SDN and the Future of Service Provider Networks

shaping tomorrow with you

FUJITSU

SDN's Origins: The Rise of the Human-Centric Network

The world has seen rapid and historically unparalleled changes in technology over the past ten to twenty years, particularly in the area of mobile communications. As an example, in 1995 there were approximately five million cell phone subscribers in the US, less than 2% of the population. By 2012, according to CTIA, there were over 326 million subscribers, meaning that 102% of the American population had cell phones, of which over 123 million were smartphones. This rapid increase in uptake of mobile devices has made the availability of information location-independent, instead of being resident on fixed-location devices such as laptop computers. Ubiquitous information availability has led some to call this the "human-centric network," since network demands are being driven by individual, often mobile, users.

The consequence of explosive growth in mobile devices has been correspondingly dramatic growth in bandwidth demand, which has taken a toll on the mobile network infrastructure. In the datacenter environment, creating monolithic server platforms to support traffic growth was simply not a scalable solution. In 2005, the concept of "hypervisors," originally developed in the 1960s by IBM, began to gain popularity. Clouds, clusters, and virtual machines are all forms of elastic computing platforms that support the growing human-centric network.

But VMs are not the only virtualization taking place in the datacenter environment. A major component of modern virtualization has been in the area of storage. SDS uses policy-driven software to provision logical storage volumes, which can comprise various types of storage ranging from HDD arrays to SANs.The storage hypervisor manages all the various physical resources and understands their constraints, so it can optimize utilization and performance. Virtualized storage has, in fact, become the public face of virtualized networks. Most consumers are familiar with Dropbox, iCloud, SkyDrive, Google Drive and/or other "cloud" storage offerings leading to a general misconception by many in the public that "cloud" equals storage.

The fact is that all physical aspects of the datacenter are being virtualized. This has led to the term "SDDC", first coined by VMware, which describes the collective virtualization taking place, from computers and storage to networking and even security. But how does the virtualization of datacenter resources relate to SDN in the service provider's network? Specifically, how does SDN, designed for virtual orchestration of disparate computational resources, apply to transport networks? In this paper we will look at what SDN is and why it is not merely applicable to transport networks, but a requirement.

What is SDN?

The term SDN has become a popular concept in technology circles. So what exactly is SDN? As we know it today, SDN was a project that started at Stanford University in 2007. The core concept behind SDN is that it decouples the control layer from the data layer.

The control layer manages network devices by means of signaling. Although the original intent of SDN was primarily focused on services, all aspects of device management can be done at this layer, including end-to-end provisioning, capacity management, and performance and SLA management.

The data layer, of course, is the layer where the actual traffic flows. By separating the two, the control layer can use a different distribution model than the data layer. It also means that by not running things like path computation on the network devices, the NEs can be less expensive machines that focus on the job of transporting traffic. This leaves the network management job to software that runs on more powerful machines (computers). Distributing network management across multiple network devices is antithetical to the SDN model.

The Power of Abstraction

But the real power of SDN can be summed up in a single word: abstraction. Abstraction means that concepts are generalized and applied. Instead of sending specific code to the devices, the machines can talk to the controller in generalized terms. While at first glance this may seem like an odd concept, it is one that we use on a daily basis. An excellent example of this is our use of the printer. If you want to print a document to a printer there are unique commands that go from the PCs to the actual printer itself. Every brand and every model has its own unique way of printing. In the computer's OS, regardless of whether it is Windows, Apple, Linux or something else, a set of drivers know how to send the specific set of messages to the printers. The OS sends the generalized message to the driver and it sends the specific messages (code) required to print out the document. From the users' perspective, they simply hit the print button. This is the power of abstraction.

With SDN, concepts are abstracted and the user makes requests of the network in a similar fashion to hitting the print button from an application. But instead of the application sending print commands, it can send link up/down or bandwidth commands, or pull reports on different aspects of the network. And just as with the computer where the user is printing from an application (like a word processor) which runs on top of the OS, there are applications that run on top of the SDN network controller.

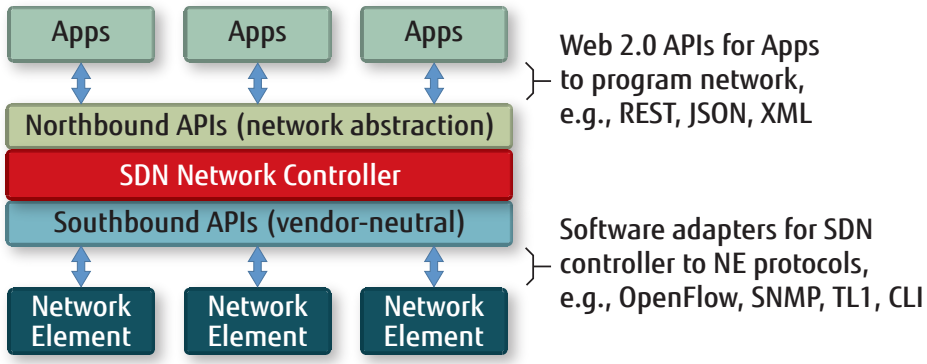

Figure 1: SDN Basic Architecture

As shown in Figure 1, applications can be written and plugged into the SDN network controller. Using an interface such as REST, applications can make requests from the SDN controller which will return the results. The controller understands the construction of the network and can communicate requests down to the various network elements that are connected to it. For example, an application could be created to provision end-to-end services. The application could query the resources through the controller to see if a path is available and if so, send the activation commands to the devices. For paths that have already been established, another application could create reports by searching for them.

The Southbound Interface and the OpenFlow Paradigm

The southbound interface handles all the communication with the network elements themselves. These interfaces provide the same function as the driver did in the earlier example of the printer. The southbound interface can take one of two forms. The first is a system that creates a more programmable network, meaning that instead of just sending commands to the devices to tell them what to do, SDN can actually reprogram the devices to function differently. The first, and currently most popular, interface of this type is the OpenFlow protocol which is managed by the ONF. Today OpenFlow is implemented on routers and switches. OpenFlow sends forwarding tables directly to the router, reprogramming it so that when a packet hits the router, it uses the table to determine its path.

The second type of southbound interface is a more traditional interface that can use existing communication protocols, which enables management of devices that do not support the ONF-standard OpenFlow interface. This includes interfaces on devices that are in deployment today, such as TL1 and SNMP.

In some cases, aspects of telecom equipment cannot effectively be managed by the existing OpenFlow paradigm. An example is that of fault switching. OpenFlow does not have an effective method to provide the hardware-based sub-50 ms protection switching we have today. Thus, there needs to be a hardware layer that can work in cooperation with the SDN controller to provide carrier-grade functionality. By the controller talking to the network devices using current communication protocols, these carrier-grade functions can continue to be delivered with the required performance levels.

Many people have come to the incorrect conclusion that OpenFlow itself is SDN. In fact, OpenFlow is merely an enabler of SDN. It is important to remember that the objective of SDN is to make control of the network more dynamic by separating the data and the control layers. The southbound interface, as important as it is, is simply a way for the SDN controller to talk to the devices, whether the device itself is configured by a forwarding table or not. The abstraction of the network is not dependent on the southbound interface.

Another driving factor in the move to SDN by service providers is the ability to control disparate technologies, not just multiple vendors' equipment. Networks are, of course, composed of different devices that manage specific segments of the network. As shown in Figure 2, a wireless service provider will have wireless transmission equipment (including small-cell fronthaul) with transport equipment to backhaul traffic to the data center. In the data center, there will be routers, switches, servers and other devices.

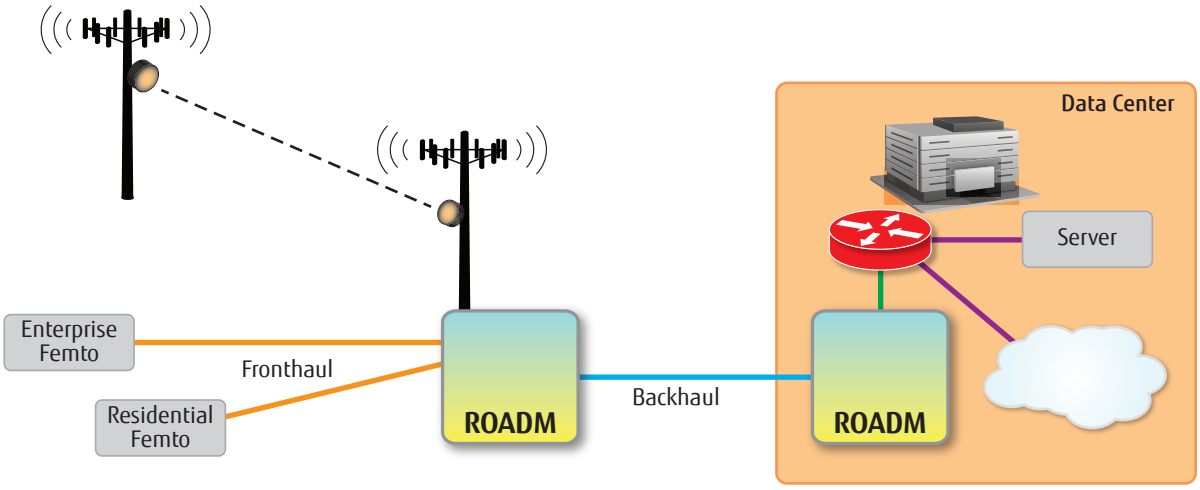

Figure 2: Cell tower backhaul

At best, these data center devices are under "swivel-chair management" and at worst, they have multiple NOCs managing their respective segments. Not only does this add OpEx in terms of both staffing and equipment cost, but this arrangement also makes provisioning difficult and time-consuming since each network section must provision its part in a coordinated fashion. If a configuration parameter is mismatched, then the service may not function as planned and troubleshooting is difficult, requiring multiple technicians to track down the network segment that is the cause of the problem.

SDN architecture provides an orchestration layer that sits above the control layer. This layer communicates with multiple controllers. As in the case of the controllers, the orchestration layer has north- and southbound interfaces. The northbound interface, as with the controllers, allows for the development of applications that can perform global network tasks such as true end-to-end service provisioning.

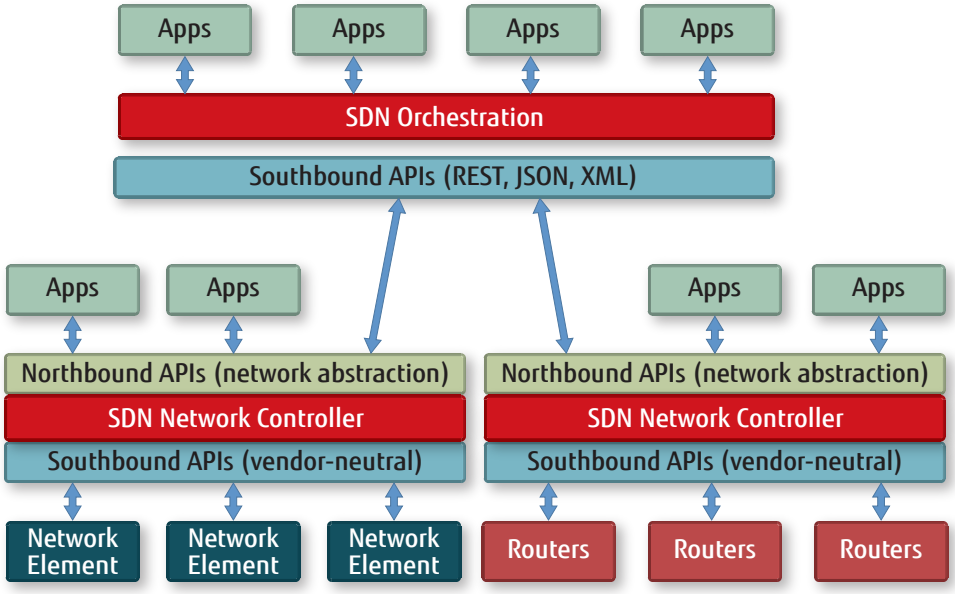

Figure 3: SDN orchestration

Why do Service Providers Need SDN?

Before considering migration to SDN, is it is important to understand the reasons for changing existing systems. Changes to network management systems should always be driven by the same generic business goals.

The first goal is to simplify the software to improve service delivery times. The second goal is to improve the use of network resources, thereby reducing costs (both OpEx and CapEx), and/or increasing revenues. Whatever system is put in place must bring these key business benefits. It is a waste of time and money to build a new system because it is the latest trend rather than because it will achieve these basic goals.

Keeping this in mind, telecom networks have traditionally been as monolithic as the servers that are being virtualized in the datacenter. As an example of how SDN can change network provisioning, consider what it would take to modify the bandwidth shown in Figure 2. If there is an existing 100 Mb Ethernet connection from the data center to the fronthaul and it is decided that this connection needs to be 150 Mb, a coordinated effort by several teams is required to accomplish this. One team must increase the bandwidth settings of the small cells; the transport team must increase bandwidth on the NEs; and routers and switches in the data center must be configured by yet another team.

In today's networks, teams like these are highly trained and skilled experts in device-specific management code. For example, those working on transport gear will often launch a terminal and type TL1 commands, while another technician will use a CLI interface to control a router or switch. In effect, these technicians function as "printer drivers" for the network. Imagine if, every time you wanted to print a document, you had to take it to an employee and ask them to write the specific commands to the printer on your floor. This, of course, would be ridiculous, yet every communications network company in the world functions by this model every day.

In short, network adds, moves, and changes are time-consuming and burdensome in an ever-changing world where dynamic bandwidth needs are no longer negotiable. What is truly needed is the ability to respond to this demand in real time, where one individual can provision the bandwidth using the power of abstraction. The infrastructure must be enabled to move at a pace that is closer to the one-click world we live in today. SDN provides the framework required to make this happen.

In effect, SDN serves as a middleware layer that allows users to write applications that sit on the northbound side and speak, via the southbound interfaces, to the devices. It leverages the same abstraction concept that the computer world has been using for so long. This means communications networks can transition from the technician-provisioned model to Web 2.0 programmable networks.

SDN Applications

A discussion of technology always has inherent value, particularly from an educational perspective. But from a practical standpoint, the reason technology exists is to deliver functional benefits. In SDN that value largely derives from what the applications can do for you. Therefore no discussion of SDN is complete without considering the specific benefits of applications.

Note that it is not within the scope of this paper to discuss the exact details of how to program an application. Specific implementations are based upon the controller or orchestrator interface as it was built by the vendor. Most interfaces will be RESTful with JSON and/or XML for data transport, but vendors are not constrained to these and individual interfaces may vary. The vendor should provide a well-documented SDK for interface building.

Many applications can be used in an SDN network. Figure 4 shows a list of examples, broken down by application type. This list is by no means exhaustive; it also contains some examples that may not apply to the transport network. However, while some of the functions may not apply to the transport section, having the orchestration layer means that you can control many different sections of your network, including those that are not transport-related. As such the list in Figure 4 is informative as to the tremendous number of functions that can be used in an SDN-capable network.

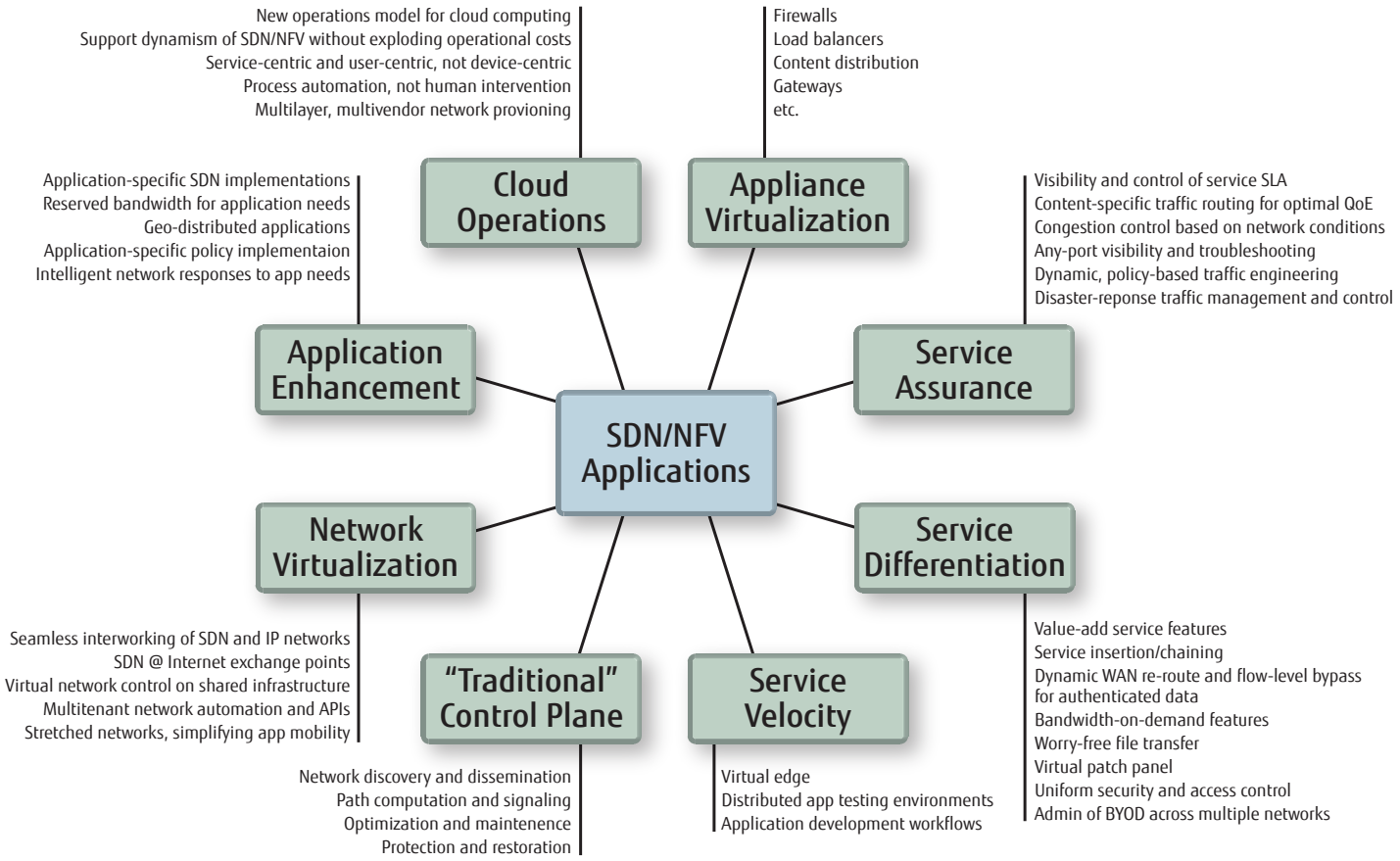

Figure 4: SDN Applications

Path Computation

Let's consider a few examples of the SDN applications that specifically apply to service provider networks. One common application will be path computation, or end-to-end provisioning. Over the years there have been many methods that have sought to provide a PCE. One attempt was to embed the PCE into the NEs. This idea is, in fact, the opposite concept from SDN, because it mingles the control and data layers. From a practical standpoint, merging these two layers has several limitations. One is that, because the hardware on the NEs is limited, the scale of the domain this hardware manages is also limited. SDN overcomes this issue by the very nature of the hardware it runs on, specifically a server. Should the server become unable to manage the network due to size, additional capacity can be added by simply increasing the hardware (e.g. adding a blade or hard drive) or, if running on an elastic computing platform, by simply requesting additional computing resources. Another limitation of merged control and data layers is interoperability with other devices via the control layer. Since not all systems share common signaling protocols, the systems with embedded controls will become isolated. In the case of SDN, the southbound adaptors mitigate this issue by not only being able to work with disparate protocols but also by being able to manage systems that do not have embedded controllers.

With the ability to compute a path across the network, another logical application is workflow or flow-through provisioning. Order entry systems can query the SDN layer to see what resources are available. These are real time queries of the actual network, and not just of a database that could potentially have out-of-date information. This means the network is used more efficiently. The order system can then programmatically build the RESTful commands needed to tell the SDN orchestrator or controller what the endpoints, bandwidth demands, and restrictions of the needed services are, and the SDN layer can automatically provision them. Additionally the SDN system can interface with the billing system for auditing purposes.

SDN and OTN Applications

A prime example of SDN being used to configure services can be seen when it is applied to OTN. OTN is a multilayered technology that allows users to densely and efficiently pack different service types into a single DWDM wavelength. OTN can greatly benefit the network by optimizing transport, but it does add some complexity that can be simplified using SDN.

A user might have multiple Ethernet flows that are headed to the same destination (perhaps a datacenter). OTN can optimally transport all of them in an aggregate flow. Assuming that the flows are smaller services (1 Gb or smaller), these can be mapped into a LO ODU0. SDN can multiplex the multiple LO containers bound for a particular destination into HO OTU containers. It can then map the HO containers onto wavelengths that connect the two points so there is no regeneration in between, which makes the network more efficient.

Needless to say, not all flows will be carried by LO ODUs. As such, SDN can determine what type of container or transport is available and required for the service. As mentioned above, SDN could map the service to an aggregated flow in an OTN container, which is riding on a wavelength. But perhaps a needed segment does not have OTN, or perhaps the service is large in and of itself (e.g. 100 Gb), so that it does not need to be combined with another service. In that case, SDN could map the service directly to a wavelength. The decision of whether to aggregate into an LO or HO OTU, or to just use a wavelength, can be made from the logic programmed into the SDN application, making the network even more efficient.

Network Optimization

Another area where SDN can improve network utilization is by optimizing the network so that over time, it can make better use of resources. Again, using the example of OTN, SDN applications can be used to reroute OTN paths to minimize latency, to prepare for cutovers, or based on churn in demand. The application can run as a background scheduled task to automatically look for opportunities to perform optimizations. It can then generate executive reports showing how it has performed the optimization.

Automated Testing

Once the system has created the end-to-end service, another application (or perhaps part of the turn-up application) can perform automated testing. Where software test connections are possible, the system can not only turn up the cross connects for the test but can activate the testing protocols. For example, as part of turning up an Ethernet service, the system could create a Y.1564 service "birth certificate." Once the test is complete, the SDN system can take the report that is generated on the test gear and, if it fails, notify a technician that there is a problem with the circuit. If the circuit passes the test, the system can send the report to the inventory database, where it can be archived with the other circuit information.

Protection and Restoration

Another application that can be built is for protection and restoration. As mentioned earlier, in order to meet the required 50 ms protection needed in carrier-class networks much of the switching must take place in the hardware on the NEs. This means the SDN controller will be passing to the hardware messages that define the protection path. One advantage is that the SDN system can systematically search for the best possible restoration paths, even as new links are added to the existing network. It can search and find the most efficient path as they become available. This means the system could discover and utilize a 1:n protection scheme instead of using a less effective one-for-one method. It also means that if there is a failure, the system can dynamically compute additional protection paths. Today there are systems that can do this using embedded controllers. Unfortunately after multiple failures, the network becomes fragmented as the system moves services to different links. This can be thought of as being similar to the fragmentation of a hard drive. Typically, tools need to be run to optimize the network, much like the manual defragmentation process that needed to be run in older versions of Windows. Like the newer versions of Windows, SDN can automatically "defragment" to optimize the network as part of the protection application.

SLA Management

Another possible application is SLA management, which has become a key challenge for many customers. End customers have SLA requirements associated with the services they have purchased, and they want to have a Web portal to view conformance data. The Web portal can be of great benefit, premiums are charged for both the SLA and the Web portal. If the user thinks they are having an issue and they look at their portal they may find that they have more traffic than what is covered by the SLA. This means more than just saving a troubleshooting call to the service department. It means that the customer may look at adding additional bandwidth, driving new revenue for the service provider. It is important to understand, whether adding an SLA portal or using an existing system, pulling the information via SDN is simpler, especially given the fact that REST is a web interface.

Custom Applications

Custom applications can also be written to meet a service provider's specific needs. One example might be an application to backup large databases without affecting traffic. Using SDN you can not only schedule the backup itself, but also an increase in available bandwidth during the file transfer. The timing can be set to a known off-peak time when transport bandwidth is underutilized. After the transfer is over the bandwidth can be restored to the original size, causing no impact to customer traffic.

Network Function Virtualization (NFV)

In addition to applications, SDN becomes an enabler of NFV. NFV allows companies to provide services that currently run on dedicated hardware located on the end user's premises by moving the functionality to the network. Content delivery, such as Netflix video on demand, is a prime example of this. Content (movies) that was once delivered exclusively on VCRs, DVDs and BluRay, can now be streamed over the network with full pause, rewind, and fast-forward functions. An example of an NFV function that service providers can deliver in context of network services would be a firewall. Instead of a customer (who is using a service-provider Internet connection) setting up and maintaining a firewall in their office, they could subscribe to the service. In NFV this is known as a virtual appliance.

Summary and Conclusion

It is time to think of the network as being more than just a collection of transport hardware. We are building a human-centric network that caters to a mobile generation who think nothing of going shopping while they are riding the bus to work, or of streaming a movie on the train.

"Over the top" providers such as Netflix have seen the value of the cloud and have already exploited it to their advantage. The transport networks of today do not match the cloud based world that we now live in. Networks need to become as agile as the services that are being deployed on them.

SDN is capable of creating a programmable network by taking both next-generation systems and existing infrastructure and making them substantially more dynamic. It does this by bringing disparate systems and technologies together under a common management system that can utilize them to their full potential. By using abstraction, SDN can simplify the software needed to deliver services, improvingthe use of the network and shortening delivery times, leading to increased revenue.

In addition to OpEx savings, new revenue streams can be realized by using SDN applications. The possibilities that can be addressed by the application's interface are practically limitless. An open northbound interface means that users can build their own applications and are not waiting on management system vendors to deliver features that they need now. More than that, there are applications that can cross multiple domains that could never have been addressed in the past.

FUJITSU NETWORK COMMUNICATIONS INC.
2801 Telecom Parkway, Richardson, Texas 75082-3515
Telephone: 888.362.7763
us.fujitsu.com/telecom

Acronyms

CapEx	Capital Expenditure
CLI	Common Line Interface
DWDM	Dense Wave Division Multiplexing
HDD	Hard Disk Drive
HO	Higher Order
JSON	JavaScript Object Notification
LO	Lower Order
NE	Network Element
NOC	Network Operations Center
ONF	Open Networking Foundation
OpEx	Operating Expense
OS	Operating System
OTN	Optical Transport Network
OTU	Optical Transport Unit
PCE	Path Computation Engine
REST	Representational State Transfer
SAN	Storage Area Networks
SDDC	Software-Defined Datacenter
SDK	Software Developer's Kit
SDN	Software-Defined Network
SLA	Service Level Agreement
SDS	Software-Defined Storage
SNMP	Simple Network Management Protocol
VM	Virtual Machines
XML	Extensible Markup Language