This is a pre-print of a paper accepted for publication in the Journal of Grid Computing.
The final publication is available at Springer via http://dx.doi.org/10.1007/s10723-017-9395-1

Implementation and Provisioning of Federated Networks in Hybrid Clouds

Rafael Moreno-Vozmediano · Rubén S. Montero · Eduardo Huedo · Ignacio M. Llorente

Received: date / Accepted: date

Abstract Federated cloud networking is needed to allow the seamless and efficient interconnection of resources distributed among different clouds. This work introduces a new cloud network federation framework for the automatic provision of Layer 2 (L2) and layer 3 (L3) virtual networks to interconnect geographically distributed cloud infrastructures in a hybrid cloud scenario. After a revision of existing encapsulation technologies to implement L2 and L3 overlay networks, the paper analyzes the main topologies that can be used to construct federated network overlays within hybrid clouds. In order to demonstrate the proposed solution and compare the different topologies, the article shows a proof-of-concept of a real federated network deployment in a hybrid cloud, which spans a local private cloud, managed with OpenNebula, and two public clouds, two different regions of Amazon EC2. Results show that L2 and L3 overlay connectivity can be achieved with a minimal bandwidth overhead, lower than 10%.

Keywords Cloud Computing · Hybrid Cloud · Network Federation · L2/L3 Overlay Networks

R. Moreno-Vozmediano · R.S. Montero · E. Huedo · I.M. Llorente
Facultad de Informática, Universidad Complutense de Madrid. C/ Prof. García Santesmases 9. 28040, Madrid - Spain

R. Moreno-Vozmediano
E-mail: rmoreno@ucm.es

R.S. Montero
E-mail: rubensm@ucm.es

E. Huedo
E-mail: ehuedo@ucm.es

I.M. Llorente
E-mail: llorente@ucm.es

1 Introduction

Cloud federation enables cloud providers to collaborate and share their resources to create a large virtual pool of resources at multiple network locations. This paradigm improves the competitiveness of small and mediums IT companies, datacenters and cloud providers, since it offers the possibility of increasing its computing and storage capacity, on an on-demand basis at a reduced cost, and also brings other important benefits, such as vendor lock-in avoidance, ease implementation of high-availability setups, enable service mobility and geographical proximity, and in general, an overall improvement on the Quality of Experience (QoE) for end-users.

Many different federation scenarios for clouds and datacenters have been proposed and implemented [1–3], such as cloud brokering, cloud peering, or hybrid cloud architectures, among others, which exhibit different level of coupling and interoperation among the cloud resources. One of the most explored federation scenarios is the hybrid cloud model [4–6], also called cloud bursting, which combines the existing on-premise cloud infrastructure (e.g., a private cloud managed by a cloud manager, such as OpenNebula and OpenStack) with external resources from one or more remote clouds, which can be either public clouds (e.g. Amazon EC2, FlexiScale or Digital Ocean), or partner clouds (managed by the same or a different cloud manager). This model allows to transform the local cloud or datacenter in a highly scalable application hosting environment.

The hybrid cloud model is an interesting solution to satisfy computational demands of many IT companies, SMEs, and educational or research institutions. These kind of organizations usually have their own on-premise private cloud infrastructures to support the internal computing necessities and workloads. These infrastructures are often oversized to satisfy peak demand periods, and avoid performance slow-down. Adopting a hybrid cloud solution enables the reduction of the on-premise infrastructure size, so that it can be dimensioned for an average load, and it can be complemented with external resources from a public cloud provider or other partner institutions to satisfy peak demands. This hybrid cloud scenario is supported by some open cloud platforms, such as OpenNebula, which offers a single management point for both local private and remote public cloud resources, enabling users to decide whether their resources are deployed in local infrastructure, or in one of the supported public cloud providers (Amazon EC2, IBM SoftLayer or Microsoft Azure).

However, although hybrid cloud is a very promising technology, which is being adopted by an increasing number of companies, current hybrid cloud platforms exhibit a major limitation: the lack of federated network provisioning services to allow the seamless and efficient interconnection of resources distributed among different clouds. Currently, in a typical hybrid cloud scenario, resources from different cloud are seen as separated resources, located at independent remote networks with their own addressing schemes and attributes. In this context the end-user is responsible for implementing their own interconnection configurations (e.g. tunnels, overlays, secure channels, etc.) to enable connectivity between geographically dispersed cloud resources.

To alleviate this problem, many public cloud providers offer the possibility of creating site-to-site Virtual Private Network (VPN) tunnels (e.g. Amazon VPN Services, Google Compute Engine VPN, or VPN Azure Service, among others), which provide secure Layer 3 (L3) connectivity between the private local cloud net-

work and the remote network deployed in the public cloud. However, these VPN services offers several drawbacks. First, each cloud provider expose their own interfaces, configurations methods, and software and/or hardware requirements to instantiate and configure the VPN tunnels between the local infrastructure and the remote cloud provider, so the manual configuration of these VPN connections for different providers would require very experienced users with advanced administrative skills. Second, these VPN-based solutions do not scale properly when the number of clouds involved increases in the hybrid configuration, due to the configuration complexity, since the user has to manually configure many different VPN tunnels between the different cloud instances. Finally, VPN services offered by public clouds only provide L3 connectivity, so each network has its own addressing scheme, and the VPN ends act as gateways that encapsulate and route the L3 traffic between the remote networks.

Currently, the implementation and deployment of Layer 2 (L2) virtual overlay networks for hybrid scenarios is not supported by any cloud provider or cloud management platform. L2 overlay networks present many advantages, first they are independent of the network protocol, so they allow to encapsulate, not only IPv4 traffic, but also IPv6 traffic, and non-IP traffic; second they support broadcast and multicast traffic, so they are suitable for deploying applications based on broadcast or multicast; third they natively support mobility and migration, since hosts in different sites share a common overlay addressing scheme, so these hosts can be moved from one cloud to another with minimal reconfiguration requirements.

In this paper, we propose a cloud network federation framework, integrated with a cloud management platform (OpenNebula), which allows the automatic provision of cross-site virtual networks to interconnect geographically distributed cloud infrastructures, in order to enable the secure, reliable, and scalable deployment of hybrid cloud infrastructures. This provision model supports different types of virtual networks (i.e. L2 and L3 virtual overlay networks), according to the application or user requirements. The proposed framework leverages on existing network virtualization technologies, such as overlay networks, software defined networks (SDN), and virtualization of network functions (VNF). To show the viability of the proposed solution, we have implemented a proof-of-concept of a real federated network deployment in a hybrid cloud, which spans a local private cloud (managed with OpenNebula) and two public clouds (two different regions of Amazon EC2), we have analyzed the overhead introduced by the different tunneling mechanisms for creating L2 and L3 overlay federated networks, and we have compared the performance of the different overlay topologies.

The rest of this paper is organized as follows. Section 2 describes the related work in research and existing commercial cloud services and products for the datacenter. Section 3 introduces a generic framework for network federation to provision virtual networks across clouds. Section 4 revises the main overlay technologies to implement L2 and L3 federated networks among different clouds in a hybrid scenario, as well as the different topologies that can be adopted to implement these networks. Finally, Section 5 shows a real deployment of a L2 federated network in a hybrid cloud, spanning a local private cloud and two public clouds, and compares the performance of the network overlay topologies.

2 Related Work

Most commercial cloud providers provide some kind of interconnection, usually using some VPN technology. For example, an AWS Virtual Private Cloud (VPC), which is a virtual network dedicated to a user and isolated from other virtual networks, can connect to private datacenter by using an IPsec hardware VPN connection, making the AWS cloud an extension of the datacenter. On the other hand, Google Cloud Platform offers dedicated interconnection, VPNs, and direct peering. Google Cloud Interconnect enables to connect the datacenter to Google through dedicated connections to Google's network edge. The interconnection, offered by carrier service providers, provides quality of service and low latencies. On top of this dedicated interconnection, Google lets connect existing private networks to Compute Engine network via an IPsec connection. Alternatively, Google also offers peering to interconnect private and cloud virtual networks. There are also several technologies for datacenter federation, like EoMPLS (Ethernet over MPLS, MultiProtocol Label Switching) or VPLS (Virtual Private LAN Service), usually requiring dedicated hardware.

Cloud brokers provide network functionality using an homogeneous interface an API that hides providers differences. Some of them provide VPN functionality to connect Virtual Machines (VMs) in different providers at L3. For example, Rightscales Network Manager provides abstraction of virtual network resources (like network gateways, networks, subnets and security groups, DHCP options or route tables) offered by cloud providers with a common user interface and API to access these resources. However, it does not currently provide federation capabilities. Also, Cloud Networking Gateways (CNG) [7], used in CompatibleOne Network Services (CONETS), interconnect VMs running on multiple heterogeneous cloud providers. CNGs are packaged as a VM which run inside each of the cloud providers. A CNG can be used to connect intra-cloud VMs and save public IP addresses through NATing. Indeed, it can connect inter-cloud VMs through VPN tunnel created between CNGs. Similarly, Enstratius offers virtual appliances providing VPNs.

Cloud network federation has been also a recurrent research topic. The EU project RESERVOIR [8] created a virtual network infrastructure called Virtual Application Network (VAN), which provides an Ethernet service to VMs by constructing an overlay between hosts and using a standard IP network as the underlying physical infrastructure. All virtual machines belonging to the same VAN instance therefore belong to the same virtual layer-2, while virtual machines belonging to different VAN instances are isolated from each other. To address scalability and security, a hierarchical network service is suggested and, in the higher levels, inter-site forwarding enables connectivity between a site VAN fragment and a fragment of the VAN located at a remote site using VAN proxies. This approach has been continued [9] and adopted by the EU project BEACON [3]. Also, the EU project NOVI [10] created a federated testbed, combining PlanetLab and FEDERICA, based on the NSwitch, a distributed virtual switch based on Open vSwitch that enables a virtual entity in one domain to be connected at protocol layer 2 with another virtual entity in a remote domain. In PlanetLab, NSwitch manages multiple EoGRE (Ethernet over GRE, which is similar to NVGRE, explained below) tunnels within a host. In FEDERICA, NSwitch functionality provides the mapping of EoGRE key values of packets originating from PlanetLab to VLAN

IDs. A similar mechanism for network federation is based on dynamic translation of VLAN tags using a SDN controller [11].

The EU project BonFIRE [12] created a testbed where all the created VMs are assigned to a dedicated private network offered by a local provider in a given address range. In order to connect the several VM instances at different locations, a BonFIRE-wide VPN was established, which allows communication between separated VMs and internal BonFIRE services. For the VPN network installation, the tinc software was used, which allows the creation of a decentralized network, so there is no single point of failure if one instance loses its connection. Similarly, the EU project OPTIMIS [1] establishes an Inter-Cloud VPN (ICVPN) for each multi-cloud application, connecting all its VMs. ICVPN uses a universal P2P overlay to provide a scalable and secure service infrastructure to initiate and bind multiple VPN overlays to different cloud services.The peers of the universal overlay act as super peers for the nodes of the underlying overlays and let new nodes enroll, authenticate, bootstrap and join a particular ICVPN overlay based on the cloud service requiring a VPN service.

3 Cloud Network Federation Framework

None of the previous approaches provide a secure, simple and elastic solution to create and manage federated networks across heterogeneous cloud providers and products. Therefore, in this section, we propose a generic framework for network federation to provision virtual networks across clouds. Considering the nature of federated cloud networks, the framework should exhibit the following features:

- **Abstraction**: The user should not be aware of how the federated network is configured in terms of network elements (e.g. switches or routers) and technologies (e.g. tunnels or encapsulation techniques).
- **Heterogeneity**: Clouds in a federation can belong to the same provider (e.g. different availability zones), or to different providers. The framework should be built upon basic and common functionality, both in terms of cloud APIs and the networking functionality provided by each cloud.
- **Elasticity and mobility**: According to the service needs (e.g. for proximity aware services), the user should be able to extend the federated network to a new cloud location, to remove a location from the network, or to move network resources (such as a floating IP) from one location to another one.
- **Security**: Federated overlay networks should provide the same security guarantees from external threats than a single-cloud network. Moreover it must be assumed that interconnection will happen across public networks so just network edge security is not valid.
- **Non-trivial Quality of Service (QoS)**: The federated network should be able to provide network functions, apart from the simply interconnect, for example: DHCP, load balancers or DNS. Moreover it should support a programmatic operation of the network to implement advanced functions like service chaining or traffic prioritization and balancing.
- **Dependability**: The federated network should be able to react to component failures and readjust traffic flows to ensure the maximum possible connectivity across clouds.

– **Simplicity**: Deploying a federated network should be a simple process easy to deploy and operate. The number of components and their interaction should be reduced to a minimum and they should be based on standard protocols and applications.

The rest of this section is organized as follows: First, in Section 3.1, we will define the main components of the framework; then in Section 3.2 we describe the proposed architecture and the interaction between each component; and finally in Section 3.3. we analyze advanced usage scenarios of the proposed solution. The framework presented in this section has been developed in the context of the EU project BEACON [3].

3.1 Terms and Definitions

A *federated cloud network*, FCN hereafter, is a set of networks under control of a single entity. The networks are interconnected by a network overlay built on top of the Internet. The administrative entity is responsible for the address space assigned to the FCN and under control of all the FCN elements. This entity is usually incarnated as a single tenant, i.e. a group of cloud users who share access to cloud resources. Figure 1 depicts the main components and structure of a FCN.

A FCN consists of multiple networks termed *network segments*. A network segment is an isolated LAN provided by a cloud, with at least one element with an IP address publicly routable on the Internet. The network segments may be real LANs or implemented through any VLAN protocol, e.g. VXLAN or 802.1Q. Moreover the implementation of each network segment is independent and does not impose any requirement on any other network segment in the FCN.

Every network segment is interconnected through an overlay on top of the Internet. This overlay is built by special network elements in each network segment termed *federation agents*. Federation agents may also provide additional network functions apart from the pure interconnection of the segments. Additionally, the overlay may be implemented at the link layer (L2) or network (L3) levels. Usually these federation agents are implemented as virtual appliances.

All the network elements of a FCN, including the hosts attached to it, are provided by one or more cloud management platforms (CMP). The CMPs are assumed to be also federated following any architecture, e.g. hybrid or brokering, but no special requirements are imposed on the API, or nature of the CMPs. In this paper we will concentrate on a hybrid federation but the principles and conclusions can be applied to any other federation models without losing any generality.

Finally, federated cloud networks, network segments and federated agents needs to be coordinated and controlled by a single entity that would allow a simple deployment procedure of a FCN. The federated cloud SDN, provides this central point and exposes the functionality needed to define, consume and setup FCNs. As any other SDN it controls the underlying network elements to implement the FCN. In this case the the federated cloud SDN interacts with the SDNs of each cloud, and with the federated agents.

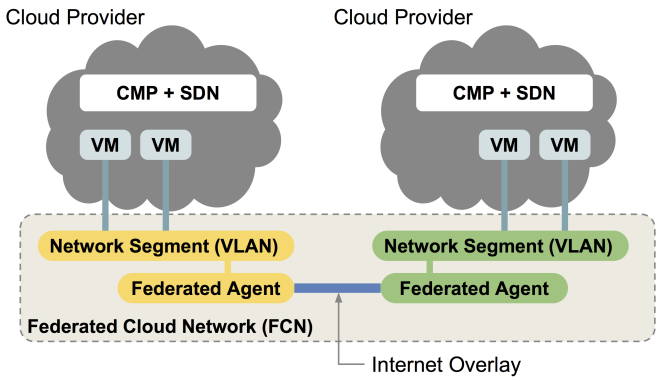

Fig. 1 Main components of the Federated Cloud Framework.

3.2 Architecture

Figure 2 shows the proposed architecture for the federation cloud network framework. The federated cloud SDN exposes a northbound API for the management of the FCNs, typically these interface is a REST API with operations including, but not limited to: create and delete FCNs, add and remove segments from a FCN (and so to allow the dynamic extension of the FCN); and update network attributes to set specific features on the FCN. The southbound API will interact with the SDN of each cloud, usually exposed through the CMP API, to create or import network segments. The southbound API will also create the federated agents by instantiating a suitable virtual appliance.

Once created, the federated cloud SDN will instruct the agents to build the corresponding overlay. As discussed in detail in Section 4, this overlay may be performed at L2 or L3 depending on the target FCN. Also note that each overlay may require different implementations of the federated agents. Moreover, additional network functions may be virtualized and packed into the agents, and configured through the federated cloud SDN, for example: DNS, DHCP servers or firewalls, among others.

The federated agents need also to expose a management interface (northbound) used by the federated cloud SDN to set up links with other federated agents and configure virtual network functions (VNF). The agent also implements a southbound API to control the network element used by the overlay. As discussed in Section 4, this element can be a software bridge or the agent routing facilities. It is important to note that the approach proposed here is network centric, i.e. all the federation elements are defined, configured and operated network-wise. All the components are distributed, as opposed to a cloud-centric approach where federation components are shared across FCNs, i.e. a single agent per cloud and not per network segment. This design option is motivated by:

– **Heterogeneity**, public clouds only expose a limited and specific set of networking operations that may not include the functionality needed to build a FCN.

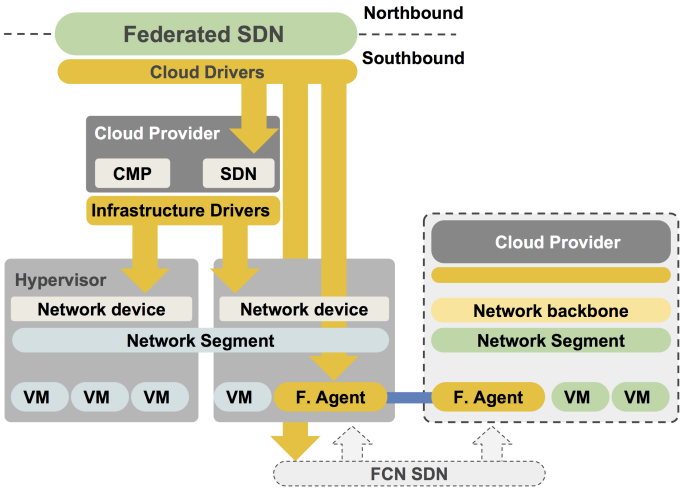

Fig. 2 Architectural overview of the cloud federation network.

- **Simple deployment and operation**, as the same components can be deployed in any cloud, managed in the same way and easily handle their life-cycle, e.g. upgrades.
- **Scalability**, as the number of network flows managed by a single component will impose important constraints on the performance and scalability of the agent implementation.
- **Reliability and security**, as the failure of an agent means the failure of a single segment and not all the segments in a cloud.
- **Multi-tenancy**, each FCN is under the complete control of a single tenant (i.e. group of cloud users) giving the framework a great flexibility and autonomy, as required by such a distributed setup.

3.3 Usage Scenarios

A FCN enables the transparent interconnection of virtual machines running on different clouds. The framework also allows the addition of advanced networking functionality when the networking elements of the federated agents also expose an OpenFlow interface, e.g. an openvswitch bridge. In this case, a Software Defined Network (SDN) controller can be used to manage the virtual switches of the agents, see also Figure 2. The runtime applications of the FCN SDN may include for example, a reactive flow engine that will install flows to interconnect hosts without the need of broadcasting ARP traffic, a flow balancer, or it may readjust flows in case of the failure of an agent or link.

Being able to allocate resources in multiple clouds connected through a FCN opens up avenues to a new type of elasticity based on where the new capacity is allocated, the localization-aware elasticity. These location-aware rules enables the the allocation of a new service components based on the cloud properties, such as price or network proximity, in terms of bandwidth or latency, to a client network.

Although not discussed in this paper, the federated cloud SDN will be typically integrated with a cloud broker that will orchestrate the deployment of the FCN and the VMs associated to a multi-VM service, implementing the elasticity rules mentioned above.

4 Implementation of Federated Virtual Networks

In this section we analyze the main overlay technologies to implement L2 and L3 FCNs among different clouds in a hybrid scenario, as well as the different topologies that can be adopted to implement these networks.

Overlay network technology enables the creation of different virtual networks, each one with a particular network topology, on top of an underlying physical network, so that the traffic between different devices connected to the overlay network is tunneled across the physical network using some encapsulation technology. Overlay networks can be L2 or L3 virtual networks. In L2 overlays, Ethernet frames (which can contain both IP and non-IP packets) are encapsulated and transported by the underlying network, enabling the emulation of LAN segments. L2 overlays are useful to emulate physical topologies and to deploy network protocols that run directly over Ethernet. On the other hand, in L3 overlays, IP datagrams are encapsulated and transported by the underlying network, which is useful to emulate IP based connectivity. Overlay networks can range from simple virtual LANs that run on a single network infrastructure to complex distributed virtual networks that span different geographically dispersed datacenters. In fact, there is an emerging interest in overlay network technology in the design of modern distributed datacenters, since this technology can improve the scalability, mobility, flexibility, resiliency, and response time of the datacenter and its integration with cloud technology.

It is important to notice that the perspective of overlay networks from the point of view datacenter distribution is different to the perspective of cloud interconnection in a hybrid scenario. In distributed datacenters these virtual networks are usually managed and configured by experienced administrators in a centralized way, since these networks can involve a large number of devices, with strict requirements regarding fault tolerance, disaster-recovery, or response time. On the other hand, in a hybrid cloud environment there can be many different users that instantiate their own virtual federated networks, usually ranging from small to medium size, so major requirements are: scalability, to support an increasing number of users; transparent and easy configuration, to enable unskilled users to create and configure their own networks; elasticity and interoperability, to allow the users networks increase their size and span different clouds on a on-demand basis.

4.1 Layer 2 Federated Networks

Implementing virtual L2 overlay networks spanning different sites is a very challenging problem, which recently has been faced up by many distributed datacenters, to support the creation of multiple Virtual LANs (VLANS) that interconnect

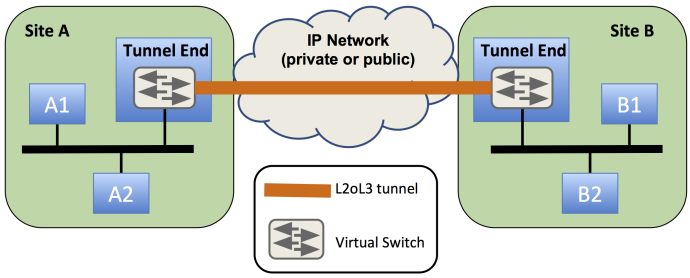

Fig. 3 Encapsulation of L2 traffic over L3 network.

geographically dispersed resources. To achieve this goal, it is necessary to encapsulate L2 (Ethernet) traffic over public or private L3 (IP) networks, using some kind of tunneling protocol (i.e. L2 over L3 or L2oL3 tunnels), as shown in Figure 3. Some of the most outstanding technologies to encapsulate Ethernet traffic over TCP, UDP or IP packets are the following:

- *VXLAN (Virtual Extensible LAN)* [13] is an Ethernet over UDP/IP encapsulation solution to support a large-scale multi-tenant L2 networking environment over a shared common physical infrastructure. VXLAN uses a 24-bit VXLAN network identifier (VNID), which supports up to 16 million VXLAN segments in the same administrative domain. VXLAN technology is not the most suitable solution to create L2 overlay over distributed datacenters or clouds, since it relies on multicast IP traffic, that is not usually supported on public IP networks.
- *STT (Stateless Transport Tunneling)* is an Ethernet over IP encapsulation method defined by an IETF draft [14]. Although STT utilizes a TCP-like header inside the IP header, it is stateless, because it does not use any TCP connection state. STT uses a 12-bit VLAN identifier, supporting up to 4096 VLAN segments. In the context of Hybrid cloud computing, STT exhibits two major limitations: first, it is still a draft not fully supported by current operating systems; and second, is particularly useful when some tunnel endpoints are in physical end-systems, as it utilizes the offloading capabilities of network interface cards to improve performance.
- *NVGRE (Network Virtualization Using Generic Routing Encapsulation)* [15] uses Generic Routing Encapsulation (GRE) to encapsulate Ethernet frames inside IP packets. Similar to VXLAN, the NVGRE header includes a 24-bit network identifier (Tenant Network ID or TNI) to support up to 16 million tenant virtual networks over the same network infrastructure.
- *GRETAP* is a simple tunneling mechanism supported by Linux systems, which also allows to encapsulate Ethernet frames on IP packets using GRE tunnels. GRETAP is implemented as a Linux Kernel capabilities and does not require any userland daemons. Unlike the previous encapsulation technologies, GRETAP is not designed for multi-tenant scenarios, as it does not support VLAN IDs. So, it is only useful for individual users that manage their own virtual networks in a independent way, using private non-shared endpoints for each virtual network.

It is important to note that none of above mentioned L2oL3 encapsulation techniques (VXLAN, STT, NVGRE or GRETAP) provides security in the communication channel established between the tunnel endpoints. If the tunnel is established over a L3 public network, it could be necessary to combine the selected L2 encapsulation method with some encryption technique, depending on the user requirements. One of the lightest mechanisms for establishing secure channels over Internet connections is IPSec, so in a general case, it would be possible to establish the L2 tunnel over an IPSec tunnel configured between both tunnel endpoints. However, this configuration is not straight for those public cloud providers that use NAT to map public IP addresses to virtual machines located at the private cloud subnets (e.g. Amazon), because IPSec is not compatible with NAT. In these situations, some alternative solutions can be adopted. A possibility is using the VPN services offered by the cloud provider, but, as we mentioned before, the user should have to deal with the different interfaces, configurations methods, and software/hardware requirements of the different cloud providers. Other solution is to establish a GRE tunnel between both tunnel endpoints, then the IPSec and the L2oL3 tunnels run over the GRE tunnel. Another possibility is to establish a site-to-site Layer 4 (L4) secure tunnel (e.g. a SSL or TLS tunnel using OpenSSH or OpenVPN) between both tunnel endpoints, and then configure the L2oL3 tunnel over this L4 secure tunnel. This solution is easier to configure, but reduces the connection throughput due to the overloading introduced by the L4 protocols.

Independently of the the encapsulation techniques or security mechanisms used, in the implementation of a L2 federated (overlay) network spanning different clouds in a hybrid scenario, it is possible to follow different alternative topologies (e.g. star, tree, or mesh). To analyze the different topologies, we assume that there is a federation agent deployed on each cloud involved (if needed), as explained in Section 3, which acts as L2oL3 tunnel endpoint, implements the L2oL3 encapsulation, and performs the required virtual switching functionality. The main L2 federated network topologies in a hybrid cloud scenario are the following:

- **Star topology** (Figure 4). This is the most simple topology, where a single federation agent is deployed in the local cloud, implementing virtual switching capabilities. Using L2oL3 tunnels, the local virtual switch is connected to remote virtual machines located in different clouds of the hybrid scenario. The main advantage of this topology is its simplicity (only one virtual switch must be configured), but it presents also some disadvantages: the local switch can become a bottleneck, since it must support all the traffic of the network; each remote virtual machine must be assigned a public IP address; communication latencies between nodes in the same remote cloud can increase significantly.
- **Tree topology** (Figure 5). In this topology there is a federation agent deployed on each cloud, implementing virtual switching capabilities. The switch running in the agent of the local cloud acts as central switch, so that all the other remote switches are connected to this central switch by means of L2oL3 tunnels. In Figure 5 we can observe that nodes in each remote cloud are also connected with L2oL3 tunnels to the switch deployed within the cloud. This is because most public cloud providers (e.g. Amazon EC2) implement MAC filtering in the local network, so any Ethernet frame with a source or destination MAC address that does not belong to the local network is discarded. In this situation, the only way to communicate a node in a remote cloud with a node in the local

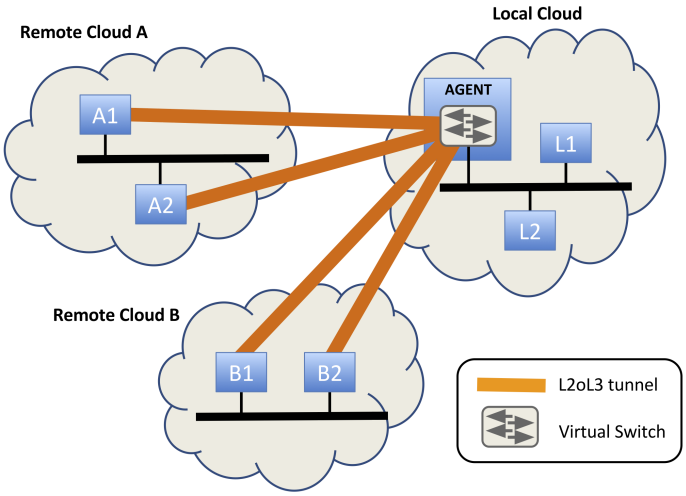

Fig. 4 L2 overlay with star topology.

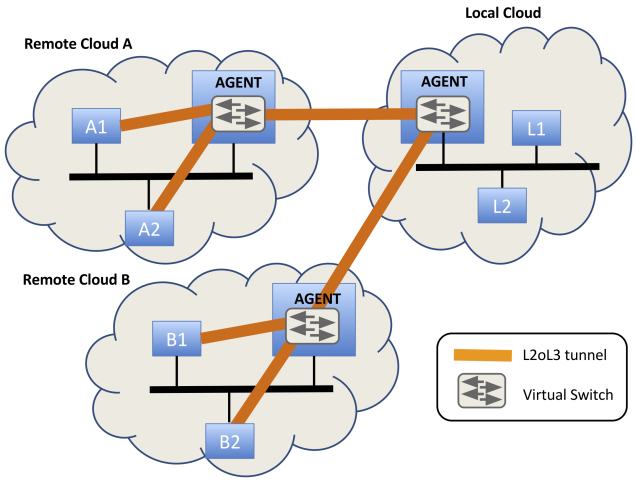

Fig. 5 L2 overlay with tree topology.

cloud, is to extend the L2 overlay network inside each remote cloud. The main advantages of this topology are that only one public IP is needed per cloud (for the federation agent), and lower communications latencies between nodes in the same remote cloud, compared to the star topology.

– **Mesh topology** (Figure 6). This topology is an extension of the tree topology, where each federation agent in each cloud is connected to all the other agents using L2oL3 tunnels, constituting a full mesh interconnection overlay network among different federation agents. Obviously, this configuration is more complex than the previous one, as it requires to configure more L2oL3 tunnels. However this topology also presents important advantages: first it improves

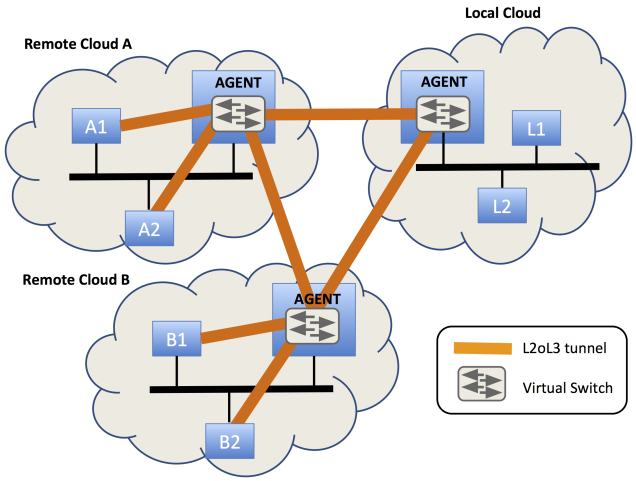

Fig. 6 L2 overlay with mesh topology.

communication latencies between nodes located in different remote clouds (e.g. between Cloud A and Cloud B in Figure 6); second, it reduces bottleneck issues, as traffic is better balanced among the different federation agents; and third, it exhibits better fault tolerance, as there are different communication paths among federation agents, so, in case of agent or link (tunnel) failure, traffic can be forwarded by other alternative paths.

Regarding the previous topologies, there are several alternatives to implement the virtual switches within the federation agents. One of the most simple solutions, in a Linux environment, is to use Linux bridges, which perform basic bridging functionality, implement the Spanning Tree Protocol (STP) to avoid loops in case of a mesh topology, and support traffic filtering at L2 level (e.g. using *ebtables*). Alternatively, more advanced virtual switches can also be used, such as Open vSwitch, which supports SDN flow-based control (OpenFlow compatible), link aggregation, VLAN support, and trunking. An important issue in L2 overlay networks is the management of broadcast traffic (e.g. ARP or DHCP traffic), which is usually costly, and should be avoided or limited. A common solution is to use an ARP proxy, that can be implemented with an *ebtables* rule, and update the switch forwarding database. Note that this process needs to be updated for every switch whenever a VM is added or removed to the network. However, using Open vSwitch and SDN technology it is possible to capture ARP or DHCP requests and reactively install flows in the switches to reach a host when it first appear in the network; this effectively reduces the need of broadcasting ARP traffic in the L2 overlay.

4.2 Layer 3 Federated Networks

L3 overlay networks have been broadly used for the interconnection of remote private LANs located at different datacenters, different sites of the same company,

or different clouds. This kind of site-to-site overlay networks are mostly based on well-known VPN technologies [16] that encapsulate private IP traffic inside IP packets that go over a public network. The most common encapsulation techniques in VPNs are the following:

- *IP-in-IP* [17] is the most simple encapsulation technique, as each IP packet is directly encapsulated inside another IP packet. The inner IP header usually contains the private IP addresses of the source and destination hosts, while the outer IP header contains the public IP addresses of the L3 tunnel endpoints. IP-in-IP does not implement any security mechanisms, so data are transmitted in plaintext over the public IP network.
- *GRE (Generic Routing Encapsulation)* [18] is a generic encapsulation mechanism that enables to tunnel any protocol inside any other protocol. It is very similar to the IP-in-IP encapsulation, but includes an additional GRE header between the inner and outer IP headers. Similar to IP-in-IP, GRE also lacks of security mechanisms.
- *IPSec* [19] is a protocol suite for secure IP communications broadly used to implement secure site-to-site VPN tunnels. IPsec provides several security features such as data origin authentication, data integrity, data encryption, and replay protection.
- *BGP/MPLS VPN* [20]. MPLS (Multiprotocol Label Switching) is a generic encapsulation mechanism for various network protocols (IP, ATM, etc.) which allows to forward packets based on path labels, instead of using network addresses and large routing tables. MPLS can be used to implement, configure and maintain complex VPNs infrastructures consisting of a full mesh of tunnels with multiple VPN endpoints. This kind of VPNs use also the Border Gateway Protocol (BGP) for the distribution of routing information among the VPN gateways.
- *L4 VPNs.* SSL (Secure Socket Layer) and TLS (Transport Layer Security) transport layer (L4) tunnels [21] can also be used to deploy secure VPNs. This kind L4 VPN tunnels can be implemented by open software such as OpenSSH or OpenVPN.

As we mentioned in Section 2, many public cloud providers offer L3 VPN services, mostly based on IPSec technology (e.g. Amazon VPN Services, Google Compute Engine VPN, or VPN Azure Service, among others), which enable to interconnect the local private network at the user premise to the remote cloud provider network. These VPN services could be used to implement L3 federated in hybrid environment. However, due to the heterogeneity of interfaces, configurations methods, and software/hardware requirements of each provider, the cloud network federation framework proposed in this paper will implement L3 overlay networks using generic VNF-based virtual gateways deployed in each federation agent of each cloud, and VPN tunnels between these gateways based on any of the encapsulation techniques previously discussed, according with the user needs. The two main L3 federated network topologies in a hybrid cloud scenario are the following:

- **Tree topology** (Figure 7). In this topology, the gateway deployed in the local cloud acts as a central router, so that it is connected to each remote cloud gateway using a VPN tunnel. This topology is the most simple and easiest

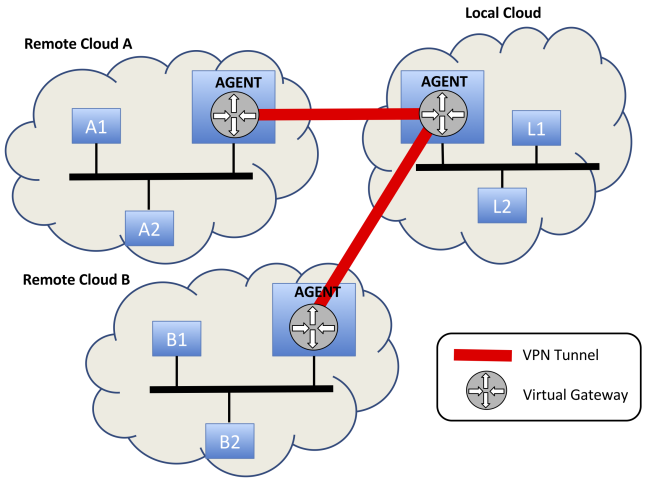

Fig. 7 L3 overlay with tree topology.

to configure, regarding the reduced number of VPN tunnels (only one tunnel per remote cloud), and the simplicity of routing tables, since all the traffic goes through the gateway of the local cloud, so route tables can be manually configured using this gateway as default router. However, this central gateway represent a central point of failure and potential bottleneck.

– **Mesh topology** (Figure 8). This topology is an extension of the tree topology, where each virtual gateway in each cloud is connected to all the other remote gateways using VPN tunnels, constituting a full mesh interconnection overlay network. Similar to the L2 overlay case, the mesh topology presents several advantages compared to the tree topology: it improves the communication latencies between nodes located in different remote clouds (e.g. between Cloud A and Cloud B in Figure 8); it reduces bottleneck issues; and it exhibits better fault tolerance, as there are different communication paths among gateways. However, this topology is more complex, as it requires to configure more VPN tunnels, and routing and route table construction is also more complex, so it requires to configure some kind of routing mechanism (e.g. BGP) among gateways to avoid loops, to detect the best routes to each destination, and to re-configure route tables in case of a change in the topology (e.g. a gateway or link failure).

5 Proof-of-concept and Performance Evaluation

In this section we show a real deployment of L2 and L3 FCNs in a hybrid cloud scenario, spanning a local private cloud (managed with OpenNebula) and two public clouds (two different regions of Amazon EC2). In the network federation framework described in Section 3, the management of FCNs is done on a per-tenant basis. That means that each tenant is responsible for managing its own FCNs, independently of the rest of tenants, and deploying their own federation agents

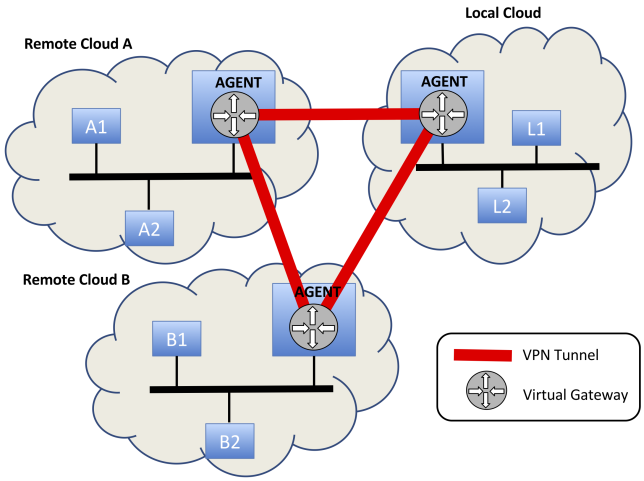

Fig. 8 L3 overlay with mesh topology.

to interconnect the different clouds, and build the L2 or L3 overlay networks. For this proof of concept, we will assume a simple scenario, where a particular tenant wants to deploy a single L2 or L3 FCN. So, in case of deploying a L2 FCN, no VLAN tags are needed to manage the overlay network. In this situation, the encapsulation method selected for configuring the L2oL3 tunnels is based on GRETAP, as it is one of the simplest methods, and is natively supported by all the Linux systems. However, this encapsulation method could be easily replaced by any of the above mentioned methods if needed. Regarding security, if the user demands secure channels, GRETAP tunnels are configured over GRE+IPSec tunnels, otherwise, plain GRETAP tunnels are used. In the case of deploying a L3 FCNs, the encapsulation method selected for configuring the L3oL3 tunnels is based on GRE, because of the same reasons than before. If the user demands secure communications, GRE+IPSec tunnels are used.

The hybrid cloud testbed used for this deployment (see Figure 9) consists of a private cloud located at the Universidad Complutense of Madrid (UCM) premises and managed by OpenNebula, and two public Amazon EC2 clouds located in two different Amazon regions: Ireland and Frankfurt. In the local UCM cloud we have deployed a private LAN (with address 192.168.0.0/24), a couple of hosts attached this LAN, and a federation agent attached to both, the private LAN and the Internet. It is important to note that this private LAN has been configured without MAC filtering, so it can accept Ethernet frames coming from or going to any MAC address. In each region of Amazon, we have created a Virtual Private Cloud (VPC) and a VPC subnet with different and independent private address ranges. In each VPC subnet of each Amazon region we have deployed a federation agent, and a couple of hosts for testing purposes. It is important to explain some particularities of the Amazon VPCs. First, Amazon performs address filtering inside each VPC subnet. This fact makes difficult the creation of L2 overlays using the original MAC addresses assigned by the VPC, and the First, Amazon performs address filtering inside each VPC subnet. This fact makes difficult the creation

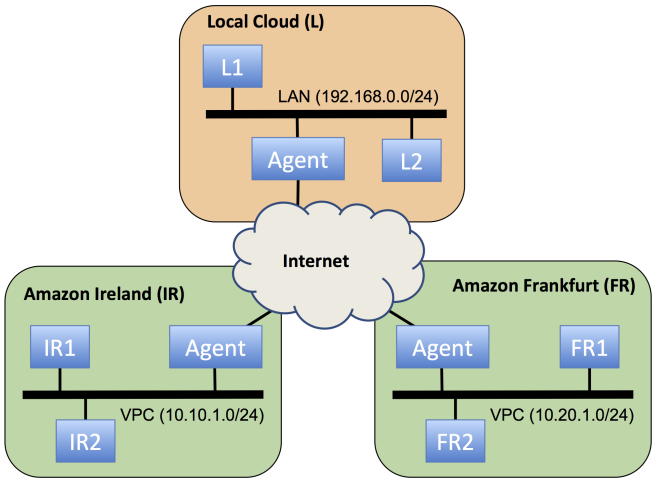

Fig. 9 Hybrid cloud testbed.

of L2 overlays using the original MAC addresses assigned by the VPC, and the configuration of a virtual machine as a virtual router to forward packets. Second, the assignment of public addresses to VMs inside a VPC is done using NAT. This fact makes difficult the creation of IPSec tunnels with remote machines.

To demonstrate the viability of the proposed framework in a hybrid scenario, we first must analyze the performance of L2 and L3 overlay networks, by comparing the network bandwidth between a Local Cloud node and an Amazon node in different overlay scenarios (L2oL3 or L3oL3 tunnels, with or without security), compared to the maximum bandwidth that it is possible to reach in a direct connection (without any kind of tunnel) between the same nodes. For this set of experiments, we have used only the Local Cloud and one of the Amazon regions (Ireland). We have compared the following scenarios, which are shown in Figure 10:

a) *Direct connection between agents* (Fig. 10.a). In this scenario we will measure the bandwidth between the Local and the Amazon agent, both using their respective public IP addresses. No tunnels are used in this experiment, so it represents the maximum achievable bandwidth between both clouds.
b) *Direct connection between nodes* (Fig. 10.b). In this scenario we will measure the bandwidth between a Local Cloud node and an Amazon node without tunnels. In this case, the Local Cloud node is connected to the Internet through the Agent, which acts as NAT router. In the Amazon side, we have configured the node with a public IP address. Note that the bandwidth of this configuration should be lower than the previous scenario, due to the overhead of the NAT router. The bandwidth obtained in this scenario will be used as baseline data to compare the performance of the different overlay configurations.
c) *L2oL3 plain tunnel* (Fig. 10.c). In this scenario we have configured a GRETAP tunnel (without security) between Local and Amazon agents. These agents implement a virtual switch (based on a Linux bridge), which interconnect both private LANs. In the Amazon side, due to the address filtering issue, the differ-

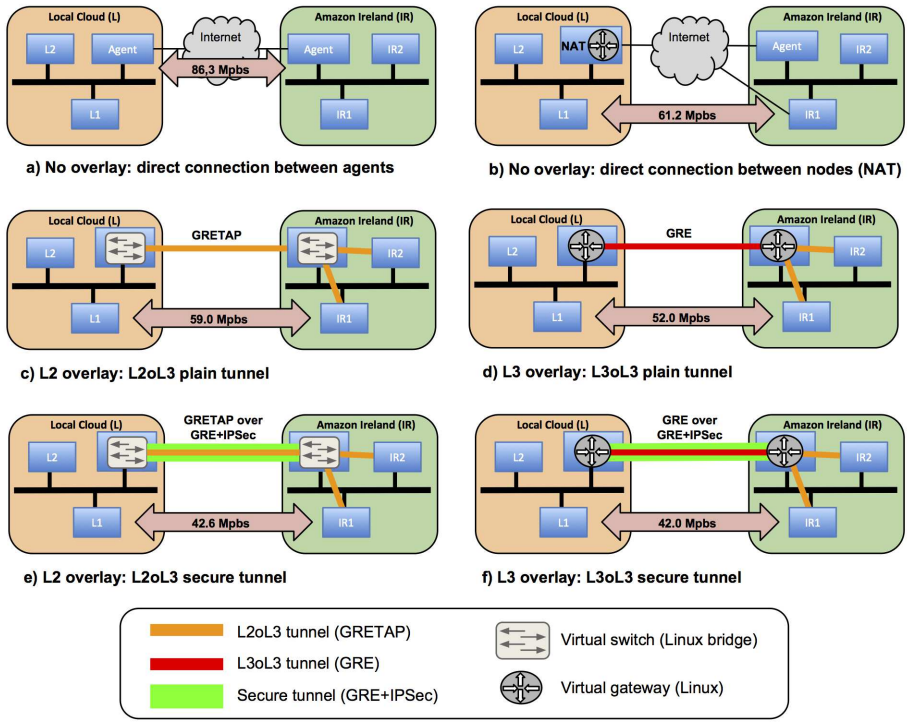

Fig. 10 Maximum bandwidths with different interconnection configurations.

ent nodes must be connected to the agent using also GRETAP tunnels, making up a private overlay LAN on top top of the VPC subnet.

d) *L3oL3 plain tunnel* (Fig. 10.d). In this scenario we have configured a GRE tunnel (without security) between Local and Amazon agents. These agents implement a virtual gateway, which interconnect both private LANs. As in the previous case, in the Amazon side, it is necessary to configure a private overlay LAN on top top of the VPC subnet (using GRETAP tunnels between the nodes and the agent), to overcome the address filtering issue.

e) *L2oL3 secure tunnel* (Fig. 10.e). This scenario is similar to scenario (c), but the GRETAP tunnel between agents is built over a secure L3 tunnel, based on IPSec. However, as public addresses in Amazon are assigned using NAT, which is not suitable for IPSec configuration, the IPSec tunnel has been configured over a GRE tunnel between both agents.

f) *L3oL3 secure tunnel* (Fig. 10.f). This scenario is similar to scenario (d), but the GRE tunnel between agents is built over a secure L3 tunnel, based on IPSec. As in the previous case, to solve the problem of NAT compatibility, the IPSec tunnel has been configured over a GRE tunnel.

Figure 10 shows the maximum bandwidth obtained with each configuration, where bandwidth has been measured using the *Iperf* tool, with the Local Cloud nodes acting as Iperf client (*iperf -c*) and the the Amazon nodes acting as Iperf server (*iperf -s*). If we compare scenarios (a) and (b), both representing direct con-

nections without tunnels, the bandwidth obtained in the scenario (b) is 30% lower than the scenario (a). This is due to the overhead introduced by the NAT router in the scenario (b). However, as the goal of this experiment is to probe the viability of the different overlay configurations, by analyzing the communication bandwidth between nodes connected to private networks of different clouds, we have to choose scenario (b) as baseline scenario. If we compare the communication bandwidth of L2 plain overlay network (scenario (c)) with the baseline case (scenario (b)), we observe that the bandwidth reduction due to the overhead introduced by the L2oL3 (GRETAP) tunnels and virtual switches is almost negligible (about 4%). In the case of a L3 plain overlay (scenario (d)) the bandwidth reduction due to the overhead introduced by the L3oL3 (GRE) tunnels and virtual gateways is a bit more significant, but lower than 10%. In the case of L2 and L3 secure overlay networks (scenarios (e) and (f)), we can observe that the bandwidth reduction, compared to the baseline case, is about 30%, which is mainly due to the overhead introduced by the encryption IPSec mechanisms. In conclusion, regarding the minimal overhead introduced by the different tunneling mechanisms, we can confirm the viability of the proposed solutions for implementing federated L2 and L3 overlay networks in a hybrid cloud scenario.

In Section 3 we introduced different topologies for implementing L2 and L3 overlay networks, so, it could be interesting to compare the performance of these topologies. For this analysis, we have chosen to analyze only the L2 case, because this kind of deployments is more complex and challenging than L3 overlay networks, and, because in the context of hybrid clouds, L2 network deployments have not been deeply explored and analyzed.

Using the testbed shown in Figure 9, we have deployed the three L2 overlay topologies presented in Section 4: star, tree, and mesh, as shown in Figure 11. We have measured the bandwidth between different nodes at different locations using Iperf tool, and establishing client-server Iperf connections in both directions. Results are shown in Figures 12 to 15. Notice that measures for the mesh topology are only included in Figure 15, which shows the communication bandwidth between different Amazon regions. For all the other combinations (Figures 12, 13 and 14), mesh topology does not introduce any improvement compared to tree topology, so it is not shown. As we can observe in Figures 12 and 13, for communications between Local Cloud nodes and Amazon nodes (Ireland or Frankfurt), star and tree topologies exhibit a very similar throughput (in some cases star topology outperforms tree topology, and in other cases it is the opposite). However, regarding the communication bandwidth between Amazon nodes within the same region (Figure 14), we can observe that tree topology clearly outperforms the star topology. Finally, regarding the communication throughput between nodes in different Amazon regions (Figure 15), we observe that the best throughput is provided by the mesh topology, followed by the tree topology.

6 Conclusions

In this work we have presented a cloud network federation framework for the automatic provision of L2 and L3 virtual networks to interconnect geographically distributed clouds in a hybrid scenario. We have shown the main features that guide the design of the network federation framework, and have defined its ar-

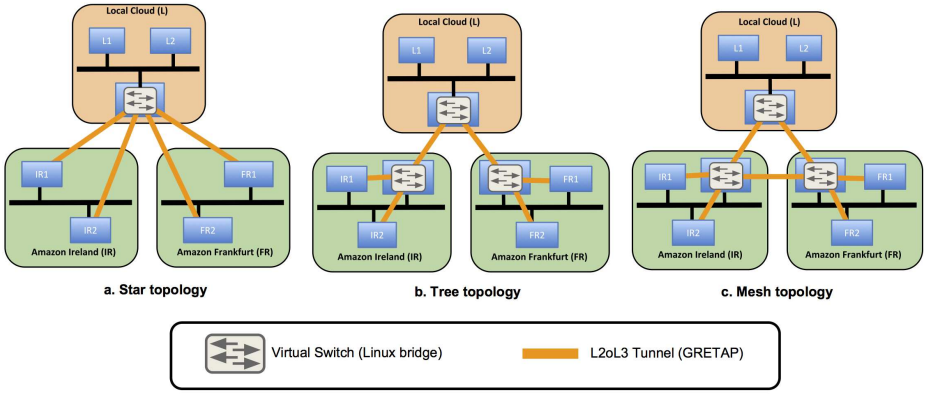

Fig. 11 L2 federated network topologies.

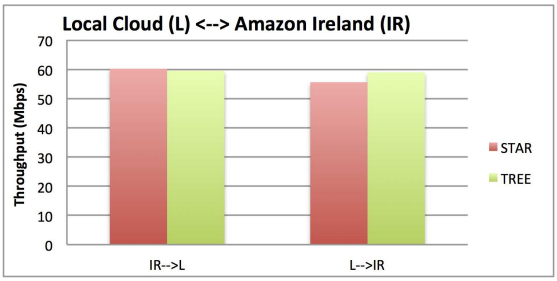

Fig. 12 Communication bandwidth between nodes in Local Cloud and Amazon Ireland.

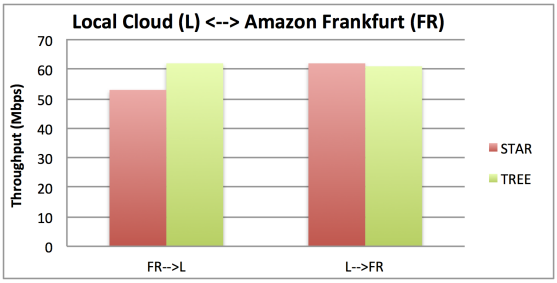

Fig. 13 Communication bandwidth between nodes in Local Cloud and Amazon Frankfurt.

chitecture and main components. We have also analyzed different implementation alternatives for the deployment of L2 and L3 overlay federated networks, including different L2oL3 and L3oL3 tunneling technologies, as well as different L2 and L3 overlay topologies. To show the viability of the proposed framework we have performed a real deployment of a hybrid cloud spanning a local private cloud (managed with OpenNebula) and two public clouds (two different regions of Amazon EC2). In this proof-of-concept, L2 overlay networks have been created using GRETAP tunnels between local and remote federation agents, and L3 overlay networks have been created using GRE tunnels. Optionally, to provide security,

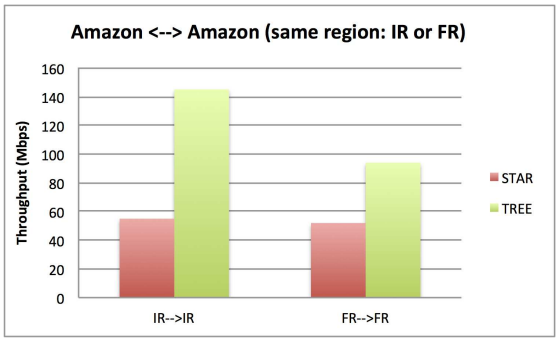

Fig. 14 Communication bandwidth between nodes within the same Amazon region (Ireland or Frankfurt).

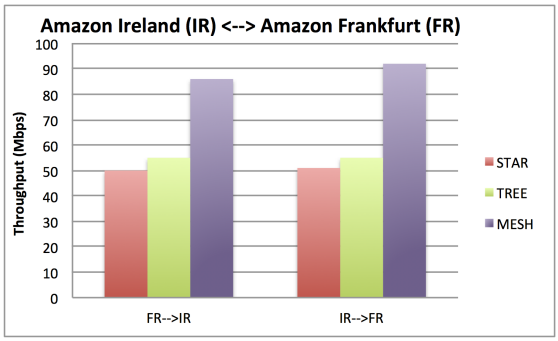

Fig. 15 Communication bandwidth between nodes in different Amazon regions (Ireland and Frankfurt).

these tunnels can be built over secure tunnels, based on GRE+IPSec. Bandwidth measurements between local and external cloud nodes show that the overhead introduced by the L2 and L3 overlay networks, compared to direct communications (without tunnels), is lower than 10% for plain tunnels (without security), and lower than 30% in the case of secure tunnels. We have also compared the performance of the different topologies for the L2 overlay case, and we have shown that, in general, tree and mesh topologies exhibit a better throughput.

As future work, we plan to improve the management of L2 and L3 overlay networks by introducing SDN-based control for virtual network devices (e.g. switches and gateways), and implementing other alternative encapsulation techniques (e.g. VXLAN, NVGRE, etc.). These two improvements will also enable the management of multiple VLANs by the same federation agents. We also plan to integrate the cloud federation framework with the OpenNebula Cloud Manager, This work has been done in the context of the BEACON project (EU H2020 framework programme), so the results of this research will be also used to provide feedback to the project development.

Acknowledgements This research was supported by the European Unions Horizon 2020 Research and Innovation Program under the Grant Agreement No 644048 (BEACON), and

by Ministerio de Economía y Competitividad of Spain through research grant TIN2012-31518 (ServiceCloud).

References

1. A. Ferrer, F. Hernandez, J. Tordsson et al.: Optimis: A holistic approach to cloud service provisioning. Future Generation Computer Systems, 28(1), 66–77 (2012)
2. R. Moreno-Vozmediano, R.S. Montero, I.M. Llorente: IaaS Cloud Architecture: From Virtualized Data Centers to Federated Cloud Infrastructures. Computer, 45(12), 65–72 (2012)
3. R. Moreno-Vozmediano, E. Huedo, I. Llorente et al.: BEACON: A Cloud Network Federation Framework. 1st Workshop on Federated Cloud Networking (FedCloudNet) (2015)
4. R.S. Montero, R. Moreno-Vozmediano, I.M. Llorente: An Elasticity Model for High Throughput Computing Clusters. Journal of Parallel and Distributed Computing, 71(6), 750–757 (2011)
5. B. Sotomayor, R.S. Montero, I.M. Llorente, I. Foster: Virtual Infrastructure Management in Private and Hybrid Clouds. Internet Computing, 13(5), 14–22 (2010)
6. E. Sturrus, O. Kulikova: Orchestrating Hybrid Cloud Deployment: An Overview. Computer, 47(6), 85–87 (2014)
7. M. Mechtri, D. Zeghlache, E. Zekri, I.J. Marshall: Inter and intra Cloud Networking Gateway as a service. IEEE 2nd Int. Conf. Cloud Networking (CloudNet), 2013, 156–163 (2013)
8. B. Rochwerger, D. Breitgand, E. Levy et al.: The RESERVOIR Model and Architecture for Open Federated Cloud Computing. IBM J. Res. Dev. 53 (4), 535–545 (2009)
9. A. Levin, K. Barabash, Y. Ben-Itzhak, S. Guenender, L. Schour: Networking Architecture for Seamless Cloud Interoperability. IEEE 8th Intl. Conf. Cloud Computing (CLOUD), 1021–1024 (2015)
10. R. Lapacz, B. Pietrzak: Networking solutions in the federation of clouds. 9th Intl. Conf. Network and Service Management (CNSM), 394–397 (2013)
11. M.Y. Luo, S.W. Lin, J.Y. Chen: Towards Network Virtualization Management for Federated Cloud Systems. IEEE 6th International Conference on Cloud Computing (2013)
12. D. Garcia-Perez, J.A. Lorenzo del Castillo, Y. Al-Hazmi et al.: Cloud and Network Facilities Federation in BonFIRE. 1st Intl. Workshop on Federative and Interoperable Cloud Infrastructures (FedICI 2013), Euro-Par 2013, 126–135 (2014)
13. M. Mahalingam el al.: RFC 7348: Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks. IETF Network Working Group (2014)
14. B. Davie: A Stateless Transport Tunneling Protocol for Network Virtualization (STT). IETF Network Working Group Internet-Draft, https://tools.ietf.org/html/draft-davie-stt-01 (2002)
15. M. Sridharan et al.: NVGRE: Network Virtualization using Generic Routing Encapsulation. IETF Network Working Group Internet-Draft, https://tools.ietf.org/html/draft-sridharan-virtualization-nvgre-00 (2011)
16. M. Rathore, A. Razzaq, M. Hidell, P. Sjodin: Site-to-Site VPN Technologies: A Survey. KTH, Telekommunikationssystem (2009)
17. C. Perkins: RFC 2003: IP Encapsulation within IP. IETF Network Working Group (1996)
18. D. Farinacci et al.: RFC 2784: Generic Routing Encapsulation (GRE). IETF Network Working Group (2000)
19. S. Kent and K. Seo: RFC 4301: Security Architecture for the Internet Protocol. IETF Network Working Group (2005)
20. E. Rosen and Y. Rekhter: RFC 4364: BGP/MPLS IP Virtual Private Networks (VPNs). IETF Network Working Group (2006)
21. C. Fei, W. Kehe, C. Wei, Z. Qianyuan: The Research and Implementation of the VPN Gateway Based on SSL. Fifth International Conference on Computational and Information Sciences (ICCIS), 1376–1379 (2013)