Service Chaining for NFV and Delivery of other Applications in a Global Multi-Cloud Environment

Subharthi Paul, Raj Jain
Dept. of Computer Science and Engineering
Washington University in St. Louis
St. Louis, MO 63130, USA
{pauls, jain}@cse.wustl.edu

Mohammed Samaka, Aiman Erbad
Computer Science and Engineering Dept.
Qatar University
Doha, Qatar
{samaka.m, aerbad}@qu.edu.qa

Abstract— **Network Function Virtualization (NFV) allows Internet Service Providers (ISPs) to implement key function modules, such as, BRAS (Broadband Remote Access Server), IMS (Internet Multimedia System), etc. in virtual machines in a cloud environment. One of the key problems in NFV implementation is the placement of virtual machines (VMs) in clouds managed by different cloud service providers each with its own management interface. It would be helpful if the clients can implement their policies in a multi-cloud environment using a single interface. Our proposed solution is a modular multi-cloud management system called OpenADN that provides a common interface for resource allocation in a multi-cloud environment. The solution is also applicable to non-ISP applications, such as, banking, financial, and other sectors that need to use globally distributed multi-cloud resources. This paper presents a brief overview of the OpenADN architecture. The key feature of OpenADN is that multiple tenants can share the resources and all resource owners keep complete control over their resources. The data plane module of OpenADN is called OpenADN (Open Application Delivery Network). OpenADN has been implemented and brief details of implementation are also presented in this paper.**[1]

Keywords—Cloud Computing; Multi-Cloud; Inter-Cloud; Network Function Virtualization; NFV; Software Defined Networking; SDN

I. INTRODUCTION

Four recent innovations that have significantly impacted the field of computer networking are: Virtualization, Cloud Computing, Smart Mobile Phones, and Software Defined Networking (SDN). Network Function Virtualization (NFV), which is a recent proposal from Internet Service Providers (ISPs), is the next innovation that promises to revolutionize the carrier networking.

SDN has been a topic of much interest and debate in the industry recently and its definition has undergone a significant change over the last two years.

[1] This work has been supported under the grant ID NPRP 6 - 901 - 2 - 370 for the project entitled "Middleware Architecture for Cloud Based Services Using Software Defined Networking (SDN)", which is funded by the Qatar National Research Fund (QNRF). The statements made herein are solely the responsibility of the authors.

In this paper, we briefly present the current industry activities in SDN and extend those concepts to solve one of the generic problems of NFV, called service chaining. We show that service chaining is a generic problem and solving it will help enterprises in general since many of them use multiple clouds.

The organization of the rest of the paper is as follows. In Section II, we present current evolution of SDN to a multi-protocol version. In Section III, we provide some background on NFV, followed by a discussion on the service chaining problem in Section IV. In Section V, we discuss how NFV can be generalized to address enterprise application delivery problems. In Section VI, we present our solution- the OpenADN architecture, followed by a brief discussion on the features of OpenADN in Section VII and some of the application use-cases that may benefit from OpenADN in Section VIII. Finally, we summarize in Section IX.

II. MULTI-PROTOCOL SDN

SDN has changed the cloud computing and data center management landscape. The concept originated with OpenFlow [1], which requires separating the control plane from network elements and centralizing it in a controller. The central controller allows programming the entire datacenter network and implementing the organization's policies uniformly on a large number of network devices. This is a very powerful and useful concept and so it immediately caught the attention of the entire networking industry.

SDN was originally proposed with a single southbound protocol (OpenFlow). The key debate about SDN in the industry is that; if what we need is the uniform implementation of policies then what is the easiest way to get there? Every company has its own way to achieve this goal. In particular, there is a debate about whether complete separation of the control plane is necessary, possible, or desired.

Strictly speaking, data plane consists of all bits sent by the user. Control plane consists of all bits that are added by the network to transport these user/data bits. In circuit switched networks, separate (virtual) channels are used for the control information. In packet switched networks, such a separation is difficult because control bits are added to each user payload in the form of headers and then additional control packets are used to determine routing and enforcing other policies.

Complete separation of control plane will leave the network vulnerable to total stoppage in case of controller

failures so a significant amount of work has been done using distributed implementation of controllers that make it failsafe. However, if the key objective is to enforce policies, then some argue that the control plane can be left intact in the network elements and the policies can be sent to them directly. This is what OpFlex [2] is being designed to do. Also centralization of all control plane functions requires micromanaging millions/billions of flows from the controller which raises scalability issues.

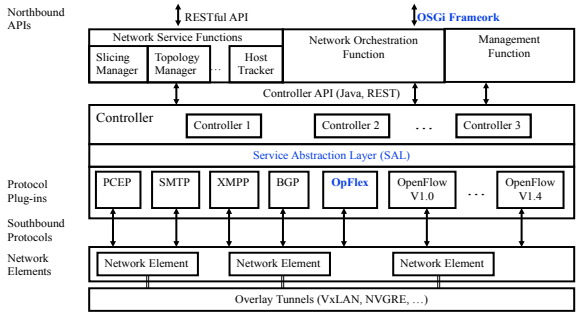

Fig. 1: Multi-protocol implementation of SDN in OpenDaylight

OpenDaylight [3], which is currently the leading industry effort on a software defined networking controller, uses a modular approach to software defined networking. As shown in Fig. 1, it allows many different southbound protocols including, PCEP (Path Computation Element Protocol), SMTP (Simple Mail Transfer Protocol), XMPP (eXetensible Messaging and Presence Protocol), BGP (Border Gateway Protocol), OpFlex [2], and OpenFlow V1.0 through OpenFlow V1.4 [4]. The service abstraction layer (SAL) allows many different control modules to talk to the underlying physical hardware (network elements) using any of these protocols.

Later in this paper, we will extend this modular multi-protocol design concept to apply to multi-cloud environment.

III. Network Function Virtualization

In October 2012, ETSI published the Network Function Virtualization (NFV) whitepaper [5]. The key idea is that similar to the economies other industries are achieving by moving to cloud computing, Internet Service Providers (ISPs) can also reduce capital expenditure (CapEx) and Operational expenditure (OpEx) by virtualizing various ISP functions. Examples of such functions are IMS (Internet Multimedia System), EPC (Evolved Packet Core), BRAS (Broadband Remote Access System), etc. As shown in Fig. 2, each of these functions can be implemented as a virtual machine in one or more clouds.

Cloud computing has proven to be a successful business model for both the cloud service providers and for the cloud service clients. The clients save by not having to worry about large CapEx and OpEx to manage the physical infrastructure. Sharing of infrastructure by many tenants results in the service provider also making significant profits. So cloud computing results in a win-win business relationship between the service provider and the clients. NFV will bring similar savings to the ISP businesses.

Multiple copies of virtual network function (VNF) modules can be instantiated on demand as required. As traffic increases in an area, more VNFs can be created to meet the demand. When the traffic goes away, VNFs can be shut down and the available capacity can be used for other tasks. This also allows the possibility of the physical NFV infrastructure (NFVI) to be owned by a separate entity and shared by multiple client ISPs resulting in the same win-win relationship between the ISPs and NFVI providers as between the cloud service clients and the cloud service providers.

The key advantages of NFV are:

1. **Programmability**: The entire network can be managed from the cloud management interface.
2. **Multi-Tenancy**: The datacenter network can be partitioned into multiple virtual partitions with each partition being managed by a different ISP. Each tenant has different policies that are conveyed by the tenant controller to the Cloud Service Provider's (CSP's) management interface and implemented from there.
3. **Orchestration**: Policies can be implemented on a large number of network elements simultaneously from the cloud management platforms.
4. **Dynamic Scaling**: Each tenant's network can be scaled up and down as needed.
5. **Automation**: Manual configuration of each individual device is avoided. Modules in the cloud management software can compute various configuration parameters automatically resulting in significant OpEx savings.
6. **Visibility**: With network devices reporting to the central cloud management software, it is easy to see all the activities in the network and diagnose problems quickly.
7. **Performance**: Visibility and programmability allow the network performance to be optimized with the changing traffic patterns.
8. **Unified Management**: Virtualization of computing and storage already provides the above features for computing and storage in clouds. NFV adds these features to networks, thus, allowing a unified management of computing, storage, and networking.

IV. Service Chaining

One key problem in the NFV implementation is that of connecting various VNFs. This is called "Service Chaining." For VNFs located inside a single cloud, this consists of programming the data center network so that the traffic flows through the various VMs according to the policies of the tenant. The network links in the data center can be programmed accordingly. Similarly, for VNFs located in different cloud data centers, the tenants would like their traffic to be handled by their policies. However, this requires elastic network links between the clouds that can be instantiated on demand. Since the WAN link

capacities are extremely limited and expensive, the available capacity often dictates the VNF placement to a subset of available clouds.

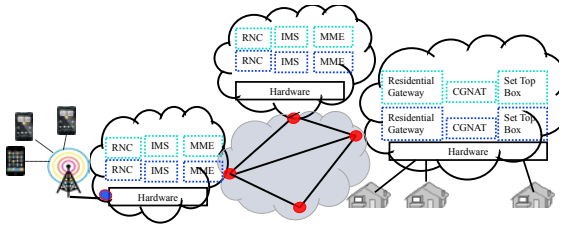

Fig. 2: Service chaining in a multi-cloud multi-tenant environment

The key challenges in service chaining are:

1. **Dynamic**: Forwarding changes with the state of servers and links. If servers at a particular cloud become overloaded, new or even existing flows must be forwarded to other clouds. The same applies to links connecting clouds. Cloud operators may want to move the VMs themselves for security, reliability, performance, or in anticipation of load changes.

2. **QoS vs. Cost**: The latency is determined by link utilization. Unlike intra-datacenter LAN links, the WAN links are expensive and so these often have low bandwidth and high utilization.

3. **Content Sensitive**: Forwarding of messages depends upon the message content. For example, video, accounting, and data messages for the same service may need to be sent to servers located in different clouds. Read and write requests may need to be sent to different servers. Much of this information belongs to Layer 5-7 headers and may not be available in Layer 2-4 headers.

4. **Distributed Control**: While, the application and the application level information belong to the application service providers (ASPs), the network belongs to Internet service providers. ISPs may not have access to L5-7 header and ASPs sharing a multi-tenant network may not be able to directly control the forwarding behavior.

5. **Massive Scale**: The number flows in many of these global applications can be huge, with each flow requiring a different forwarding depending on the user context (e.g., Cell phone users, lap top users, administrators, etc.)

6. **Stateful Services**: Some middle boxes, e.g., intrusion detection, are packet level services and not all messages or packets of a flow need to pass through them. Therefore, the flows may be diverted around these boxes for better efficiency. Other middle boxes, such as firewalls, are stateful and need to see all the packets of the flow. These flows should always be sent to the same VMs.

V. Generalization of NFV Concepts to Other Enterprises

It should be pointed out that the NFV concept was originated by the ISPs. However, the concept is general and can be used by any other industry. For example, banking industry could come up with a list of functions that are commonly used by different banks and could develop a set of virtual machine implementations of those functions. If ISPs can solve the problem of service chaining for them, they can provide it as a service to other industries and help them place VMs among global clouds that have the required connectivity.

It has been estimated that most (74%) enterprises use more than one cloud [6]. For the case, where two clouds belong to the same cloud service provider, the link connecting the two clouds may be a high-capacity link and a desired share of it can be allocated to a tenant sharing the two clouds. However, if the two clouds belong to different cloud service providers, generally the link connecting the two clouds is limited to the best effort Internet connection. If the ISPs can come up with an elastic capacity link creation and allocation methods for service chaining, these can be used by other enterprises also.

VI. Our Proposed Solution: OpenADN

To solve the problem of service chaining in a multi-cloud environment, we have developed a platform that we call "OpenADN." As shown in Fig. 3, on the north side, it offers three interfaces – for application developers, application architects, and application deployment administrators, respectively. On the south side it has many modules, one for each of the cloud/network management systems. In the figure, we have shown OpenStack, EC2, and OpenDaylight as examples of cloud/network management systems.

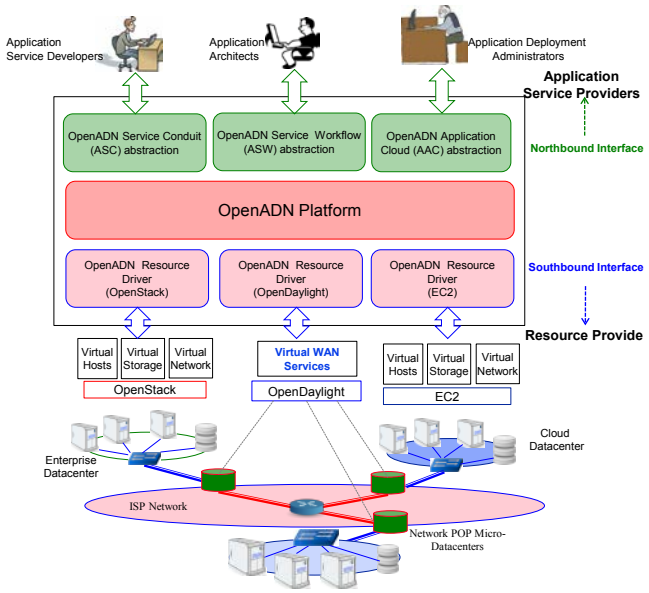

Fig. 3: OpenADN Platform for Services in a Cloud of Clouds

Notice that the OpenADN architecture has a modular structure similar to the OpenDaylight SDN controller. The northbound interface of OpenDaylight becomes one of the southbound interfaces of OpenADN. While OpenDaylight allows implementing client policies in one cloud, OpenADN allows implementing client polices uniformly among all the clouds.

OpenADN by itself does not meddle with the resources inside the clouds; it simply requests the respective cloud manager to create those resources. The polices of when and where to create the resources are specified by the Application Deployment Manager.

VII. Features of OpenADN

OpenADN allows automatic creation and deletion of application workflows as the load or traffic locality changes.

The inputs to OpenADN are 3 policy modules (currently specified using XML to be replaced later by web interfaces) which indicate the resources required by the application (from Application developers), final configuration including all middle boxes (from Application Architects), and Workflow instantiation guidelines (from Application Deployment Managers). The OpenADN Platform computes the required virtual resources and creates instructions that are passed on to the various cloud management systems.

The OpenADN Service Workflow (ASW) abstraction allows the application architects to specify how the traffic flows over the application delivery network. The architect can specify the service chain - chain of the application modules (servers and middle boxes) that the application traffic will follow. They can also specify the application level policy routing (APR) giving rules for classifying packets or messages and routing them through various modules.

The OpenADN Application Cloud (AAC) abstraction allows the deployment administrators to specify policies for creating/using/destroying resources over various clouds. The rules could be based on usage patterns and current load distribution of users. The rules also include those for handling failures and planned maintenance.

Note that the application developers, the architects, and the deployment managers have a different view of the workflows. For example while developing a simple web server application, the developer's view consists of a client and server module communicating through a socket interface. The developers have no idea of the middle boxes that may be inserted later by the application architects. Similarly, the architects do not know about the cloud systems on which these applications will be instantiated. That is the responsibility of the deployment manager, who balances the required performance and the associated costs with various cloud services.

OpenADN supports massively distributed application deployments using a combination of distributed data plane and a centralized control plane. The designs of these two planes are described in the next two sections. The application is managed from a central global manager shown on the top.

VIII. Design of The OpenADN Data Plane

The data plane consists of various virtual machines, storage units, and the network between them. The resources may belong to several different service providers. The data plane of OpenADN is called OpenADN. Although the data plane is fully distributed, OpenADN presents the abstraction of a single OpenADN Distributed Virtual Switch (ADVS) to ASPs as shown in Fig. 4. The switch has a varying number of ports to which various services, middle boxes, and network components can be connected. Any two modules that need to communicate are connected by the switch. The connection can be at the application message level or at the IP packet level. The switch is fully programmable and the connections are continuously programmed by the control plane to meet the requirements of the current context.

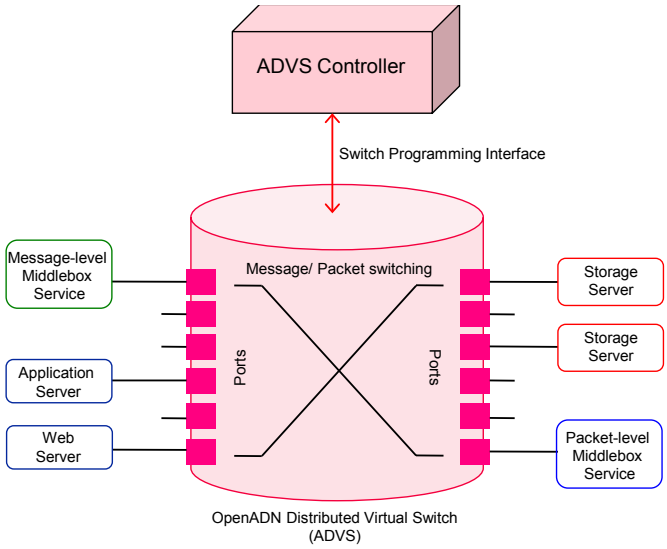

Fig. 4: OpenADN ADVS Abstraction

OpenADN introduces two shim layers: L3.5 between IP and transport layer, and L4.5 between transport and session/application layer. Packet level middle boxes and services are connected via L3.5 tunnels. Message level middle boxes and services are connected via L4.5 Tunnels. The tunnels are nested such that a L4.5 tunnel may have many intermediate L3.5 tunnel end points.

Fig. 5 shows an example of L3.5 and L4.5 Tunnels. In the figure, IDS is a packet-level middle box, while the transcoder is a message level middle box. The connection (tunnel) between the VM#1 and the transcoder is a L4.5 tunnel. Nested in this tunnel are two L3.5 tunnels: Tunnel 3a between VM#1 and IDS and Tunnel 3b between IDS and the transcoder. These tunnels may be implemented over VxLAN tunnels [7] commonly used in data centers. The VxLAN tunnels themselves may be over an MPLS transport profile (TP) in the case of inter-cloud wide-area networks.

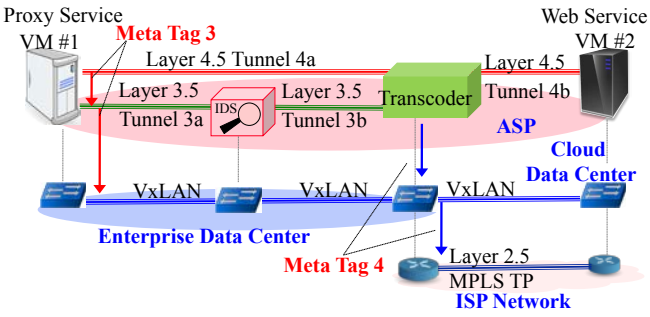

Fig. 5: Layer 3.5 and 4.5 Tunnels

Note that L3.5 and L4.5 meta tags (headers) can be interpreted only by OpenADN aware services and modules. Legacy OpenADN-unaware services are supported by providing "proxy" service modules that add/remove these shim headers before passing the packets to the legacy services.

IX. Design of the OpenADN Control and Management Plane

The control plane is hierarchical with at least two levels of hierarchy consisting of the global controller and a number of local controllers. There is a local controller for each cloud and the network connecting them. The global controller determines the desired resources and passes the information to the local controllers that negotiate the resource creation and deployment with the respective cloud or network service provider. Each of these local controllers can have another level of controllers to manage a group of VMs, storage, or network resources.

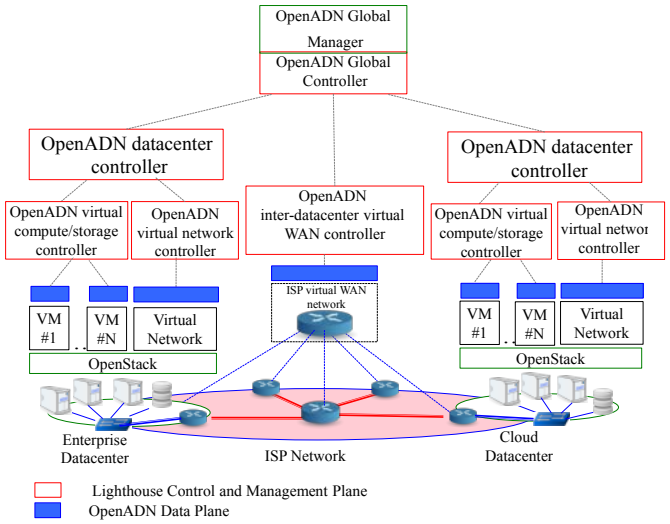

Fig. 6: Data, Control, and Management Planes of OpenADN

OpenADN allows multiple tenants to share the service chaining facilities offered by a service provider. Each tenant can convey its policies to its global controller which in turn can communicate with service provider controllers/management systems via local controllers. Note that the service provider in this case could be an ISP or a cloud service provider that provides a multi-cloud application delivery service.

One of the key features of OpenADN is that the cloud and Internet service providers keep complete control of their resources and the tenants have the flexibility to implement their policies on their virtual resources. The tenants have the flexibility to locate their middle boxes and VMs anywhere on the global internet where appropriate computing, storage, and networking resources are available.

X. OpenADN Prototype and Validation

OpenADN prototype has been implemented in C and Python and currently has over 10,000 lines of code. We have validated most of the features claimed above using an emulated Mininet [8] environment. While this environment is suitable for functional validation, it is not appropriate for performance measurement since the emulation overhead is difficult to isolate. So we are transferring the implementation to non-emulated physical environment suitable for conducting the performance benchmarking, finding bottlenecks, and optimizing most used parts of the code. A detailed design of the system and its benchmarking results are presented in [9].

XI. OpenADN Use Cases

The applications of this technology are numerous. This solution provides a new business opportunity for ISPs who can use the technology to provide smart elastic WAN services to their global clients. By providing WAN links that provide the capacity and QoS as needed and/or forwarding the flows as instructed by the ASP, ISPs can provide network services similar to the compute/storage services provided by the cloud service providers (CSPs) inside the cloud.

XII. Summary

The key messages of this paper are as follows:

1. The industry is moving towards a modular approach to SDN with multiple southbound protocols that allow a uniform implementation of policies on a variety of legacy and new network devices.
2. Network Function Virtualization (NFV) allows Internet Service Providers to use standard virtual machine implementations of various ISP functions, thus allowing them to get full advantage of cloud computing.
3. Service chaining of VMs distributed in multiple clouds has several challenges and significantly affects performance.
4. NFV concept can be extended to other non-ISP enterprises most of which use multiple clouds from multiple service providers.
5. The OpenADN platform allows an ISP or a non-ISP enterprise to use multiple clouds with different cloud management interfaces and automatically create/delete workflows as needed.

A brief overview of the OpenADN architecture was presented in this paper. OpenADN provides uniform and automated policy implementation over multiple clouds just as SDN provides policy implementation inside a cloud/data center. OpenADN interfaces with both cloud management systems and network SDN controllers (e.g., OpenDaylight).

XIII. References

[1] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. OpenFlow: enabling innovation in campus networks. ACM SIGCOMM Computer Communication Review, 38(2):69-74, 2008.

[2] M. Smith, M. Dvorkin, Y. Laribi, V. Pandey, P. Garg, N. Weidenbacher, "OpFlex Control Protocol," IETF draft-smith-opflex-00, April 2, 2014.

[3] Linux Foundation,"Opendaylight," http://www.opendaylight.org/ [Online; accessed July 20, 2014]

[4] Open Networking Foundation, "OpenFlow switch specication version 1.4.0, https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.4.0.pdf [online; accessed July 20, 2014]

[5] ETSI, "NFV – Update White Paper," Oct 2013, http://www.tid.es/es/Documents/NFV_White_PaperV2.pdf [Online accessed July 20, 2014]

[6] Right Scale, "Cloudcomputing trends: 2014 State of the Cloud Survey," April 2, 2014, http://www.rightscale.com/blog/cloud-industry-insights/cloud-computing-trends-2014-state-cloud-survey [Online accessed July 20, 2014].

[7] M. Mahalingam, D. Dutt, K. Duda, et al., "Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks," IETF RFC 7348, August 2014, 22 pp.

[8] Bob Lantz, Brandon Heller, and Nick McKeown, "A network in a laptop: rapid prototyping for software-defined networks," In Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks (Hotnets IX), 2010, Article #19.

[9] S. Paul, G. Vaszkun, R. Jain, et al., "OpenADN: A Platform for Next Generation Application Delivery over Multi-Cloud Environments," Submitted to IEEE Transactions on Cloud Computing, October 2014, 30 pp.