Cloud Atlas: A Software Defined Networking Abstraction for Cloud to WAN Virtual Networking

Stephan Baucke, Racha Ben Ali, James Kempf, Ramesh Mishra
Ericsson Research
San Jose, CA
first dot last at Ericsson dot com

Franco Ferioli, Angelo Carossino
Ericsson Telecomunicazioni S.p.A.
Genoa, Italy
first dot last at Ericsson dot com

***Abstract*—One of the primary principles of Software Defined Networking (SDN) is that representing networks as a collection of simple abstractions implemented as an API - rather than as a collection of standardized and proprietary protocols and command line interfaces - will lead to networks that are easier to build and manage. Application of this principle has advanced furthest in cloud computing, where the OpenStack Quantum network service provides tenants in a data center with an abstraction of an enterprise LAN. The Quantum API implements a virtual network through a plug-in, which might in fact require protocols and command line interfaces to drive the physical hardware. However, existing OpenStack support for wide area connectivity is restricted to L3 IPsec and SSL VPNs which do not in general support quality of service (QoS). In this paper, we present Cloud Atlas, a SDN abstraction and API extending the Quantum virtual network into the WAN. Cloud Atlas is built on top of existing WAN network services (L1-, L2-, and L3VPNs) that do support QoS. Cloud Atlas makes these services available to OpenStack through a tight integration with Quantum. We discuss two prototypes we have built of Cloud Atlas, one based on command line scripts and one based on a network management system. We conclude the paper with some observations on applying the cloud service model to networking and the value of SDN abstractions in supporting such a service model.**

Keywords-component; elastic networking; networking as a service; OpenStack;, software defined networking; SDN

I. Introduction

In most areas of computer science, abstraction has been used to simplify the interface between system components, thereby simplifying the construction of large scale, complex systems. Networking has long resisted this trend. Shenker [1] has pointed out that most research work in networking focuses on mastering the complexity of protocols and interfaces rather than on simplifying complexity to enable the construction of larger systems from components with well-defined interfaces. Rexford [2] has described the networking literature as being concerned primarily with a proliferation of protocols, a large collection of header formats, and tools to manage them, rather than with abstracting out the essential technical features to make networking easier to use and understand.

In response to these concerns, the networking research community has moved over the last few years toward *Software Defined Networking* (SDN). In its current version, SDN is about defining abstractions that expose the appropriate level of detail for complex network functions and implementing those abstractions in an API for programmability. The programmability facilitates automation of provisioning, configuration, and management.

The area of networking where SDN is furthest along is in cloud management/operating systems. For example, OpenStack [3] defines a network virtualization service for data center networks called Quantum [4] that provides isolation between tenants, connectivity between VMs for the tenant owning the virtual network, and an API for tenants to directly manage their virtual networks within the data center. The automation of cloud network management removes human intervention; speeding up the deployment and provisioning of tenant networks in the data centers in the same way that compute and storage virtualization have for computation and storage. The Quantum API presents an abstraction of an enterprise local area network that is easy for tenants to understand and use. Other cloud operating systems provide similar network virtualization APIs.

A similar service for automatically setting up end-to-end virtual networks to destinations outside the cloud would be desirable. OpenStack supports an API for IPsec or SSL-based tunnels (over-the-top VPNs) between VM gateways and clients outside the cloud [5] but over-the-top VPNs do not support quality of service (QoS). Most enterprise networks typically require QoS on their VPNs so that they are assured of not running out of bandwidth at critical times. Some public cloud providers offer provider-provisioned L3VPN services into the data center with QoS properties, but they are typically more restricted than over-the-top VPNs and are static. What would be desirable is a service offering wide area virtual networks with QoS properties integrated with Quantum and having the same dynamic elasticity and dispatchability as compute and storage resources in the cloud.

To alleviate these issues, we propose *Cloud Accessible Transport LAyer Service (Cloud Atlas)*, an SDN abstraction for cloud to WAN virtual networking. Cloud Atlas provides an abstraction of inter-site WAN connectivity, similar to the virtual switch abstraction of local-area connectivity, making wide-area connectivity accessible through cloud operating systems. An easy-to-use API implementing the abstraction allows wide area connectivity to be provisioned by the cloud tenant, thus allowing the tenant to construct an end-to-end virtual network from the data center to destinations in the WAN. The abstraction allows network providers to make their wide-area network assets tenant-managable, allowing tenants to manage them in coordination with computer/storage resources, and to enable automated management of end-to-end virtual slices of the physical connectivity. Cloud Atlas can additionally be used to manage over-the-top VPNs if desired. The abstraction is largely independent of the underlying physical implementation, which is provided by a plug-in.

In Section II, we discuss the technical contribution of Cloud Atlas with respect to related work. In Section III, we present a few use cases of how Cloud Atlas could implement connectivity services that are difficult to achieve with existing over-the-top, data center to WAN solutions. Section IV briefly reviews the OpenStack cloud operating system, with focus on the Quantum intra-data center virtual networking component, upon which Cloud Atlas is based. While we believe the general approach is applicable to other cloud operating system, we used OpenStack for our prototype because of its modular architecture and the Quantum virtual networking system. Section V presents the abstractions and algorithm in Cloud Atlas and describes how it is integrated with OpenStack and the underlying WAN. In Section VI, we briefly present a couple of prototypes we have built with Cloud Atlas and Virtual Private LAN Service (VPLS) [6], a L2VPN service built on top of Multiprotocol Label Switching (MPLS) [7]. VPLS is a widely deployed L2VPN service for connecting enterprise sites and data centers. Finally, in Section VII, we summarize the paper and present a few conclusions.

II. Related Work

Many research projects and most commercial offerings utilize IPsec or SSL VPNs running inside VM gateways to provide cloud to cloud or cloud to user connectivity. Virtuoso, VIOLIN, and QCG [8][9][10] are three academic projects that provide VPN connectivity between nodes in a grid for grid computing using over-the-top VPNs. Most commercial cloud providers such as Amazon EC2 offer IPsec or SSL VPNs for connectivity between enterprise networks and the cloud data center [11]. Over-the-top VPNs don't provide guaranteed QoS Service Level Agreements (SLAs) which most enterprises demand and which are necessary for virtualizing telecommunication services.

Some commercial cloud providers with international data networks offer provider-provisioned L2VPNs with guaranteed SLAs. Savvis offers a L2VPN service for connecting enterprises and data centers based on MPLS VPNs with 6 QoS levels [12] called Application Transport Services. While this addresses the need for QoS, it is not clear whether the services actually provide end-to-end virtual networking with the same level of elastic dispatchability as virtualized compute and storage. Such features, together with an API integrated into the cloud operating system, are necessary if the service is to be tenant provisioned. Amazon also offers a service called Direct Connect [13] integrated with their Virtual Private Cloud service which consists of BGP peering through a dedicated 1Gb or 10Gb fiber Ethernet link. L2 bridging, which is required for L2VPN, is not supported however.

A few research projects have also explored using L2VPNs with elastic dispatchability. Baldine, et al [14] describe an architecture and testbed for a distributed network of data centers with emphasis on integrating networking. The control framework is implemented within the ORCA extensible control platform [15] for leasing resources in a shared infrastructure, which was developed as part of the NSF GENI project [16]. They describe an architecture where the user interacts with brokers to obtain tickets for resources at sites. The user then redeems the tickets for utilizing the resources at the sites. In order to facilitate having the framework understand and reason about heterogeneous resources owned by different owners, the authors developed an ontology using the OWL language. The language is applied to the problem of stitching together VLANs end-to-end to provide a slice of the network. The authors developed three models of stitching – hop-by-hop, centralized, and co-ordinated. The models are distinguished by what entity is responsible for the stitching.

Wood, et al in two papers [17] [18] describe how WAN connectivity resources can be more tightly integrated into the cloud, so that L2VPNs can be elastic and dispatchable like other cloud resources. In the first paper, the authors describe an abstraction of a virtual data center and a VPLS service connecting the data center to an enterprise or another data center that they call a virtual private cloud[1]. In the second, they describe an implementation of a particular use case, involving moving a VM from one data center to another. For this, they modified the Xen hypervisor to accommodate the lower bandwidth available on wide area networks. Their focus is on traditional networking topics, like improving performance over the wide area link under the load of VM movement.

In an approach similar to that of Baldine, et al, Schaffrath, et al describes a resource description language for configuring virtual networks called FleRD [19]. The language is designed to be used in a particular business ecosystem with virtual network providers offering resources from physical network providers [20]. In contrast to similar efforts, the language allows the specification of vague requirements, e.g. that a network connection not run thorough Afghanistan. The language covers virtual servers and storage as well as network connections, and is designed to be a general solution for describing cloud deployments. A generic NetworkElement type describes network objects and a NetworkInterface type allows network objects to be interconnected. NetworkElement objects are associated with two sets of attribute-value pairs: Resource and Features. Resource models any characteristic that can be measured (e.g. bandwidth) while Features indicate characteristics that are qualitative (e.g. geographical location). The virtual service instance description is then handed off to a virtual network provider who brokers the resources from a physical network provider. With FleRD, it should be possible to compose simple qualitative features and build measurable resources with, for example, a lower bound on latency between two geographical locations, or other composed qualitative features (LAN, MAN or WAN).

In contrast to the work of Baldine, et al, Cloud Atlas leaves the details of constructing the end-to-end path to existing

[1] Wood, et al's definition is distinct from the Amazon EC2 Virtual Private Cloud product in that, originally, EC2 did not support provider provisioned VPNs. That support was added after the paper was published.

underlying WAN mechanisms, and concentrates on the case where the end-to-end path through the WAN is controlled by a single network operator. Stitching is only required between the WAN VPN and the Quantum VPN, and that is handled by the Cloud Atlas elastic networking subsystem. In comparison with the work of Wood, et al, the underlying L2VPN technology used in Cloud Atlas is the same (VPLS), but Cloud Atlas is more focused on integrating WAN networking into the cloud operating system and providing a simple API to allow easy tenant self-provisioning. The Cloud Atlas API could in principle be used with any type of underlying VPN technology even L1VPNs, since it hides the complexity of WAN VPN provisioning. Cloud Atlas also focusses specifically on the WAN network API unlike the work of Schaffrath et al. The abstractions Cloud Atlas presents to the cloud OS tenant are targeted at WAN networking rather than being generic, because there are differences in offered latency and bandwidth and in topology between WAN and internal cloud LAN services that tenants may need to be aware of. Finally, unlike the work of both Baldine et al and Schaffrath et al, Cloud Atlas is an API rather than a language. An API seems a better match for cloud operating systems like OpenStack, and the goal in Cloud Atlas was not to provide a way of describing virtualized systems across providers, as it was in the two earlier works, but rather to provide tend-to-end, cloud to WAN virtual networking with QoS that tenants can self-provision across a single provider.

III. Example Use Cases

Existing over-the-top VPNs provide virtual, isolated, elastic connectivity to applications running in the cloud. What Cloud Atlas provides in addition is the ability of the network operator to offer SLAs for network bandwidth and possibly also latency. In addition, many WAN VPN technologies also feature the ability to extend a corporate LAN into the cloud, rather than just provide a simple point-to-point connection. Applications that require these properties are most likely to benefit from Cloud Atlas.

Hybrid cloud, in which a private data center offers service in a public cloud when resources in the private data center are short, is emerging as the most commonly used cloud service model for enterprise cloud applications. What Cloud Atlas brings to hybrid cloud services is the ability to rapidly and elastically move an entire corporate LAN into the cloud on demand, rather than just a single application. In addition, because Cloud Atlas supports SLAs, telecommunication applications like realtime voice or video over IP can be provided with guaranteed QoS.

Another emerging application is *bandwidth calendaring*. Bandwidth calendaring involves scheduling bandwidth for particular times when backups or bulk data transfer are required between an application in the cloud and either another cloud or the enterprise. The cloud tenant only pays for the connectivity when it is provided, just as with compute and storage. Typically the network operator can provide larger bursts of bandwidth when the link doesn't need to be shared by multiple customers, so the tenant's job completes more rapidly. This can lead to higher reliability, since if the link goes down, only one tenant's job will be affected.

Finally, *disaster recovery* is another application that can benefit from Cloud Atlas. Disaster recovery requires periodic checkpoints of running VMs into another data center. These checkpoints can be made by bringing up the virtual WAN link, checkpointing the VM, then bringing it down. Disaster recovery is distinguished from bandwidth calendaring in that the bandwidth for the VM copy is not scheduled, it is brought up on-demand at a checkpointing trigger, which is a minimal atomic transactional change to a VM memory/virtual disk that can be applied without losing its consistent state. If the data center in which the original VM was running crashes, the archived copy can be quickly brought up, restoring the service.

IV. The OpenStack Cloud Operating System

OpenStack is an open source IaaS cloud operating system, developed under the Apache 2.0 license, managed by the OpenStack Foundation [3]. OpenStack consists of services that manage compute, storage, and networking resources. Additional functions such as an internal message bus and GUI components are also supported. OpenStack operates in the layer above the virtualization layer. It strives to be hardware and hypervisor agnostic and currently supports several major hypervisors as well as a variety of hardware configurations.

OpenStack employs a modular structure consisting of services for specific tasks. OpenStack Compute handles activation of VMs and OpenStack Swift handles distributed object storage. In our work, there are two specific services we utilized:

- OpenStack Image Service ("Glance"), which manages and delivers virtual disk images in a variety of formats. We utilized Glance in our application prototype described in Section VI.
- OpenStack Networking ("Quantum") [4] which provides a multi-tenant, intra-data center Layer 2 service modeled on an enterprise LAN. We utilized Quantum in the basic Cloud Atlas implementation.

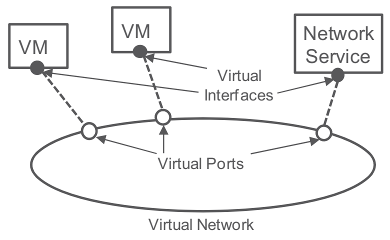

Figure 1. Quantum Abstractions

The Quantum service offers exactly the kind of flexible, elastic support for "Networking as a Service" within the data center that we seek to achieve with Cloud Atlas. Quantum provides simple abstractions and an extensible API that allows cloud tenants to configure and manage virtual network topologies and to plug advanced network services such as firewalls into tenant networks. Quantum aims to offer a more flexible and extensible framework for connectivity services.

As shown in Figure 1. , the base abstractions used by Quantum are Virtual Networks and Virtual Ports. Virtual

Networks provide an isolated virtual Layer 2 service upon which tenants can build. Virtual Ports are connection objects where Virtual Interfaces, for example the virtual NICs of virtual machines, can be attached to a Virtual Network. Together, these objects provide a generalized abstraction conceptually similar to the virtual distributed switches that are utilized in many cloud deployments today. However, Quantum is not restricted to any particular underlying network technology (such as VLANs) to implement the virtual tenant networks on the physical hardware. Rather, it employs a plug-in architecture that allows the implementation of the Quantum services using a variety of different open or proprietary network technologies. This way, Quantum opens up the cloud operating system for new and advanced SDN technologies.

V. CLOUD ATLAS

A. Overview

Cloud Atlas is designed to work with a variety of underlying WAN VPN technologies via a plug-in architecture, and to be interoperable with the OpenStack Quantum service, so tenants can use Cloud Atlas to interconnect Quantum networks with remote sites. To this end, Cloud Atlas performs three main tasks:

- Expose an abstract API that allows the cloud tenant or a higher-level orchestration layer to request and manage WAN connectivity to remote sites, and to interconnect remote sites with the local tenant networks in a dynamic and elastic fashion, thereby achieving end-to-end connectivity for cloud tenants.
- Interact with the local cloud OS used at the sites to interconnect (stitch) the WAN connections with the local tenant networks.
- Interact with an underlying network infrastructure to provision virtual inter-site connections across the WAN. Existing WAN virtualization techniques, for example L2VPN, are used to support multi-tenancy over the WAN.

By providing an abstraction layer and generic cloud OS API for WAN connectivity (rather than requiring cloud tenants to deal with the details of specific WAN technologies themselves), the system simplifies the framework that cloud and network operators use to integrate existing WAN services transparently into the cloud. The API can either be used directly by a cloud tenant, or programmatically by a higher-level distributed cloud orchestration system.

B. Cloud Atlas Components

The Cloud Atlas system consists of three types of components, shown in Figure 2. :

- The *Elastic Networking (EN) Manager* coordinates cross-site connections and provides the Cloud Atlas API for use by tenants or orchestration systems. In addition, it can host plug-ins that configure networks or network elements that do not belong to either of the participating sites, for example by accessing a wide-area network management system (NMS).
- Each participating site runs an *EN Agent*. The Agents carry out functions that require local access at the site. They interact with the local OpenStack instance to stitch the WAN connection and the local Quantum networks together. They allow the EN Manager to discover the gateways that are available at the site and can be used for WAN connections. They also can host plug-ins to configure internal network elements if required for a specific wide-area network. This can include, for example, the configuration of gateways or customer edge routers. In addition to these functions, the agents enable the site's operator to enforce policies for accessing the local resources.
- Network-specific plug-ins, indicated by *EN Plug-in* in the figure, can be hosted by both the EN Manager and the EN Agents. EN Plug-ins implement functions that are specific to the underlying WAN network deployment and hardware, for example, a CLI script manager for configuring a VPLS service on a particular kind of router.

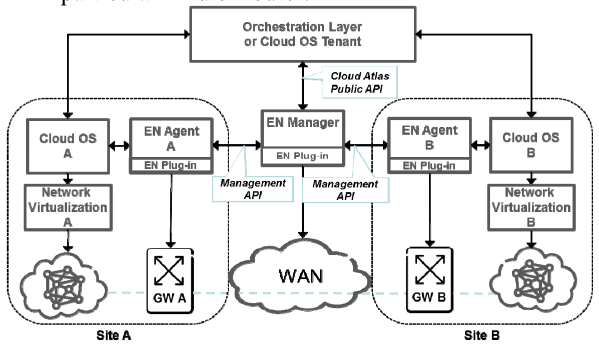

Figure 2. Cloud Atlas Components

For each site that wants to participate in cross-site connections managed by Cloud Atlas, an EN Agent needs to be registered with the EN Manager. The EN Manager then interacts with the EN Agents through a well-defined management API not available to tenants. While the EN Agents need to be hosted at their sites, the EN Manager can run anywhere, as long as control plane connectivity with the EN Agents is provided. Typically, it will either run at one of the sites, or together with a higher-level orchestration system at a separate location.

C. Abstractions and API

Cloud Atlas uses a set of abstractions, similar to the abstractions used by Quantum, which each site exposes to enable the tenant or an orchestration system to create end-to-end connectivity between virtual networks at multiple sites. The main object types are illustrated in Figure 3.

A Site is a collection of cloud resources managed by an instance of a cloud operating system. Each site is associated with an EN Agent. Sites host Gateways that provide connectivity between local and wide-area networks. Each

Gateway has a type representing a particular wide-area network virtualization technology (e.g. IPSec VPN, VPLS, etc.), and is associated with a wide-area network domain. A Virtual Link is a virtual point-to-point connection between two Gateways of the same type at different Sites. A Virtual Port is a logical endpoint of a Virtual Link and represents the local gateway configuration state associated with that Virtual Link. Finally, a VPN is a virtual multipoint-to-multipoint network that connects two or more sites using Virtual Links of the same type. The VPN object itself has a type as well, which corresponds to the common Virtual Link type. Only Sites with a Gateway supporting this type can be added to the VPN.

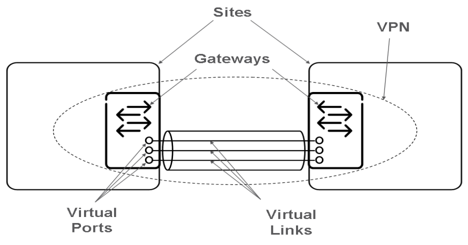

Figure 3. Cloud Atlas Abstractions for Elastic Networking

1. Public API

The EN Manager exposes a number of operations to tenants or an orchestration layer, based on the abstractions described above. These include the creation of VPN objects, addition of Sites to VPNs, Gateway discovery, and the creation and management of Virtual Links. A Virtual Link re-configuration method can be used to dynamically change the bandwidth. Finally, the API supports a method for retrieving and collecting Quantum-compatible Virtual Interface Identifiers corresponding to Virtual Ports from the EN Agents. These identifiers are forwarded to the tenant who uses them to plug a Virtual Link endpoint into local Quantum networks at the participating sites. The EN Agents bridge EN Virtual Ports with Quantum virtual ports (e.g. using VLANs between EN GWs and Quantum GWs.) creating end-to-end connectivity.

2. Management API

In addition to the public API, a well-defined management API is used by the EN Manager to communicate with the EN Agents. This API provides access to local operations at the participating sites. The management API includes methods for the discovery of local Gateways on behalf of the EN Manager, the management and configuration of Virtual Ports in the local Gateways, and the reporting of local status information to the EN Manager. The Agent also offers a method for the construction of Quantum-compatible Virtual Interface Identifiers in cooperation with the local Quantum service as already mentioned.

In our prototype, the APIs are implemented as RESTful interfaces, similar in style to the existing OpenStack APIs. Since the APIs need to support a variety of network technologies with different capabilities, some of the parameters are optional. This particularly affects QoS parameters, since different network types have different QoS capabilities Over-the-top VPNs don't support QoS, for example. The methods for the creation and configuration of Virtual Links and Virtual Ports take this into account by supporting optional parameters in the form of optional JSON or XML attributes.

The management API also supports the transport of opaque objects between the EN Agents and the EN Manager. This accommodates the needs of the network technology-specific EN plug-ins. For example, in our prototype implementation using VPLS as the underlying VPN technology and CLI scripts to implement the router configuration, the MPLS labels used to implement the VPLS tunnels are negotiated between sites. As another example, the mechanism could be used to exchange keys for an IPSec VPN.

3. Algorithm

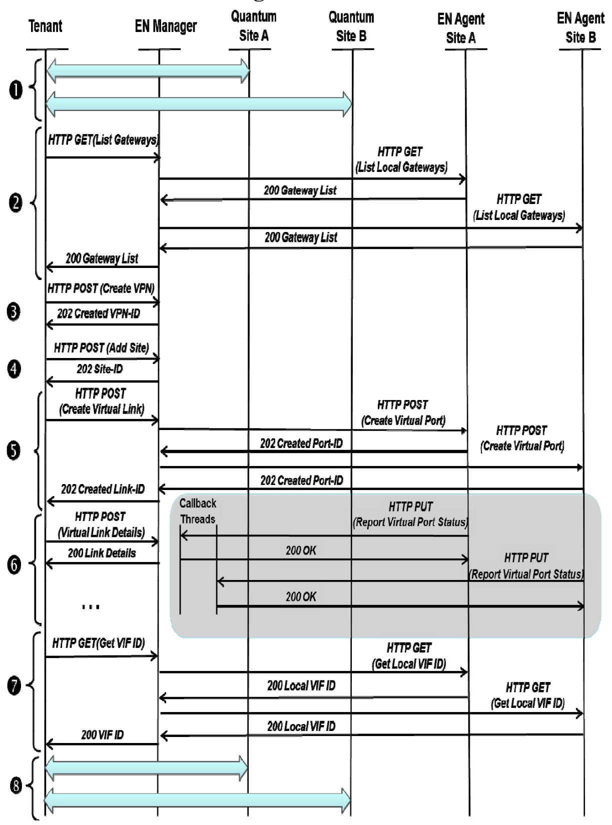

Figure 4. Message Sequence Diagram for Creating an Elastic Network Connection

A typical call sequence to create an end-to-end Layer-2 network spanning two OpenStack sites A and B using the public EN API and the local Quantum APIs is shown in Figure 4. Note that this call sequence corresponds to an EN Agent plug-in that uses CLI scripts to configure the data center gateway routers. The steps are as follows:

1. Create Quantum networks Q_A and Q_B, and ports, P_A and P_B in Q_A and Q_B, at both sites,
2. Discover Gateways of a common VPN type at sites A and B. The EN Manager passes the calls along to the

EN Agents in sites A and B, which return the local Gateways.

3. Create a new VPN of the type found in step 2,
4. Add Sites A and B to the new VPN,
5. Create a Virtual Link between A and B. The EN Manager executes a plug-in specific for configuring a new VPN link on the particular network deployment and hardware, in this case a plug-in to configure the gateway routers with CLI scripts. The EN Manager calls the Create Virtual Port method in the EN Agents for sites A and B, passing to them URLs to use as a callback for reporting on link connection status,
6. The Tenant polls the EN Manager for link status. The EN Manager starts threads to respond to periodic reports on status from the EN Agents at sites A and B. The EN Agents call their plug-ins, which send CLI scripts to the PE routers to set up the Virtual Ports. Call back threads running in EN Agents at sites A and B update the link status at the EN Manager after they finish configuring the gateways. When both EN Agents report that the connection is established, the next poll by the Tenant reports “Connected”.
7. Retrieve Virtual Interface IDs (VIF-IDs) V_A and V_B for the Virtual Link endpoints at sites A and B. The EN Manager passes the call along to the two EN Agents.
8. Call the Quantum instances at sites A and B to plug V_A into P_A and V_B in P_B.

The Virtual Link re-configuration method can be called at any time after creation of the VPN link to change the bandwidth allocation.

If the EN Manager plug-in handles the network deployment as a whole, then step 6 is considerably simplified. In that case, the EN Manager calls the plug-in to set up the virtual link, and the implementation of the plug-in takes care of configuring the networking equipment in the WAN properly. No Virtual Ports or callbacks are involved, and the local EN Agents play no role. Examples of such a plug-in are a northbound API from a NMS or from an OpenFlow controller managing the WAN [21].

D. Interaction with OpenStack

After the creation of a new virtual link over the WAN, the EN Agents interact with the local OpenStack instances at their respective sites to create a binding between the Virtual Port representing the local endpoint and a *Virtual Interface Identifier* (VIF-ID) that is compatible with Quantum. This identifier can then be retrieved by the tenant or orchestration layer, and can later be used to “plug” the Virtual Link endpoint into a local Quantum network as in step 8 above, creating the interconnection between the virtual wide-area link and the local virtual network.

Cloud operating systems typically use abstractions like *virtual distributed switches* [22] or, as in the case of OpenStack Quantum, more generalized abstractions like virtual networks to manage local-area virtual networks. These abstractions include the notions of virtual ports and virtual interface objects. One of the main tasks of the virtual network management system in the cloud OS is to maintain a mapping between virtual and physical ports and interfaces. The EN Agents need to perform a stitching operation in cooperation with the cloud OS to create such a mapping for a specific virtual interface that represents the endpoint of a Virtual Link. The result of the stitching operation is a VIF-ID, which can, for example, represent a specific VLAN on a specific internal port of the gateway which bridges between the site's local network and the WAN.

The stitching operation depends on the cloud OS and the underlying WAN technology. In the case of OpenStack using Quantum networks, the binding operation depends on the active Quantum plug-in. In our prototype implementation, we used the Nicira NVP [24] system with the corresponding Quantum plug-in. In this case, the VIF-ID is constructed using an NVP identifier referring to the NVP gateway and a VLAN tag that is chosen by the Elastic Networking Manager for each virtual link. The NVP gateway identifier is looked up in a pre-configured table that maps the EN Gateway objects to their corresponding NVP identifiers. This table is maintained by the EN Agents. The gateway identifier along with the VLAN tag uniquely identifies a Virtual Port on the gateway. The VIF-ID is then constructed by putting the gateway identifier and VLAN tag into a special text format that is understood by the Quantum service.

The method described in the previous paragraph is just an example implementation of the stitching operation. For the virtual network management components in other Quantum plug-ins, and potentially in other cloud operating systems, EN Agents that are specific to the cloud OS implement the specific binding operation for each cloud OS.

E. Interaction with the Underlying Network

Cloud Atlas has two axes of variation in interaction with the underlying network. One axis is what particular underlying WAN VPN technologies are supported. The other is how the existing physical network deployment and hardware is programmed to deploy the WAN virtual network. These factors determine how the EN Manager, EN Agents, and network specific plug-ins interact with the network.

Regarding the first point, the capabilities available to clients of the API depend on the capabilities of the supported VPN technologies. In particular, if the virtual network is to support SLAs, it requires virtualization capabilities that enable the partitioning of the existing physical bandwidth to support multi-tenancy. Virtualization can be performed at the edges with over-the-top VPNs at Layer 3, in which case no SLAs are supported, with logical overlays at Layer 3 or Layer 2, e.g. in the case of a managed MPLS or a carrier Ethernet [25] network in which case SLAs in the megabit/second range can be supported, or with dedicated physical bandwidth at Layer 1, e.g. in the case of Layer 1 VPNs running over a dedicated wavelength in a DWDM network in which case SLAs in the gigabit/second range can be supported. If a particular VPN technology does not support some of the parameters in an API call, they are ignored if provided.

Regarding the second point, to support over-the-top VPNs, the EN Manager only needs to support VPN gateways that terminate the IPsec or SSL session in virtual servers in the data centers or virtual or dedicated physical servers in the enterprise networks. These can typically be handled by the agent with a plug-in running in one of the participating sites. If the supported technologies include L3VPNs (based on MPLS), L2VPNs, or L1VPNs, then the EN Manager either needs code specific to the gateway networking equipment in the data centers and in the intervening network or must use a network management system (NMS). With a small network, a collection of command line interface (CLI) scripts is sufficient, but for larger, multivendor networks, an NMS is essential, particularly for topology management and path computation. An NMS will typically work through element managers that handle a specific vendor's equipment, in which case, the EN Agents play no role in configuring the hardware. If all the intervening networking equipment is OpenFlow enabled, such as for example in Google's Gscale network [23] , an alternative implementation could utilize Cloud Atlas as a northbound API to an OpenFlow controller [21].

VI. Prototypes and Future Work

As part of a research project developing distributed cloud technologies, we have deployed a testbed consisting of two simulated mini-data centers connected by a wide area networking link within a lab. The "data centers" are two racks of HP servers, one containing 19 HP DL165G5s with 32Gb of memory and 160Gb of disk, and the other containing 10 HP DL380G7s with 32 Gb of memory and 480Gb of disk. One of the DL380s was upgraded with 2 TB of storage to serve as a NAS for both data centers, so network attached storage is available. The two racks are on separate subnets connected through two 48x1G Ethernet switches. Both data centers run OpenStack Essex [26] with Nicira NVP [24] for Quantum virtual networking.

The prototype of an edge-based deployment is shown in Figure 5. We connected the Ethernet switches to two Ericsson SmartEdge (SE) 800 routers [27] and connected the routers together though a 1G link to simulate the WAN. The SE 800s are configured using CLI scripts to set up on-demand, per-Virtual Link VPLS pseudowires on fixed MPLS label switched paths. Elastic partitioning of the available aggregate bandwidth is achieved by performing traffic shaping on the gateway routers. Edge-based deployments would typically only be useful in simple networks when the routes between edge routers are fixed, since no topology discovery or path computation services are provided.

The prototype of a network-based virtualization deployment is shown in Figure 6. We removed the routers and connected the two Ethernet switches to Ericsson SPO 1410 optical/Ethernet switches which are connected together to simulate the WAN. The per-Virtual Link VPLS pseudowires are configured through Ericsson's IP Transport Network Management System (IP-TNMS). IP-TNMS manages L3- and L2VPNs across a connected network of routers and switches. The IP-TNMS exports a CORBA northbound API which is used by the EN Manager to set up the networking. Bandwidth partitioning is achieved directly by the SPO 1410s. Using a network management system simplifies topology discovery and path computation in complex networks, since these functions are typically built-in.

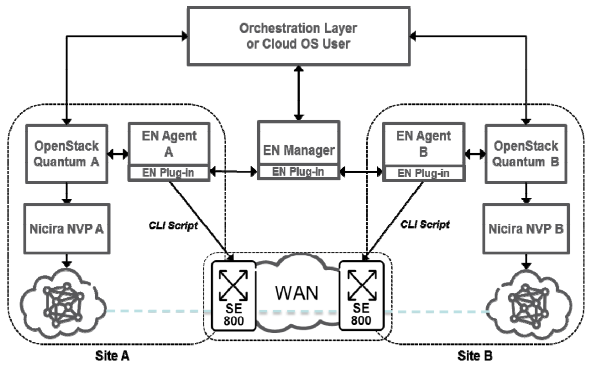

Figure 5. Edge-Based Virtualization Deployment

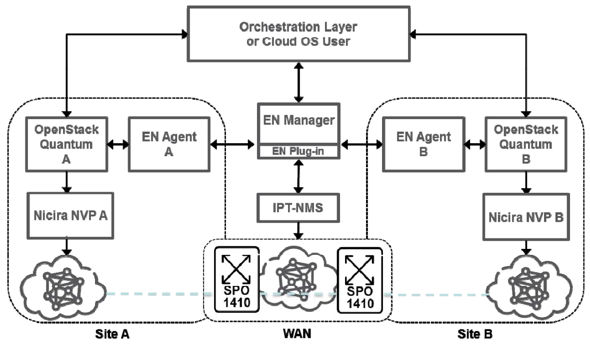

Figure 6. Network-Based Virtualization Deployment

We also implemented a VM checkpointing service that could be used for disaster recovery. The tenant checkpoints a VM through the OpenStack Horizon dashboard into the Glance image service in the other data center. Checkpointing an image of a few hundred MB takes around two minutes between two data centers in our lab network due to the NAT VM between the data centers. In a production deployment, the NAT VM would be replaced by a hardware appliance or removed. Currently this application is not automated, but it could be automated to take periodic checkpoints and start them if the originating data center goes offline. If the two data centers are in the same metro area, the connectivity could be sufficient to ensure uninterrupted service should the originating data center go down.

The performance of Cloud Atlas is intimately tied to the actual VPN system doing the work. Thus, a router CLI based plug-in requires very little time to complete because the routers just need to interpret the script and install the MPLS labels on both sides. The network-based plug-in requires more time because the NMS must perform topology discovery and path computation.

Going forward, we plan to support at least one additional VPN type in order to ensure that the API has sufficient flexibility. The current IPT-NMS northbound API does not yet support bandwidth flexing, and we are working to implement that. We also plan to tie into the OpenStack role-based authorization framework and linking that with WAN security, so that changes to networking are authorized.

VII. Summary and Conclusions

In this paper, we have described Cloud Atlas, an abstraction implemented by an API for elastic networking between a cloud OS and the WAN. The Cloud Atlas API presents a collection of abstractions to the cloud tenant that allow the tenant to dynamically create and configure VPN connections and to plug these connections into their intra-data center virtualized slices, thereby creating end-to-end virtual networks with QoS between a tenant's VMs in a data center and clients and servers outside the data center. With Cloud Atlas, networking becomes elastic and dispatchable, exactly as for computation and storage.

Cloud Atlas supports a broad variety of WAN VPN technologies, but is limited in the services that it can support by the underlying VPN technology plug-in. For example, if the VPN technology does not support bandwidth SLAs, then neither will Cloud Atlas. In addition, if a physical connection does not exist, human intervention will be necessary to make it. The analogy with a data center is when a new rack of servers or a new storage array must be installed.

Despite this limitation, we believe that Cloud Atlas has the potential to revolutionize access to the WAN in exactly the same fashion as elastic computation and storage have for data centers. Cloud Atlas reduces the complexity of deploying end-to-end virtual networks by not modeling the details of the particular VPN technologies; but rather, it provides a convenient tenant-friendly abstraction with the details tenants care about that should be adequate for most tenants. By reducing the time and complexity of providing VPN service, the deployment of new applications that require provisioned service with SLAs can be automated on cloud platforms just as the current three tiered Web applications have been. Existing provider-provisioned WAN services become self-provisioned for tenants due to the simplicity of the SDN abstractions, reducing operations costs for the network operator. With Cloud Atlas, network operators can aggregate multiple data centers into a true distributed cloud computing facility.

VIII. Acknowledgments

The authors would like to thank Martin Casado and the other folks at Nicira for providing early access to their network virtualization product NVP, and Sonny Thorelli for help managing the complexities of building an international testbed.

IX. References

[1] S. Shenker. (2012). "The future of networking, and the past of protocols" [Online]. Available: http://www.slideshare.net/ martin_casado/sdn-abstractions.

[2] J. Rexford, "The networking philosopher's problem", *Computer Communication Review*, 41(3), pp. 5–10, July 2011.

[3] OpenStack Foundation. (2013). Available: http://www.openstack.org,

[4] OpenStack Foundation. (2013). Available: http://docs.openstack.org/ trunk/openstack-network/admin/content/Architecture.html

[5] OpenStack Foundation. (2013). "Quantum Cloudpipe" [Online]. Available: http://wiki.openstack.org/QuantumCloudpipe

[6] W. Augustyn, and Y. Serbest, "Service Requirements for Layer 2 Provider-Provisioned Virtual Private Networks", RFC 4665, Internet Engineering Task Force, September 2006.

[7] E. Rosen, A. Viswanathan, R. Callon, "Multiprotocol Label Switching Architecture", RFC 3031, Internet Engineering Task Force, January, 2001.

[8] A. Sundararaj and P. Dinda, "Towards virtual networks for virtual machine grid computing", 3rd conference on Virtual Machine Research And Technology Symposium, 2004.

[9] P. Ruth, et. al., "Autonomic live adaptation of virtual computational environments in a multi-domain infrastructure", IEEE International Conference on Autonomic Computing, Washington, DC, USA, 2006.

[10] B. Bosak, et. al., "New capabilities in qoscosgrid middleware for advanced job management, advance reservation and co-allocation of computing resources --- quantum chemistry application use case", LNCS 7136 (2012) p.40.

[11] Amazon. (2013). "Amazon Virtual Private Cloud (Amazon VPC)" [Online]. Available: http://aws.amazon.com/vpc/

[12] Savvis. (2013). Available: http://www.savvis.com/en-us/infrastructure-services/network/network_professional_services/pages/home.aspx

[13] Amazon. (2013). "AWS Direct Connect" [Online]. Available: http://aws.amazon.com/directconnect/

[14] I. Baldine, Y. Xin, A. Mandal, C. Heermann, J. Chase, V. Marupadi, A. Yumerefendi, and D. Irwin, "Networked Cloud Orchestration: A GENI Perspective", GLOBECOM Workshops, December 6-10, 2010.

[15] RENCI. (2013). "Welcome to the ORCA-BEN Project" [Online]. Available: https://geni-orca.renci.org/trac/

[16] GENI. (2013). Available: http://www.geni.net/

[17] T, Wood, et. al., "The Case for Enterprise-Ready Virtual Private Clouds", HotCloud'09, Usenix, 2009.

[18] T, Wood, et. al., "CloudNet: Dynamic Pooling of Cloud Resources by Live WAN Migration of Virtual Machines", 7th Annual ACM International Conference on Virtual Execution Environments, Newport Beach, California, USA, March 9–11, 2011.

[19] G. Schaffrath, et. al., "A Resource Description Language with Vagueness Support for Multi-Provider Cloud Networks", International Conference on Computer Communications and Networks (ICCCN), 2012.

[20] G. Schaffrath, et al, "Network Virtualization Architecture: Proposal and Initial Prototype", VISA workshop, SIGCOMM, 2009.

[21] N. McKeown, et. al., "OpenFlow: enabling innovation in campus networks." ACM SIGCOMM Computer Communication Review 38.2 (2008): 69-74

[22] VMWare. (2013). Available: http://www.vmware.com/products/ datacenter-virtualization/vsphere/distributed-switch.html

[23] Open Network Foundatino (2012). Available: http://www.youtube.com /watch?v=VLHJUfgxEO4

[24] VMware. (2013). Available: http://www.nicira.com

[25] "Carrier Ethernet Network Architecture Framework: Parts 1 & 2",MEF 12.1, Metro Ethernet Forum, April, 2010.

[26] OpenStack Foundation. (2012). "OpenStack: The "Essex" Release" [Online]. Available: http://www.openstack.org/software/essex/

[27] Ericsson. (2013). "Ericsson SE Family" [Online]. Available: http://www.ericsson.com/ourportfolio/products/se-family

[28] Ericsson. (2013). "Ericsson SPO 1400" [Online]. Available: http://www.ericsson.com/us/ourportfolio/products/spo-1400