2018 IEEE/ACM 11th International Conference on Utility and Cloud Computing (UCC)

# Enhancement of LivCloud for live cloud migration

Ibrahim Ejdayid A.Mansour
Dept. of Computing & Informatics
Bournemouth University
Bournemouth, UK
imansour@bournemouth.ac.uk

Hamid Bouchachia
Dept. of Computing & Informatics
Bournemouth University
Bournemouth, UK
abouchachia@bournemouth.ac.uk

***Abstract*—Virtualization techniques aim at handling the growing demand for computing, storage and communication resources in cloud computing. However, cloud providers often offer their own proprietary virtualization platforms. As a result, cloud users' VMs are tightly coupled to providers' IaaS, hindering live migration of VMs to different providers. A number of live cloud migration approaches have been proposed to solve this coupling issue. Our approach, named LivCloud, is among those approaches. It is designed over two stages, basic design stage and the enhancement stage. The implementation of the basic design has been introduced and evaluated on Amazon EC2 and Packet bare metal cloud. This paper discusses the implementation of the second stage, the enhancement of the basic design on Packet. In particular, it illustrates how LivCloud is implemented in two different scenarios. The first scenario deploys KVM bridge networking, OpenvSwitch and C scripts used to meet the network configuration changes during the VMs relocating. This scenario achieves better downtime of one second compared to the basic design of LivCloud. The second scenario uses OpenVPN, OpenDayLight (ODL) and Cisco OpenFlow Manager (OFM) to successfully live migrate VMs back and forth between LivCloud and Packet. This scenario achieves better downtime between 400 and 600 milliseconds. As part of the discussion, the paper proposes a third potential scenario to successfully meet the live cloud migration requirements. This scenario aims to eliminate any downtime occurred in the first two scenarios by utilizing the Open Overlay Router (OOR), Locator Identifier Separator Protocol (LISP) and ODL.**

***Keywords*—Virtualization; Virtual Machine; Network Virtualization; Nested Virtualization; Live Cloud Migration; Cloud infrastructure (IaaS); Software Defined Networking (SDN)**

## I. Introduction

There is a growing trend in adopting cloud computing services. In 2017, RightScale conducted cloud computing trends survey in which 1,002 IT professionals at large and small enterprises were interviewed about their adoption of cloud infrastructure and related technologies [27]. 85 percent of enterprises deploy multi-cloud services, up from 82 percent in 2016. On the other hand, private cloud adoption decreased from 77 percent to 72 percent as enterprises focus more on public cloud services. Despite the notable upwards trend, there are still concerns about cloud computing security, interoperability and managing cost [3], [27]. On the other hand, the security concerns fell from 29 to 25 percent in comparison with 2016. Moreover, according to [13], those issues are not the only future challenges to cloud computing, but also: (i) scalability and elasticity, (ii) resource management and scheduling, (iii) reliability, (iv) sustainability and (v) heterogeneity.

Every provider have been developing their own APIs and proprietary features to their selected hypervisor. This has made it difficult for cloud users to live-migrate VMs to other providers - one aspect of vendor lock-in with substantial consequences [28], [32].

Live migration across the Internet takes a notable amount of time due to transferring the storage, limited Internet bandwidth, traffic re-routing, faulty behavior of Internet links and IP address management [7], [33]. It must keep the existing connections of the migrated VM to other VMs and cloud users. As a result, the live migration process can maintain the continuity of delivering the hosted services on migrated VMs. Various approaches from industry and academia have been proposed to improve live cloud migration of VMs at cloud IaaS [3], [32]. The implementation of those solutions are still challenging because they are implemented on top of uncontrolled public cloud IaaS [30]. As a result, a number of approaches succeeded to overcome virtualization heterogeneity by devising Software Defined Networking (SDN) and nested virtualization [4], [28]. However, they suffer limitations in terms of flexibility (decoupling VMs from underlying hardware), performance (migration downtime) and security (secure live migration). Our proposed live cloud migration, LivCloud, considers these three criteria as critical. It is designed over two stages, the basic design and its enhancement of the basic design. The basic design has been implemented and evaluated in a previous paper [2].

The basic design evaluation outperforms a number of previous approaches in terms of security and the migrated VMs hardware specifications (RAM & virtual disk sizes) despite its relatively acceptable performance (downtime of 2 seconds).

In this paper, the enhancement of the basic design is introduced and evaluated by conducting live cloud migration in two different scenarios. Despite both scenarios achieve better downtime than the basic design stage, Dynamic DNS and a script written in C are still needed to successfully finish the process. As a result, a third potential scenario is proposed to tackle these limitations of the first two scenarios by:

1) Using IPsec VPN and OpenvSwitch (OvS).
2) Using OpenVPN Ethernet Bridging [15], OvS, Cisco OpenFlow Manager (OFM) [10] and OpenDayLight (ODL) controller [14].

3) Introducing ODL, OvS, Locator Identifier Separator Protocol (LISP) [9] and Open Overlay Router (OOR) [4].

With respect to security, IPsec VPN is used for the first time in such an environment and it has no effect on performance. A study in [1] shows that fully live migrating VMs with their virtual disks and large RAM is still an ongoing effort to tackle instability and performance. Hence, the next step is implementing LivCloud using LISP, ODL, OvS and OOR.

Before discussing the three scenarios in more detail, we highlight the structure of the rest of the paper. Section II introduces a brief summary of related work highlighting existing approaches to achieve live cloud migration. Section III presents LivCloud's architecture that covers the enhancement of the basic design. It also highlights the experimental setup. Section IV discusses the implementation of the two live cloud migration scenarios on Packet and the empirical results of the experiments. In Section V, a third potential scenario is introduced to successfully meet live cloud migration requirements. Future work and conclusion are presented in Section VI.

## II. Related Work

The literature review reveals that there are a number of approaches that aim to achieve live cloud migration using SDN technologies such as OpenFlow protocol. In [16], an SDN architecture named, LIME, is introduced to live-migrate VMs and virtual switches. It is built on Floodlight controller. It simultaneously runs and clones the virtual switches on multiple physical switches. If this process is not implemented correctly, it may lead to services corruption. This architecure needs the provider's agreement to be implemented on top of public cloud IaaS. In [28], an interesting approach is introduced which is implemented on top of a number of cloud providers, including Amazon EC2, Rackspace and HP Cloud. It uses nested virtualization (Xen-Blanket [28]) that copes with cloud heterogeneity. Xen-Blanket leverages the paravirtualization (PV-on-HVM) drivers on Xen, which cannot run unmodified operating systems (i.e., Windows) [28]. The approach achieves relatively acceptable performance, about 1.4 seconds migration downtime [34]. It is claimed that OvS was used in, but without any details.

Another approach in [5] proposes an open LISP implementation for public transportation based on Open Overlay Router with an SDN controller, OpenDayLight. This approach is implemented on an emulated environment, GNS3 [6]. The real challenge is how to implement such design on uncontrolled environment, such as Amazon EC2 because the provider's networking system is highly complicated [2]. Also, networks are hard to manage because their configurations change during VMs re-instantiation on the new location. In [7], Migration of a VM cluster is suggested to various clouds based on different constraints such as computational resources and better economical offerings. It is designed based on SDN OpenFlow protocol and allows VMs to be paired in cluster groups that communicate with each other independently of the cloud IaaS. It separates the VM internal network from the cloud IaaS network. Consequently, VMs can be migrated to different clouds overcoming network complexity such as static IPs. The design also adopts SDN architecture for rerouting traffic when VMs relocation migration occurs. The design is evaluated on OpenStack environment.

In [18], an IaaS framework with regional datacenters for mobile clouds is presented. It is designed based on software-defined networking (SDN) to address the network bandwidth consumption during migration. The framework is simulated and evaluated in Mininet-based test environment [17]. Implementing such a design can be more challenging on uncontrolled environments. Finally, virtual network migration is designed and tested on the Global Environment for Networking Innovation (GENI) [19], [20] which is Wide-Area SDN-enabled infrastructure. The migration in this study is the process of remapping the virtual network to the physical network to dynamically allocate the resources during migration, manage hosts connected to the virtual network and minimize packet loss. However, maintaining transparent migration to the users and the running applications is still challenging.

## III. LivCloud architecture

The LivCloud design is distilled into two stages: basic design and the enhancement of the basic design [1]. The basic design stage helps connecting the local network to the cloud IaaS through *nested virtualization* and *secure network connectivity*. Firstly, nested virtualization is achieved by configuring QEMU-KVM on the local network and public cloud IaaS. Nested virtualization is configuring one hypervisor (in the upper layer) within a virtual machine hosted on another hypervisor [35]. It is known of low perofmrnce, but the high hardware specifications of today's servers overcome this issue [26]. Most of legacy hypervisors, such as QEMU-KVM, Xen and VMware can run nested virtualization [33]. LivCloud uses QEMU-KVM as a hypervisor on both sides. Virtual machine manager is a user interface for managing virtual machines mainly on QEMU-KVM. Any physical or virtual machine that has QEMU-KVM configured can be connected locally or remotely over SSH to virtual manager [1]. The basic design has been implemented and tested [2].

At this development stage, an enhancement of basic design of LivCloud is implemented. It deploys various technologies such as OpenDayLight (ODL), OpenFlow and LISP protocols to:

1) Enhance network throughput.
2) Maintain VMs connections and configurations.
3) Reserve resources and prediction of potential failure.

Figure 1 shows the final configurations of LivCloud. Live cloud migration is implemented and evaluated in Scenario 1 and Scenario 2. The next section explains these scenarios in more detail. Both scenarios are built and tested on a general experimental setup that can be distilled as follows:

1) QEMU-KVM is enabled on the local network and public cloud IaaS. QEMU-KVM supports running modified and unmodified OS. QEMU has high emulation capability of drivers (i.e network card driver) and KVM provides high

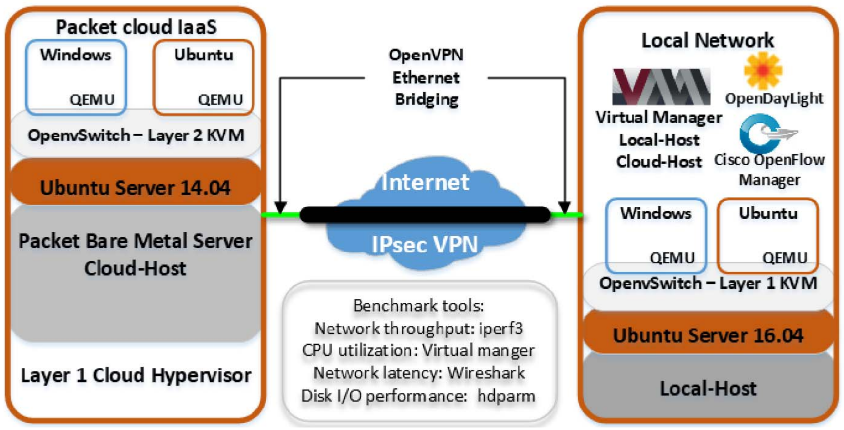


Fig. 1: The final configuration of LivCloud [1]

acceleration to enhance drivers performance. Also, KVM needs to access the underlying CPU architecture to pass it to the virtualized CPU of the hosted VMs [1], [29].

2) IPsec VPN tunnel is configured to secure the migration. The secure connection between local network and Packet's network is an essential part of live cloud migration.
3) Both sides are connected to Virtual Machine Manager (VMM) [25] in order to live migrate VMs between the local network and cloud IaaS.
4) Both sides are connected to the shared storage on the local network.
5) Dynamic DNS is used to maintain the migrated VM's connections and configurations. Dynamic DNS is used to keep a domain name pointing to the same physical or virtual server connected to the Internet regardless of any IP addresses changes [11].

## IV. Live cloud migration scenarios

Two different live cloud migration scenarios are implemented and evaluated in this section. These scenarios are chosen to cover the potential solutions of live cloud migration. These solutions may help cloud users live migrate their VMs with very low costs. The technologies used in the approach are either open-source or very low cost.

### A. The general experimental setup

This setup is used in both scenarios and some elements may be added or removed accordingly. It can be used at this level or at a larger size with respect to the number of servers and virtual machines. To implement the general setup, a local network (172.16.10.0/24) based in Bournemouth (UK) which has two physical servers (Local-Host and NFS server) is connected to a Ubuntu server (Cloud-Host) 14.04 (private address, 172.20.20.0/24) on Packet's datacenter in Frankfurt (Germany). Moreover, Network throughput, CPU utilization, network latency, migration downtime and disk I/O performance are the main parameters used to analyze the live migration impact. Network throughput is measured using iPerf [21], while network latency is measured by pinging the migrated VM's DNS record. Disk I/O performance is tested on Local-Host and Cloud-Host using *hdparm* command [23]. If any downtime happens during the process, Wireshark is used to calculate it [22].

Packet Bare Metal Cloud provides customers with dedicated single tenant-physical servers [26]. The bare metal server complements or substitutes virtualized cloud services with a dedicated server that eliminates the overhead of virtualization, but maintains flexibility, scalability and efficiency [26]. Figure 2 shows the enhancement implementation on Packet. The lab setup as shown in Figure 2 consists of one HP Z440 workstation, Local-Host is connected to the Internet through EdgeRouter X and Netgear L2 switch providing 1 Gbps. The workstation has 32 GB of RAM, 1TB disk and 4-core 2.8GHz Intel(R) Xeon(R) E5-1603 v3 CPU. 64-bit Ubuntu Server 16.04 LTS, QEMU-KVM (Layer 1 hypervisor), OpenvSwitch (OvS) and QEMU-KVM bridged networking are installed and configured on the machine [1]. OvS has flow classification, caching and better performance over the traditional Linux Bridge. Moreover, it has its own load balancer which is used to distribute loads across available routes [8].

The other machine on the private network is configured as NFS server (FreeNAS 9.3) for the lab. The Packet 64-bit Ubuntu TYPE 1E server 14.04, Cloud-Host is connected through two bonded network cards providing 20 Gbps. The server has 32 GB of RAM, 240 GB disk and 4-physical core 2.0GHz/3.4GHz burst Intel E3-1578L v3 CPU. By default nested virtualization or hardware-assisted virtualization features (Intel VT-x, Intel VT-d and Extended Page Tables) are enabled on any Packet server [26]. QEMU-KVM (Layer 2 hypervisor), OpenvSwitch (OvS) and QEMU-KVM bridged networking are installed and configured on the server.

Packet offers various types of bare metal servers including, Type 1 and Type 1E servers which both have similar hardware specifications as specifications of Local-Host [26]. As a result, the live migration has no issues in terms of hardware architecture. Previously, Type 1 in Packet's datacenter in

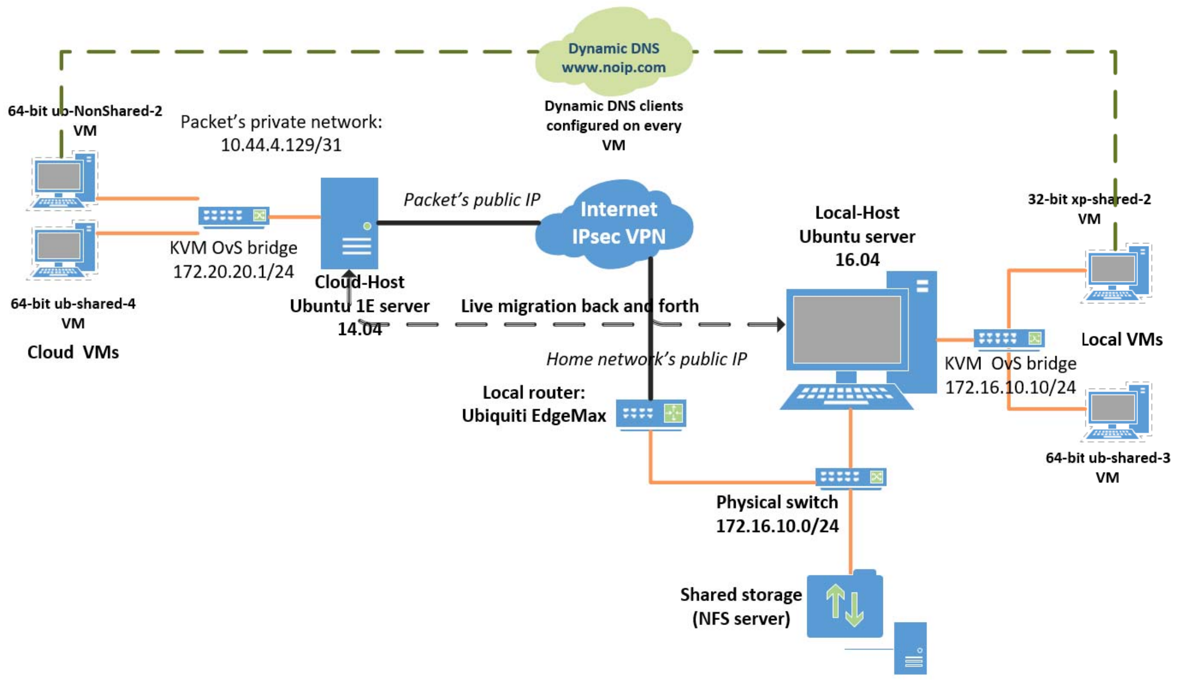

Dynamic DNS
www.noip.com
Dynamic DNS clients configured on every VM
64-bit ub-NonShared-2 VM
Packet's private network: 10.44.4.129/31
Packet's public IP
Internet
IPsec VPN
Local-Host Ubuntu server 16.04
32-bit xp-shared-2 VM
KVM OvS bridge 172.20.20.1/24
64-bit ub-shared-4 VM
Cloud-Host Ubuntu 1E server 14.04
Live migration back and forth
Local VMs
Cloud VMs
Home network's public IP
KVM OvS bridge 172.16.10.10/24
Local router: Ubiquiti EdgeMax
64-bit ub-shared-3 VM
Physical switch 172.16.10.0/24
Shared storage (NFS server)

Fig. 2: The enhancement implementation on Packet

Amsterdam, Holland, was used in implementing the basic design of LivCloud. However, KVM NAT networking had to be configured instead of the bridged option. Packet does not allow layer 2 networking in this type. This made the migration process more complicated in terms of VMs' networking and IPsec VPN configurations.

In this paper, Type 1E is deployed and KVM bridge is possible thanks to the configuration with spare Ethernet network card (eth1) [26]. Layer 2 bridge is implemented through this interface and the cloud private network (172.20.20.0/24) is installed as shown in Figure 2. Many configurations are carefully considered including PAT behind the server's public IP and enabling IPv4 forwarding to have the bridge functions correctly.

Any VM on either Local-Host or Cloud-Host can be configured with a local disk or a disk hosted on the local network NFS server. The local network and the Packet private network are securely connected via IPsec VPN tunnel. Local-Host and Cloud-host are connected through the tunnel via the virtual machine manager that is installed on Local-Host. In the case of Local-Host being temporarily not accessible, both hosts can still be connected via the virtual machine manager installed on Cloud-Host. A remote Ubuntu desktop is installed on Cloud-Host using VNC server (vnc4server) and through TightVNC, cloud users can be remotely connected to Cloud-Host [36].

At this setup, *Dynamic DNS* is used to maintain the migrated VMs' connections and configurations. Dynamic DNS is used to keep a domain name pointing to the same physical or virtual server connected to the Internet regardless of any IP addresses changes [11]. no.ip is a dynamic DNS provider that is chosen to register the DNS records. Dynamic DNS clients (noip-2.1.9-1) are installed and configured on all migrated VMs [11]. Dynamic DNS records are used to maintain the existing connections to the migrated VMS. VMs' dynamic DNS records are registered on the dynamic DNS provider, noip [11] associated with either the public IP of Local-Host or Cloud-Host. Once the migration to the host is completed, the dynamic DNS client installed on the migrated VMs updates the provider with this host's public IP, so that the name records are updated accordingly. The other VMs and the cloud users are connected to these records not to the IP addresses. Therefore, any changes of public and private IP addresses, the DNS client updates the records accordingly. Table I shows the migrated VMs' specifications, VMs' architecture and associated DNS names. As far as the related literature is concerned, the VMs' specifications are the highest in this environment.

TABLE I: Migrated VMs' specifications and DNS names

| DNS records | VM's Architecture | vCPU | RAM (GB) | Virtual disk (GB) | Shared disk/ non-Shared |
|---|---|---|---|---|---|
| ub-NonShared-2.ddns.net | 64-bit | 2 | 2 | 12 | Non-Shared |
| ub-shared-2.ddns.net | 64-bit | 2 | 2 | 15 | Shared |
| ub-shared-3.ddns.net | 64-bit | 2 | 3 | 15 | Shared |
| ub-shared-4.ddns.net | 64-bit | 2 | 4 | 15 | Shared |
| xp-NonShared-2.ddns.net | 32-bit | 2 | 2 | 10 | Non-Shared |
| xp-shared-2.ddns.net | 32-bit | 2 | 2 | 15 | Shared |
| xp-shared-3.ddns.net | 32-bit | 2 | 3 | 15 | Shared |

## *B. Scenario 1:*

The general setup described in Section IV-A is used in this scenario without adding any technology to successfully live migrate the VMs mentioned in Table I. QEMU-KVM supports live migration with different networking options, including bridged network and NAT network. Bridge network has successfully been implemented as mentioned in Section IV-A. Packet offers various server types including 1E server that its networking setup allows OpenvSwitch and the bridge configurations. QEMU-KVM live migration copies the RAM and CPU states over a number of iterations while the OS and the applications are running. This means the drivers' states, such as network cards (NICs) stay as they are on the sender side [30]. The migrated VMs' NICs are configured to request IP addresses from the NAT's DHCP server. During the migration, the VMs' NICS need to be triggered to renew their IPs on the receiver's network. To this end, we have written a script in C language to be run on Windows or Linux to enable the following:

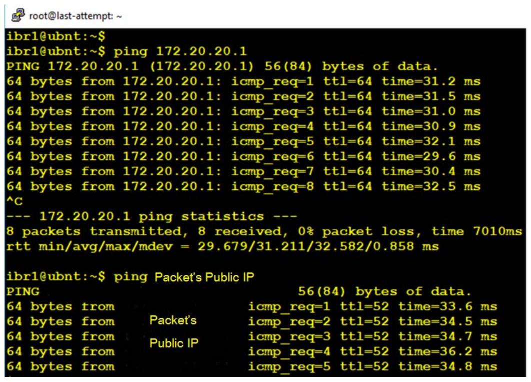
1) Continuously testing the Internet connectivity by pinging Google server (8.8.8.8). If connectivity is maintained, the script does nothing.
2) If the connectivity is lost, the script forces the migrated VM to renew the IP address and trigger the dynamic DNS client to update the VM's record on the noip.

The script has the following structure:

**Algorithm 2** Steps of C script in Scenario 1

```
Input:
while (true) do
    Sleep (T)
    if connection to 8.8.8.8 is false then
        if (Operating System is Windows) then
            - Trigger the network card to renew its IP address
            - Re-run Dynamic DNS client
        else if (Operating System is Unix) then
            - Trigger the network card to renew its IP address
            - Re-run Dynamic DNS client
        end if
    end if
end while
```

The total migration time varies because of the VM's hardware specifications and the Internet traffic. For example, live migrating the Ubuntu VM (ub-shared-4: 4GB RAM & 2 vCPU) takes on average about *7 minutes* in terms of migration time. The XP VM (xp-shared-2: 2 GB RAM & 2vCPU) takes about *3 minutes*. Unfortunately, the migration process does not yield the desired results in case of xp-shared-3 and xp-NonShared-2. However, the migration downtime in other VMs migration is just under *one second* due to the latency in updating the public IP and the DNS records. Due to the extra overhead processing and migration downtime added by security mechanism, such as IPsec to live migration, it has been avoided in many live cloud migration approaches. The downtime is increased about 5 times when IPsec is added to live migration as in [31]. The study illustrates the increase of both migration downtime and total time migration, from less than 2 seconds to almost 8 seconds downtime when IPsec VPN is implemented. However, by comparing a direct ping



```
ibr1@ubnt:~$
ibr1@ubnt:~$ ping 172.20.20.1
PING 172.20.20.1 (172.20.20.1) 56(84) bytes of data.
64 bytes from 172.20.20.1: icmp_req=1 ttl=64 time=31.2 ms
64 bytes from 172.20.20.1: icmp_req=2 ttl=64 time=31.5 ms
64 bytes from 172.20.20.1: icmp_req=3 ttl=64 time=31.0 ms
64 bytes from 172.20.20.1: icmp_req=4 ttl=64 time=30.9 ms
64 bytes from 172.20.20.1: icmp_req=5 ttl=64 time=32.1 ms
64 bytes from 172.20.20.1: icmp_req=6 ttl=64 time=29.6 ms
64 bytes from 172.20.20.1: icmp_req=7 ttl=64 time=30.4 ms
64 bytes from 172.20.20.1: icmp_req=8 ttl=64 time=32.5 ms
^C
--- 172.20.20.1 ping statistics ---
8 packets transmitted, 8 received, 0% packet loss, time 7010ms
rtt min/avg/max/mdev = 29.679/31.211/32.582/0.858 ms

ibr1@ubnt:~$ ping Packet's Public IP
PING                              56(84) bytes of data.
64 bytes from                  icmp_req=1 ttl=52 time=33.6 ms
64 bytes from    Packet's      icmp_req=2 ttl=52 time=34.5 ms
64 bytes from    Public IP     icmp_req=3 ttl=52 time=34.7 ms
64 bytes from                  icmp_req=4 ttl=52 time=36.2 ms
64 bytes from                  icmp_req=5 ttl=52 time=34.8 ms
```

Fig. 3: A direct ping latency & IPsec VPN latency

through the Internet to Cloud-Host's public IP and ping to Cloud-Host's private IP (172.20.20.1) through the IPsec tunnel from Local-Host, the round trip time (RTT) is almost identical in the first and the second scenarios. In fact, the connection through the tunnel is slightly faster. Figure 3 shows A direct ping latency & IPsec VPN latency.

### C. Scenario 2:

We update the general setup with the following technologies, **OpenVPN** [15], **Cisco OpenFlow Manager (OFM)** [10] and **Zodiac-FX OpenFlow** switch [24]. Figure 4 shows the changes made in this scenario. OpenVPN has the ability to extend one network across multiple sites (Ethernet bridging) [15], [28]. The local network (172.16.10.0/24) is extended to the cloud network, so the migrated VM has an IP address within the local network range. OFM is connected to OpenDaylight controller through RESTCONF API [10] to re-route the migrated VM internally and Dynamic DNS is used to re-route it externally. This scenario uses OpenVPN tunnel instead of IPsec tunnel. The local network (172.16.10.0/24) is extended using OpenVPN across to Packet's private network (172.20.20.0/24) using **TAP interface** [15]. Zodiac switch is added to the general topology to configure OF protocol. Zodiac switch is connected to ODL [24]. Then, OFM is connected to ODL using RESTCONF API which is an application developed by Cisco to run on top of ODL. IT visualizes OpenFlow topologies, its program paths and gather its stats [10]. Figure 5 shows how OFM is connected to ODL.

By configuring OFM, any changes of VMs or hosts location can be re-routed internally through Zodiac switch. However, Dynamic DNS is still needed to re-route the VMs' location to external users. Also, during the migration, the VMs' NICS and OpenVPN client file need to be triggered to renew their IPs on the receiver's network and update the OpenVPN configurations. This requires the modification of the C script used in Section IV-B to yield the desired results.

During the evaluation process, OpenVPN bridging, OFM and the modified script are proved to function slightly better than the previous scenario. For example, live migrating the Ubuntu VM (ub-shared-3: 3GB RAM & 2 vCPU) takes on

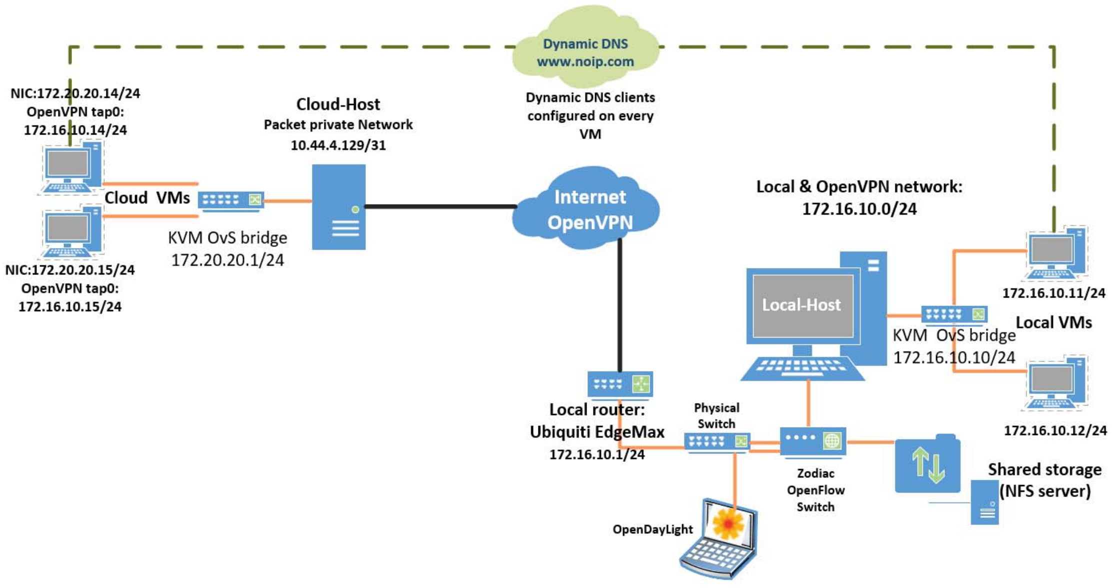


Fig. 4: The enhancement implementation on Packet using OpenVPN

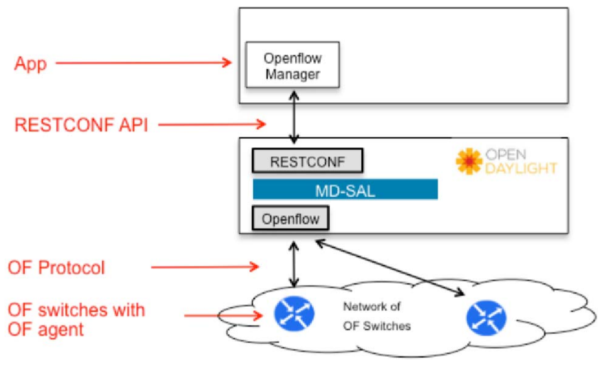


Fig. 5: The connection between OFM and ODL [10]

average about *5 minutes* in comparison to 7 minutes in the scenario 1. The XP VM (xp-shared-2: 2 GB RAM & 2vCPU) takes about the same time as the scenarion 1, *3 minutes*. Similar to the scenario 1, the migration process does not yield the desired results when live migrating xp-shared-3 and xp-NonShared-2. However, the migration downtime in other VMs migration mentioned in Table I is between *400 and 600 milliseconds* due to the latency in updating the public IP and the DNS records. The downtime is about 1 second in Scenario 1. Moreover, OpenVPN Bridging has limitations in terms of scalibility and Maximum Transmission Unit (MTU) tuning [15], [30]. The updated version of the script has the following structure:

## D. Simulation results

These results are the average of conducting the experiment of a total of 15 runs. In terms of the experiment times, it is done during the morning, afternoon and during the night.

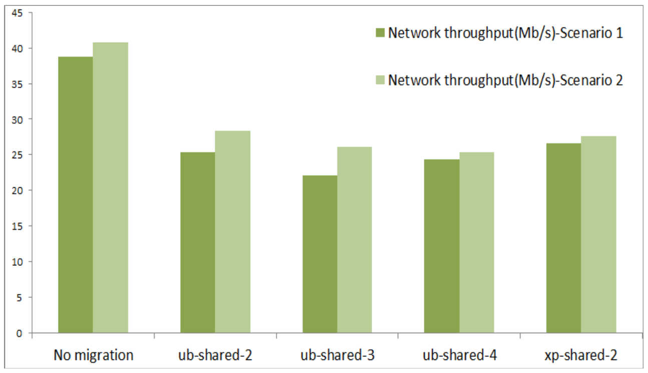
**Algorithm 4** Steps of C script in Scenario 2

```
1:  Input:
2:  while (true) do
3:      Sleep (T)
4:      if connection to 8.8.8.8 is false then
5:          if (Operating System is Windows) then
6:              - Trigger the network card to renew its IP address
7:              - Re-run OpenVPN client
8:              - Re-run Dynamic DNS client
9:          else if (Operating System is Unix) then
10:             - Trigger the network card to renew its IP address
11:             - Re-run OpenVPN client
12:             - Re-run Dynamic DNS client
13:         end if
14:     end if
15: end while
```

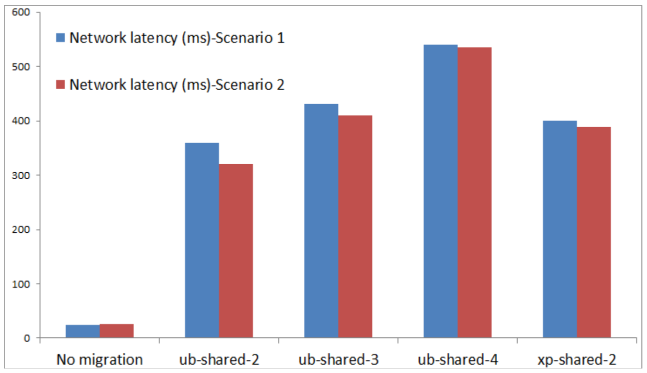
First, we compare Scenario 2 against Scenario 1 with respect to network throughput, network latency, CPU overhead and disk I/O performance.

Then, live migration of the Ubuntu VMs and XP VMs mentioned earlier is performed back and forth between Local-Host and Cloud-Host in both scenarios. Only the most notable statistics are summarized in Figure 6. In summary, deploying OpenVPN Bridging is proved to outperform using only IPsec tunnel in all evaluation aspects. Figure 6(a) shows that the network throughput is considerably affected in the first scenario than the second scenario. In the first scenario, when migrating ub-share-3 VM that has 3GB RAM, the network throughput VM is more affected than ub-shared-4 that has 4GB RAM. It is most likely due to the Internet congestion at that time.

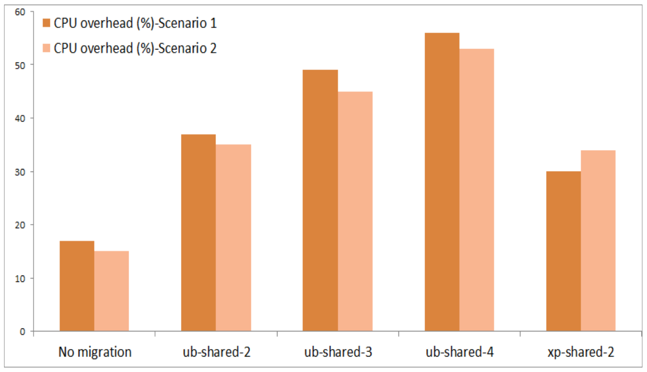
In the second scenario, when the VM's hardware size is



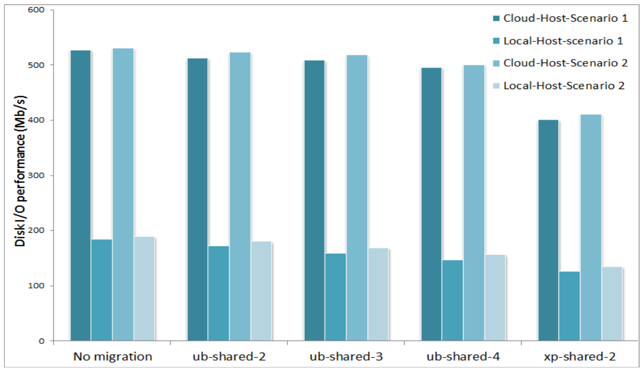
(a) Network throughput



(b) Network latency



(c) CPU overhead



(d) Disk I/O performance

Fig. 6: Results statistics

larger the network throughput decreases. In particular, Figure 6(b) shows that there is notable increase in network latency during live migration ub-shared-4 VM in both scenarios because this VM has the largest RAM size, 4GB. The total migration time reaches about 7 minutes in the first scenario and 5 minutes in the second. In case of ub-shared-2 & 3, the network latency is fairly better in the second scenario than the first one.

Figure 6(c) shows that CPU load increases by about 39% in Scenario 1 and by 38% in the second one during live migration ub-shared-4. Figure 6(d) shows that I/O performance of disks of Local-Host and Cloud-Host are slightly effected by the migration process in both scenario. However, It is affected more by the first scenario than the second one.

As mentioned earlier, in Scenario 1 there is downtime of roughly 1s during live migration back and forth between the two hosts. The downtime in the second scenario is between 400 to 600 milliseconds, which means using OpenVPN bridging is slightly faster.

## V. Discussion

In this section, we discuss the limitations of the first two scenarios and propose a third potential scenario that copes with these limitations.

### A. *The first two scenarios limitations*

As discussed in the first two scenarios, to successfully finish the live migration, a number of steps have to be considered including Dynamic DNS, the C script and OpenVPN. Yet, there are still challenges to VMs relocating and migration downtime. In a nutshell, we have to implement the following steps to achieve successful migration and maintain the downtime as low as possible:

1) Configuring Dynamic DNS on the migrated VMs and the DNS provider to maintain the external and the internal connections to other VMs and cloud users. As shown in Section IV-D, there is still downtime in both scenarios because of updating and propagating any change in Dynamic DNS records.
2) Using the C script to cope with any change in network configurations, help update DNS name records and re-initiate OpenVPN in Scenario 2. These changes should have dynamically happened without any script.

Based on these limitations, this paper discusses a potential scenario that copes with any of these challenges. In the following section, this scenario is discussed in more detail.

### B. *Scenario 3*

To improve LivCLoud downtime and cope with VMs relocation, an alternative scenario is being investigated. This scenario adds to the general setup Open Overlay Router (OOR) that can be configured to run Locator Identifier Separator Protocol (LISP), OvS, ODL and Cisco OFM. OOR, which is an open source software router to deploy programmable overlay networks. OOR runs LISP to map overlay identifiers

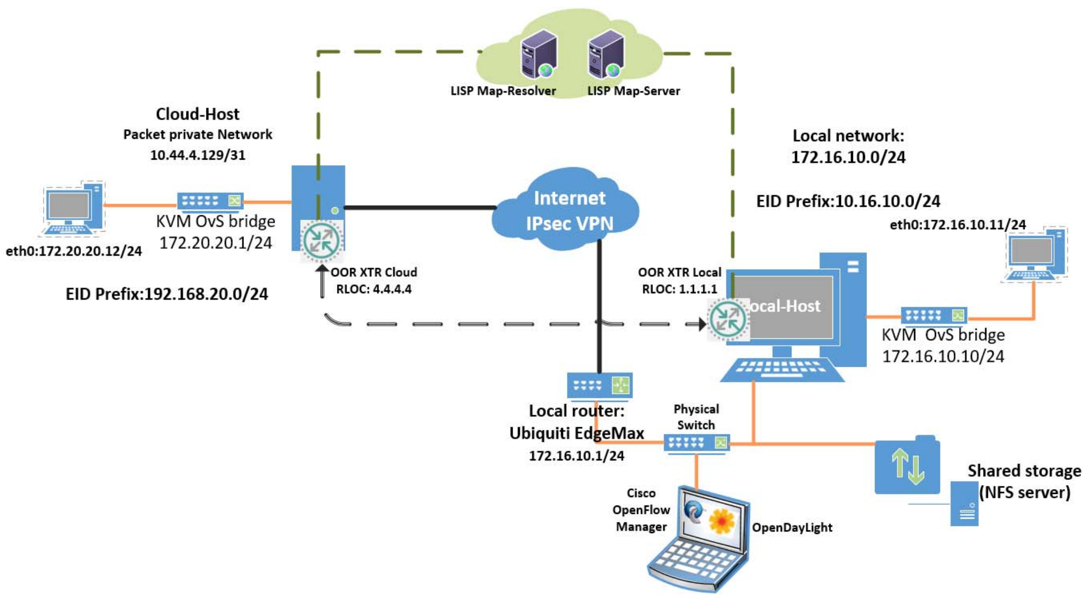


Fig. 7: The potential solution on Packet

to underlay locators and to dynamically tunnel overlay traffic through the underlay network [4]. Figure 7 shows the scenario design.

LISP creates two different namespaces: endpoint identifiers (EIDs) and routing locators (RLOCs). Each host is identified by an EID, and its point of attachment to the network by an RLOC. Traffic is routed based on EIDs at LISP sites and on RLOCs at transit networks. At LISP site edge points, ingress/egress tunnel routers (xTRs) are deployed to allow transit between EID and RLOC space.

LISP follows a map-and-encap approach. EIDs are mapped to RLOCs and the xTRs encapsulate EID packets into RLOC traffic. LISP introduces a publicly accessible Mapping System, which is a distributed database containing EID-to-RLOC mappings. The Mapping System consists of both Map-Resolvers (MRs) and Map-Servers (MS). Map-Servers store mapping information and Map-Resolvers find the Map-Server storing a specific mapping [9].

OpenDayLight controller can use the northbound REST API to define the mappings and policies in the LISP Mapping Service. OOR can leverage this service through a southbound LISP plugin. It must be configured to use this OpenDayLight service as their Map Server and/or Map Resolver. The southbound LISP plugin supports the LISP control protocol (Map-Register, Map-Request, Map-Reply messages) and can also be used to register mappings in the OpenDayLight mapping service [12]. Each VM is assigned an EID, which represents the private IP address and RLOC which represents the public IP address. When the migration occurs the RLOC is maintained and the EID is updated through ODL LISP Mapping Service, MRs and MS servers.

The OOR configuration includes setting up two overlay networks, EID prefix (10.16.10.0/24) on the local network and EID prefix (192.168.20.0/24) on the cloud network. At this development stage, Packet's architecture does not allow configuring those prefixes. The configurations need flexibility in layer 2 networking, which is not possible on Packet's datacenters in either Amsterdam or Frankfurt. Both datacenters are the closest to LivCLoud's location, Bournemouth, UK. Layer 2 networking is being considered in both centers very soon.

## VI. Conclusion and future work

LivCloud is designed to overcome the limitations of previously proposed live cloud migration approaches. The evaluation of enhancement design on Packet shows that live cloud migration can be improved by using various techniques such as, OpenVPN and Software Defined Network (SDN). Also, the evaluation shows the migrated VMs' RAM and disks sizes are larger than the previous stage of LivCloud and any previous approaches. Moreover, this stage performance outperforms any previous approaches. However, there is still improvement needed in maintaining the connectivity to the migrated VMs without using extra techniques such as Dynamic DNS. The migration downtime is most likely due to the time needed by Dynamic DNS to be propagated across both sites.

In a nutshell, this paper shows: (i) performing two successful live cloud migration scenarios; (ii) considering the migrated VM's architecture (32 or 64-bit) and hardware specifications, (iii) deploying ODL and OFM in such environment and (iv) using a customized script to dynamically change network configurations and re-run the OpenVPN.

The next step of running LivCloud on Packet is to implement and evaluate Scenario 3 that includes configuring LISP protocol on the OOR to eliminate the need for the customized script and Dynamic DNS. This scenario helps enhance the network throughput, maintain the connectivity to the migrated VMs and eliminate any disconnection between the cloud users and the migrated VMs by redirecting and re-routing the migrated VMs' new locations based on LISP and ODL LISP mapping feature.

## REFERENCES

[1] Ibrahim Mansour, Kendra Cooper and Hamid Bouchachia. "Effective Live Cloud Migration". In proceedings of The IEEE 4th International Conference on Future Internet of Things and Cloud (FiCloud 2016). DOI:10.1109/FiCloud.2016.54.

[2] Ibrahim Mansour, Hamid Bouchachia and Kendra Cooper. "Exploring Live Cloud Migration On Amazon EC2". In proceedings of The IEEE 5th International Conference on Future Internet of Things and Cloud (FiCloud 2017). DOI:10.1109/FiCloud.2017.20.

[3] Zhizhong Zhang, Chuan Wu, and David W.L. Cheung. 2013. "A survey on cloud interoperability: taxonomies, standards, and practice". In Proceedings of the ACM SIGMETRICS Performance Evaluation Review. DOI=10.1145/2479942.2479945.

[4] Alberto Rodriguez-Natal, Jordi Paillisse and Florin Coras. 2017. "Programmable Overlays via OpenOverlayRouter". In proceedings of IEEE Communications Magazine (Volume: 55, Issue: 6, 2017 ). DOI: 10.1109/MCOM.2017.1601056.

[5] T. Balan, D. Robu, and F. Sandu. 2016. "LISP Optimisation of Mobile Data Streaming in Connected Societies". In proceedings of Mobile Information Systems. http://dx.doi.org/10.1155/2016/9597579.

[6] Geraphic Network Simulator (GNS3). 2017. Available at: https://www.gns3.com/.(Accessed: 10-12-2017).

[7] Stelios Sotiriadis, Nik Bessis, Euripides G.M. Petrakis, Cristiana Amza, Catalin Negrud and Mariana Mocanud. 2017. "Virtual machine cluster mobility in inter-cloud platforms". In proceedings of Future Generation Computer Systems. DOI:https://doi.org/10.1016/j.future.2016.02.007.

[8] Ben Pfaff, Justin Petti, Teemu Koponen, Ethan J. Jackson, Andy Zhou, Jarno Rajahalme, Jesse Gross, Alex Wang, Jonathan Stringer, Pravin Shelar, Keith Amidon and Martin Casado. 2015. The Design and Implementation of Open vSwitch. In proceedings of the 12th USENIX Symposium on Networked Systems Design and Implementation (NSDI15). ISBN 987-1-931971-21.

[9] Alberto Rodriguez-Natal, Lorand Jakab and Vina Ermagan. 2015. "Location and identity privacy for LISP-MN". In proceedings of IEEE International Conference on Communications (ICC). DOI:10.1109/ICC.2015.7249159.

[10] Cisco OpenFlow Manager. 2017. Available at: https: //developer.cisco.com/site/devnetcreations/openflow-mgr.(Accessed: 09-12-2017).

[11] no.ip. 2017. Available at: https://www.noip.com/.(Accessed: 10-12-2017).

[12] LISP Flow Mapping User Guide. 2016. Available at: http://docs.opendaylight.org/en/stable-nitrogen/user-guide/lisp-flow-mapping-user-guide.html. (Accessed: 08-12-2017).

[13] Rajkumar Buyya, Satish Narayana Srirama, Giuliano Casale, Rodrigo Calheiros, Yogesh Simmhan, Blesson Varghese, Erol Gelenbe, Bahman Javadi, Luis Miguel Vaquero, Marco A. S. Netto, Adel Nadjaran Toosi, Maria Alejandra Rodriguez, Ignacio M. Llorente, Sabrina De Capitani di Vimercati, Pierangela Samarati, Dejan Milojicic, Carlos Varela, Rami Bahsoon, Marcos Dias de Assuncao, Omer Rana, Wanlei Zhou, Hai Jin, Wolfgang Gentzsch, Albert Zomaya and Haiying Shen. 2017. "A Manifesto for Future Generation Cloud Computing: Research Directions for the Next Decade". Published in Cornell University Library. Cited as arXiv:1711.09123v1 [cs.DC].

[14] OPENDAYLIGHT. 2018. Available at: https://www.opendaylight.org/. (Accessed: 05-01-2018).

[15] OpenVPN Ethernet Bridging. Available at: https://openvpn.net/index.php/open-source/documentation/miscellaneous/76-ethernet-bridging.html. (Access: 25-12-2017).

[16] Soudeh Ghorbani, Cole Schlesinger, Matthew Monaco, Eric Keller, Matthew Caesar, Jennifer Rexford and David Walker. 2014. "Transparent, Live Migration of a Software-Defined Network". In Proceedings of the ACM Symposium on Cloud Computing (SOCC '14). DOI:10.1145/2670979.2670982.

[17] Mininet. 2018. Available at: http://mininet.org/. (Accessed:08-01-2018).

[18] Wijaya Ekanayake, Heli Amarasinghe and Ahmed Karmouch. 2017. "SDN-based IaaS for Mobile Computing". In Proceedings of the 14th IEEE Annual Consumer Communications & Networking Conference (CCNC). DOI:10.1109/CCNC.2017.7983102

[19] Yimeng Zhao, Samantha Lo, Ellen Zegura, Mostafa Ammar and Niky Riga. 2017. "Virtual Network Migration on the GENI Wide-Area SDN-Enabled Infrastructure". Published in Cornell University Library. Cited as arXiv:1701.01702

[20] M.Berman, J.S.Chase, L.Landweber, A.Nakao, M.Ott, D.Raychaudhuri, R.Ricci and I.Seskar. 2014. "GENI: A federated testbed for innovative network experiments". Published in Special issue on Future Internet Testbeds Part I, ScienceDirect. DOI: https://doi.org/10.1016/j.bjp.2013.12.037

[21] iPerf3. 2017. Available at: https://iperf.fr/iperf-download.php. (Accessed: 18-12-2017).

[22] Wireshark. 2017. Available at: https://www.wireshark.org/ (Accessed: 17-12-2017) .

[23] hdparm. 2017. Available at: https://wiki.archlinux.org/index.php/hdparm (Accessed: 13-12-2017).

[24] Zodiac-FX OpenFlow Switch. 2017. Available at: https://northboundnetworks.com/products/zodiac-fx (Access: 11-12-2017).

[25] Virtual Machine Manager. Available at: https://virt-manager.org/. (Accessed: 13-12-2017).

[26] Packet. 2017. "The Promise of the Cloud Delivered on Bare Metal". Available at: https://www.packet.net/. (Accessed: 06-10-2017).

[27] RIGHTSCALE. 2017. "Cloud Computing Trends: 2017 State of the Cloud Survey". Available at:https://www.rightscale.com/blog/cloud-industry-insights/cloud-computing-trends-2017-state-cloud-survey. (Accessed: 06-12-2017).

[28] Qin Jia, Zhiming Shen, Weijia Song, Robbert van Renesse and Hakim Weatherspoon. 2015. "Supercloud: Opportunities and Challenges". In Proceedings of the ACM SIGOPS Operating Systems Review - Special Issue on Repeatability and Sharing of Experimental Artifacts, 137-141. DOI:10.1145/2723872.2723892.

[29] Kaveh Razavi, Ana Ion, Genc Tato, Kyuho Jeong, Renato Figueiredo, Guillaume Pierre and Thilo Kielmann. 2015. "Kangaroo: A Tenant-Centric Software-Defined Cloud Infrastructure". In Proceedings of the 2015 IEEE International Conference on Cloud Engineering (IC2E). DOI:10.1109/IC2E.2015.19.

[30] Ibrahim Mansour, Reza Sahandi, Kendra Cooper and Adrian Warman. 2016. Interoperability in the Heterogeneous Cloud Environment: A Survey of Recent User-centric Approaches. In proceedings of the ACM International Conference on Internet of things and Cloud Computing (ICC2016). DOI:http://dx.doi.org/10.1145/2896387.2896447.

[31] Faouzi Ben Charrada and Samir Tata. 2016. "An Efficient Algorithm for the Bursting of Service-Based Applications in Hybrid Clouds". In IEEE Transactions on Services Computing. DOI:10.1109/TSC.2015.2396076.

[32] Adel Nadjaran Toosi, Rodrigo N. Calheiros, and Rajkumar Buyya. 2014. "Interconnected Cloud Computing Environments: Challenges, Taxonomy, and Survey". The ACM Journal on Computing Surveys (CSUR). DOI:10.1145/2593512.

[33] Anita Choudhary, Mahesh Chandra Govil, Girdhari Singh, Lalit K. Awasthi, Emmanuel S. Pilliand Divya Kapil. 2017. "A critical survey of live virtual machine migration techniques". In Journal of Cloud Computing: Advances, Systems and Applications. DOI: 10.1186/s13677-017-0092-1.

[34] Mark Shtern, Bradley Simmons, Michael Smit and Marin Litoiu. 2012. "An architecture for Overlaying Private Clouds on Public Providers". In Proceedings of the 8th International Conference and Workshop on Network and Service Management (CNSM) and System Virtualization Management (SVM). E-ISBN: 978-1-4673-3134-0

[35] Zhenhao Pan, Qing He, Wei Jiang, Yu Chen and Yaozu Dong. 2011. "NetCloud: Towards Practical Nested Virtualization". In Proceedings of the IEEE international Conference on Cloud and Service Computing (CSC). DOI:10.1109/CSC.2011.6138541.

[36] TightVNC Software. 2017. Available at: http://www.tightvnc.com/. (Accessed: 03-12-2017).