [6] are two other proposals being considered in the Network Virtualization over L3 (NVO3) working group of the Internet Engineering Task Force (IETF).

Virtualization for Multi-Site Data Centers

If a company has multiple data centers located in different parts of a city, it may want to be able to move its VMs anywhere in these data centers quickly and easily. That is, it may want all its VMs to be connected to a single virtual Ethernet spanning all these data centers. Again, a medium access control (MAC) over IP approach like the ones proposed earlier may be used. Transparent Interconnection of Lots of Links (TRILL) [8], which was developed to allow a virtual LAN to span a large campus network, can also be used for this.

Network Function Virtualization

Standard multi-core processors are now so fast that it is possible to design networking devices using software modules that run on standard processors. By combining many different functional modules, any networking device — L2 switch, L3 router, application delivery controller, and so on — can be composed cost effectively and with acceptable performance. The Network Function Virtualization (NFV) group of the European Telecommunications Standards Institute (ETSI) is working on developing standards to enable this [9].

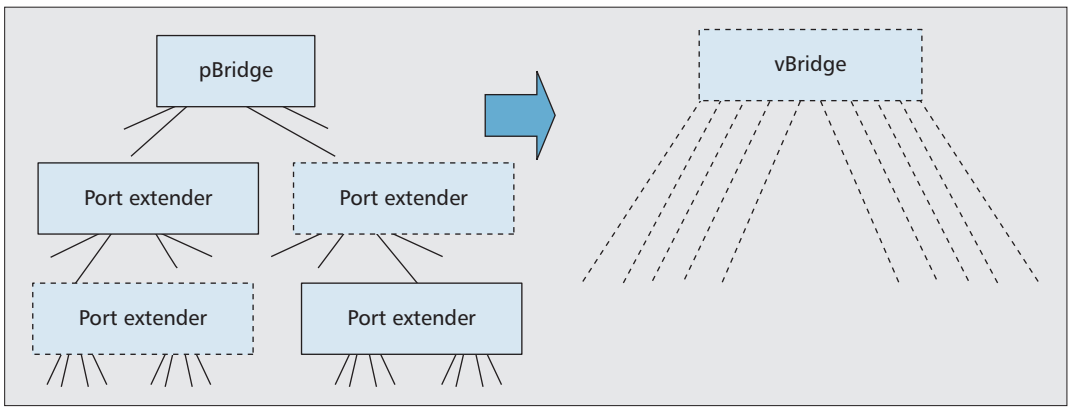

Figure 2. *IEEE 802.1BR bridge port extension.*

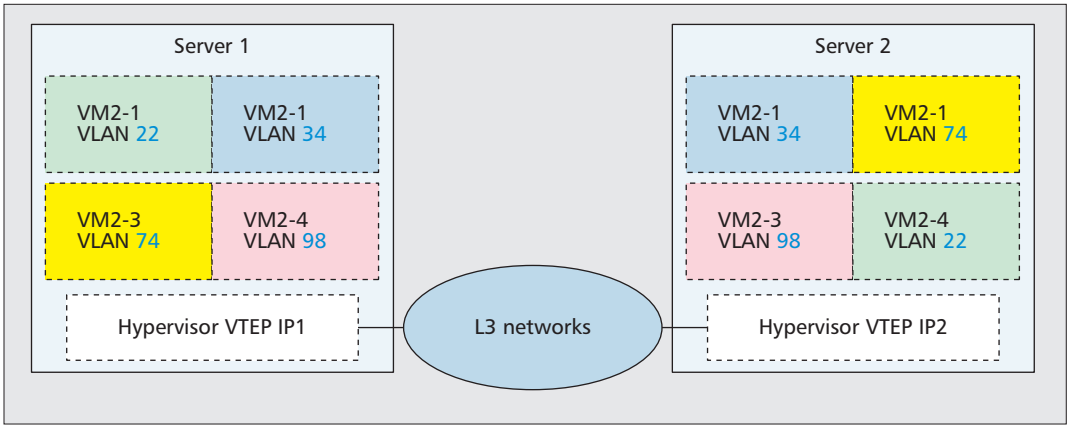

Figure 3. *Different virtual machines may be in different VLANs.*

Software defined networking is the latest revolution in networking innovations. All components of networking industry, including network equipment vendors, Internet service providers, cloud service providers, and users, are working on or looking forward to various aspects of SDN.

Software Defined Networking

Software defined networking is the latest revolution in networking innovations. All components of the networking industry, including network equipment vendors, Internet service providers, cloud service providers, and users, are working on or looking forward to various aspects of SDN. This section provides an overview of SDN.

SDN consists of four innovations:

1. Separation of the control and data planes
2. Centralization of the control plane
3. Programmability of the control plane
4. Standardization of application programming interfaces (APIs)

Each of these innovations is explained briefly below.

Separation of Control and Data Plane

Networking protocols are often arranged in three planes: data, control, and management. The data plane consists of all the messages that are generated by the users. To transport these messages, the network needs to do some housekeeping work, such as finding the shortest path using L3 routing protocols such as Open Shortest Path First (OSPF) or L2 forwarding protocols such as Spanning Tree. The messages used for this purpose are called control messages and are essential for network operation. In addition, the network manager may want to keep track of traffic statistics and the state of various networking equipment. This is done via network management. Management, although important, is different from control in that it is optional and is often not done for small networks such as home networks.

One of the key innovations of SDN is that the control should be separated from the data plane. The data plane consists of forwarding the packets using the forwarding tables prepared by the control plane. The control logic is separated and implemented in a controller that prepares the forwarding table. The switches implement data plane (forwarding) logic that is greatly simplified. This reduces the complexity and cost of the switches significantly.

Centralization of the Control Plane

The U.S. Department of Defense funded Advanced Research Project Agency Network (ARPAnet) research in the early 1960s to counter the threat that the entire nationwide communication system could be disrupted if the telecommunication centers, which were highly centralized and owned by a single company at that time, were to be attacked. ARPAnet researchers therefore came up with a totally distributed architecture in which the communication continues and packets find the path (if one exists) even if many of the routers become nonoperational. Both the data and control planes were totally distributed. For example, each router participates in helping prepare the routing tables. Routers exchange reachability information with their neighbors and neighbors' neighbors, and so on. This distributed control paradigm was one of the pillars of Internet design and unquestionable up until a few years ago.

Centralization, which was considered a bad thing until a few years ago, is now considered good, and for good reason. Most organizations and teams are run using centralized control. If an employee falls sick, he/she simply calls the boss, and the boss makes arrangements for the work to continue in his/her absence. Now consider what would happen in an organization that is totally distributed. The sick employee, say John, will have to call all his co-employees and tell them that he/she is sick. They will tell other employees that John is sick. This will take quite a bit of time before everyone will know about John's sickness, and then everyone will decide what, if anything, to do to alleviate the problem until John recovers. This is quite inefficient, but is how current Internet control protocols work. Centralization of control makes sensing the state and adjusting the control dynamically based on state changes much faster than with distributed protocols.

Of course, centralization has scaling issues but so do distributed methods. For both cases, we need to divide the network into subsets or areas that are small enough to have a common control strategy. A clear advantage of centralized control is that the state changes or policy changes propagate much faster than in a totally distributed system. Also, standby controllers can be used to take over in case of failures of the main controller. Note that the data plane is still fully distributed.

Now that the control plane is centralized in a central controller, it is easy for the network manager to implement control changes by simply changing the control program. In effect, with a suitable API, one can implement a variety of policies and change them dynamically as the system states or needs change.

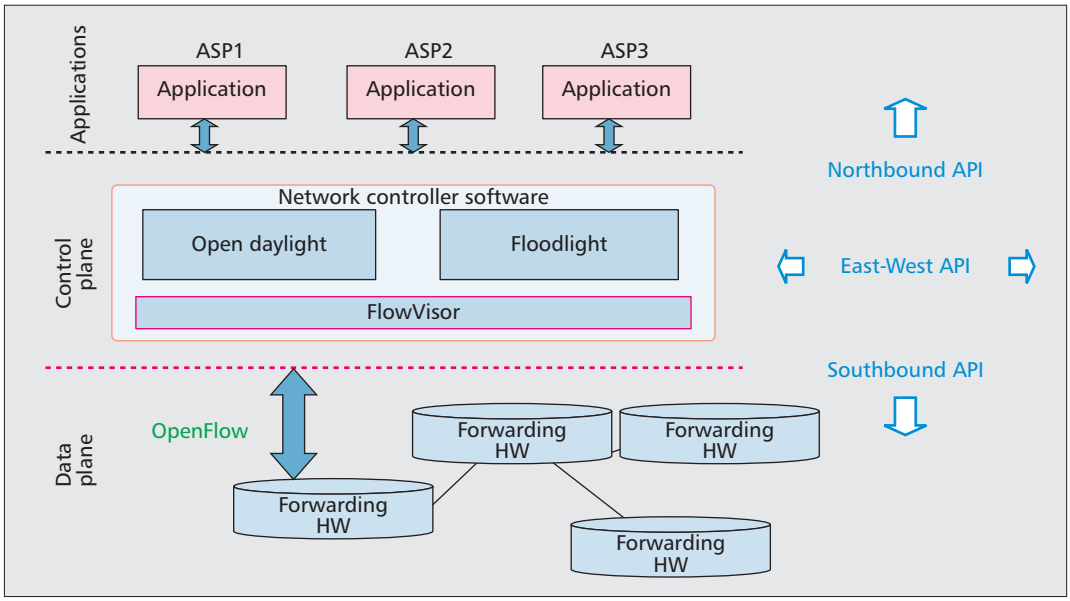

Figure 4. *Software defined networking APIs.*

Programmable Control Plane

Now that the control plane is centralized in a central controller, it is easy for the network manager to implement control changes by simply changing the control program. In effect, with a suitable API, one can implement a variety of policies and change them dynamically as the system states or needs change.

This programmable control plane is the most important aspect of the SDN. A programmable control plane in effect allows the network to be divided into several virtual networks that have very different policies and yet reside on a shared hardware infrastructure. Dynamically changing the policy would be very difficult and slow with a totally distributed control plane.

Standardized APIs

As shown in Fig. 4, SDN consists of a centralized control plane with a southbound API for communication with the hardware infrastructure and a northbound API for communication with the network applications. The control plane can be further subdivided into a hypervisor layer and a control system layer. A number of controllers are already available. Floodlight [10] is one example. OpenDaylight [11] is a multi-company effort to develop an open source controller. A networking hypervisor called FlowVisor [12] that acts as a transparent proxy between forwarding hardware and multiple controllers is also available.

The main southbound API is OpenFlow [13], which is being standardized by the Open Networking Foundation. A number of proprietary southbound APIs also exist, such as OnePK [14] from Cisco. These later ones are especially suitable for legacy equipment from respective vendors. Some argue that a number of previously existing control and management protocols, such as Extensible Messaging and Presence Protocol (XMPP), Interface to the Routing System (I2RS), Software Driven Networking Protocol (SDNP), Active Virtual Network Management Protocol (AVNP), Simple Network Management Protocol (SNMP), Network Configuration (NetConf), Forwarding and Control Element Separation (ForCES), Path Computation Element (PCE), and Content Delivery Network Interconnection (CDNI), are also potential southbound APIs. However, given that each of these was developed for another specific application, they have limited applicability as a general-purpose southbound control API.

Northbound APIs have not been standardized yet. Each controller may have a different programming interface. Until this API is standardized, development of network applications for SDN will be limited. There is also a need for an east-west API that will allow different controllers from neighboring domains or in the same domain to communicate with each other.

Flow-Based Control

Over the last 30 years (since the standardization of the first Ethernet standard), disk and memory sizes have grown exponentially using Moore's law, and so have the file sizes. The packet size, however, has remained the same (approximately 1518-byte Ethernet frames). Therefore, much of the traffic today consists of a sequence of packets rather than a single packet. For example, a large file may require transmission of hundreds of packets. Streaming media generally consists of a stream of packets exchanged over a long period of time. In such cases, if a control decision is made for the first packet of the flow, it can be reused for all subsequent packets. Thus, flow-based control significantly reduces the traffic between the controller and the forwarding element. The control information is requested by the forwarding element when the first packet of a flow is received and is used for all subsequent packets of the flow. A flow can be defined by any mask on the packet headers and the input port from which the packet was received. A typical flow table entry is shown in Fig. 5. The control table entry specifies how to handle the

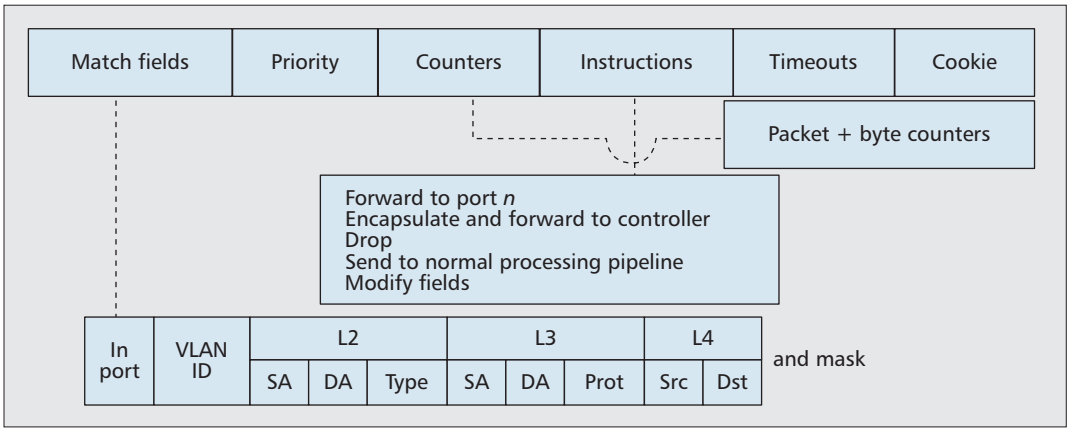

Figure 5. *OpenFlow table entries.*

packets with the matching header. It also contains instructions about which statistics to collect about the matching flows.

SDN Impact and Future

Networking industry has shown enormous interest in SDN. SDN is expected to make the networks programmable and easily partitionable and virtualizable. These features are required for cloud computing where the network infrastructure is shared by a number of competing entities. Also, given simplified data plane, the forwarding elements are expected to be very cheap standard hardware. Thus, SDN is expected to reduce both capital expenditure and operational expenditure for service providers, cloud service providers, and enterprise data centers that use lots of switches and routers.

SDN is like a tsunami that is taking over other parts of the computing industry as well. More and more devices are following the software defined path with most of the logic implemented in software over standard processors. Thus, today we have software defined base stations, software defined optical switches, software defined routers, and so on.

Regardless of what happens to current approaches to SDN, it is certain that the networks of tomorrow will be more programmable than today. Programmability will become a common feature of all networking hardware so that a large number of devices can be programmed (aka orchestrated) simultaneously. The exact APIs that will become common will be decided by transition strategies since billions of legacy networking devices will need to be included in any orchestration.

It must be pointed out that NFV and SDN are highly complementary technologies. They are not dependent on each other.

Open Application Delivery Using SDN

While current SDN-based efforts are mostly restricted to L3 and below (network traffic), it may be extended to manage L3 and above application traffic as well. Application traffic management involves enforcing application deployment and delivery policies on application traffic flows that may be identified by the type of application, application deployment context (application partitioning and replication, intermediary service access for security, performance, etc.), user and server contexts (load, mobility, failures, etc.), and application QoS requirements. This is required since delivering modern Internet-scale applications has become increasingly complex even inside a single private data center.

The *application service* may be **replicated** over multiple hosts. Also, the *service* may be **partitioned** for improved performance, with each partition hosted on a different group of servers. A service may be partitioned based on:

- **Content:** For example, even for the same service (e.g. videos.google.com) accounting messages, recommendation requests and video requests are all sent to different server groups.
- **Context:** User context, network context, or server context may require the application messages to be routed differently.

An example of user context is a mobile smart phone user vs. a desktop user. An example of network context is the geographical location of the user and the state of the network links. An example of server context is the load on various servers and whether they are up/down. Furthermore, most services require **multiple TCP segments**, where accessing the service actually requires going through a sequence of middle boxes providing security (e.g., firewalls, IDS),

Every new technology is like a new marriage. Before marriage, life is made of dreams. Both sides think there is this other person who has all the right qualities he/she needs, and that all his/her problems will be solved by this person. After marriage both parties realize that not all their beliefs were correct. There was a bit of hype. Similarly, all new technologies have a hype phase. This is when a lot of money is invested in the technology. As a result, the best the technology can do is developed. This is the time when researchers, startups, and all vendors need to pay attention since this is the opportunity to make an impact. This is when the fates of many companies are decided. Often, several competing approaches to get the same effect are developed, and the one requiring the least changes gets accepted. For example, asynchronous transfer mode (ATM) promised to solve many problems in networking. The key feature was guaranteed quality of service (QoS). Multiprotocol label switching (MPLS) offered this feature without a major replacement of legacy architecture, and survived while ATM went away.

Sidebar 3. *Technology hype.*

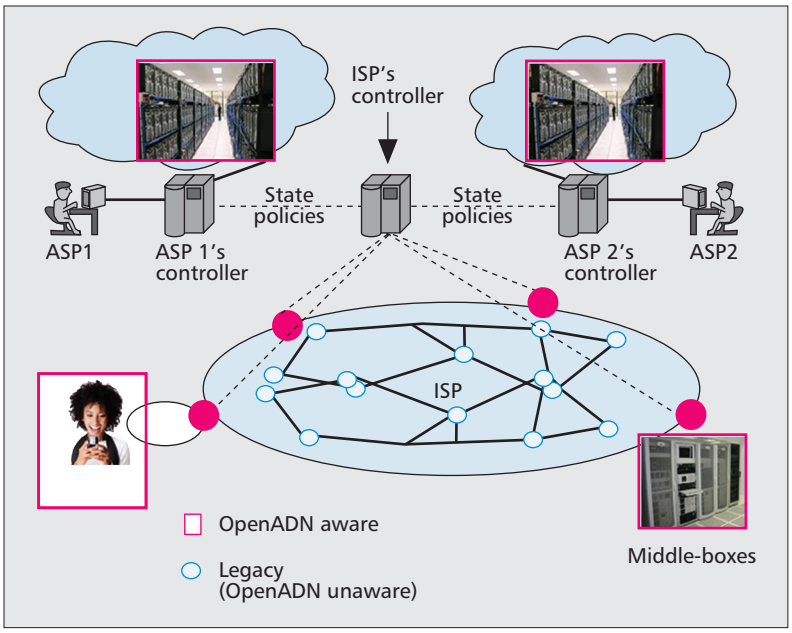

Figure 6. *In OpenADN, ASPs' controllers convey their policies to an ISP's controller in the control plane.*

transformation/translation (e.g., transcoders, data compression) and performance enhancement (e.g., SSL off loaders, WAN optimizers) functions to the service deployment. In general, a user-server connection is no longer end-to-end; it consists of many segments. Each of these segments can be served by multiple destinations (based on replication, partitioning). The application service providers (ASPs) therefore implement complex application policy routing (APR) mechanisms inside their private data centers.

Problem Statement

Most applications now (including games on smart phones) need to serve global audiences and need servers located all around the world. They can easily get computing and storage facilities using cloud services from multiple cloud providers distributed throughout the world. However, the problem of routing using ASPs' policies in a very dynamic multi-cloud environment is not possible since Internet service providers (ISPs) offer no service to dynamically route messages to a different server using an ASP's policies.

Solution Approach

Our vision is to design a new session-layer abstraction called Open Application Delivery Network (OpenADN) [15] that allows ASPs to express and enforce application traffic management policies and application delivery constraints at the granularity of application messages and packets. It allows them to achieve all the application delivery services they use today in private data centers in the global multi-cloud environment. OpenADN is based on the standardized data plane, diversified control plane design framework proposed by SDN. Using OpenADN-aware data plane entities, ISPs can offer application delivery services to ASPs.

To achieve this we combine the following six innovations: OpenFlow, SDN, session splicing, cross-layer communication, indirection, MPLS-like application flow labels (which we call APLS, application label switching).

As shown in Fig. 6, OpenADN allows ASPs' controllers to communicate with the ISP's controller and provide the ISP with their server policies and server states so that the ISP's controller can program the control plane accordingly. In addition to requiring a northbound API, OpenADN also requires some extensions to the southbound API — OpenFlow.

Key Features of OpenADN

1. OpenADN takes network virtualization to the extreme of making the global Internet look like a virtual single data center to each ASP.
2. Proxies can be located anywhere on the global Internet. Of course, they should be located in proximity to users and servers for optimal performance.
3. Backward compatibility means that legacy traffic can pass through OpenADN boxes, and OpenADN traffic can pass through legacy boxes.
4. No changes to the core Internet are necessary since only some edge devices need to be OpenADN/SDN/OpenFlow-aware. The remaining devices and routers can remain legacy.
5. Incremental deployment can start with just a few OpenADN-aware OpenFlow switches.
6. Economic incentives for first adopters are to be found by ISPs that deploy a few of these switches, and those ASPs that use OpenADN will benefit immediately from the technology.
7. ISPs keep complete control over their network resources, while ASPs keep complete control over their application data, which may be confidential and encrypted.

Summary

The key messages of this article are:

1. Cloud computing is a result of advances in virtualization in computing, storage, and networking.
2. Networking virtualization is still in its infancy. Numerous standards related to network virtualization have recently been developed in the IEEE and Internet Engineering Task Force (IETF), and several are still being developed.

3 One of the key recent developments in this direction is software defined networking. The key innovations of SDN are separation of the control and data planes, centralization of control, programmability, and standard southbound, northbound, and east-west APIs. This will allow a large number of devices to easily be orchestrated (programmed).

4 OpenFlow is the standard southbound API being defined by Open Networking Forum.

5 We are working on OpenADN, which is a network application based on SDN that enables application partitioning and delivery in a multi-cloud environment.

References

[1] PCI-SIG, "Single Root I/O Virtualization and Sharing 1.1 Specification," http://www.pcisig.com/members/downloads/specifications/iov/sr-iov1_1_20Jan10.pdf, available only to members.

[2] IEEE Std. 802.1Qbg-2012, "IEEE Standard for Local and Metropolitan Area Networks — Media Access Control (MAC) Bridges and Virtual Bridged Local Area Networks — Amendment 21: Edge Virtual Bridging," July 5, 2012, http://standards.ieee.org/getieee802/download/802.1Qbg-2012.pdf, p. 191.

[3] R. Perlman *et al.*, "Routing Bridges (RBridges): Base Protocol Specification," IEEE RFC 6325, July 2011, 99 pages, http://tools.ietf.org/html/rfc6325.

[4] M. Sridharan *et al.*, "NVGRE: Network Virtualization Using Generic Routing Encapsulation," IETF Draft draft-sridharan-virtualization-nvgre-03.txt, Aug. 2013, http://tools.ietf.org/html/draft-sridharan-virtualization-nvgre-03, pp. 17.

[5] M. Mahalingam *et al.*, "VXLAN: A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks," IETF Draft draft-mahalingam-dutt-dcops-vxlan-04.txt, May 8, 2013, 22 pages, http://tools.ietf.org/html/draft-mahalingam-dutt-dcops-vxlan-04.

[6] B. Davie, Ed., J. Gross, "A Stateless Transport Tunneling Protocol for Network Virtualization (STT)," IETF Draft draft-davie-stt-03.txt, Mar. 12, 2013, 19 pages, http://tools.ietf.org/html/draft-davie-stt-03.

[7] IEEE Std 802.1BR-2012, "IEEE Standard for Local and Metropolitan Area Networks—Virtual Bridged Local Area Networks — Bridge Port Extension," July 16, 2012, 135 pages, http://standards.ieee.org/getieee802/download/802.1BR-2012.pdf

[8] T. Narten *et al.*, "Problem Statement: Overlays for Network Virtualization," IETF Draft draft-ietf-nvo3-overlay-problem-statement-04, July 31, 2013, 24 pages, http://datatracker.ietf.org/doc/draft-ietf-nvo3-overlay-problem-statement/.

[9] ETSI, "NFV Whitepaper," Oct 22, 2012, http://portal.etsi.org/NFV/NFV_White_Paper.pdf

[10] Floodlight OpenFlow Controller, http://www.projectfloodlight.org/floodlight/.

[11] OpenDaylight, http://www.opendaylight.org/resources

[12] Flowvisor Wiki, https://github.com/OPENNETWORKINGLAB/flowvisor/wiki.

[13] Open Networking Foundation, "OpenFlow Switch Specification, V1.3.2," Apr. 25, 2013, 131 pages, https://www.opennetworking.org/sdn-resources/onf-specifications/openflow.

[14] Cisco's One Platform Kit (onePK), http://www.cisco.com/en/US/prod/iosswrel/onepk.html

[15] S. Paul and R. Jain, "OpenADN: Mobile Apps on Global Clouds Using OpenFlow and Software Defined Networking," *1st Int'l. Wksp. Management and Security Technologies for Cloud Computing*, Dec. 7, 2012.

Biographies

RAJ JAIN [F] (jain@cse.wustl.edu) is a Fellow of ACM and AAAS, a winner of the ACM SIGCOMM Test of Time Award and CDAC-ACCS Foundation Award 2009, and ranks among the top 100 in CiteseerX's list of Most Cited Authors in Computer Science. He is currently a professor in the Department of Computer Science and Engineering at Washington University. Previously, he was one of the cofounders of Nayna Networks, Inc., a next-generation telecommunications systems company in San Jose, California. He was a senior consulting engineer at Digital Equipment Corporation in Littleton, Massachusetts, and then a professor of computer and information sciences at Ohio State University, Columbus. He is the author of *Art of Computer Systems Performance Analysis*, which won the 1991 Best-Advanced How-to Book, Systems Award from the Computer Press Association.

SUBHARTHI PAUL [S] (pauls@cse.wustl.edu) received his B.S. degree from the University of Delhi, India, and his Master's degree in software engineering from Jadavpur University, Kolkata, India. He is presently a doctoral student in the Department of Computer Science and Engineering at Washington University. His primary research interests are in the area of future Internet architectures.

The key innovations of SDN are separation of control and data plane, centralization of control, programmability, standard southbound, northbound, and east-west APIs. This will allow a large number of devices to be easily orchestrated (programmed).