H-0269 (H0911-001) November 3, 2009
Computer Science

IBM Research Report

Virtual Network Services For Federated Cloud Computing

David Hadas, Sergey Guenender, Benny Rochwerger
IBM Research Division
Haifa Research Laboratory
Mt. Carmel 31905
Haifa, Israel

Virtual Network Services For Federated Cloud Computing

David Hadas
IBM Haifa Research Labs
Email: davidh@il.ibm.com

Sergey Guenender
IBM Haifa Research Labs
Email: guenen@il.ibm.com

Benny Rochwerger
IBM Haifa Research Labs
Email: rochwer@il.ibm.com

***Abstract*—Creating a seemingly infinite cloud computing infrastructure where services can be deployed, extended, and migrated on demand and across different administrative domains, IT platforms and geographies sets new challenges for virtual networking. This work highlights a set of basic principles and requirements from cloud virtual networking services. In view of such requirements, a novel overlay network is suggested to enable virtual network services across federated clouds. The design of such a virtual network service is described and compared with alternative technologies, and its traffic performance validated. The work described here enables the establishment of large scale virtual networks, free of any location dependency, that result in completely 'migratable' virtual networks. Network scalability concerns are mitigated by taking specific advantage of cloud computing. The service can be offered using different underlying network technologies and includes a hierarchical design to ensure the privacy, security, and independence of the federated infrastructure providers.**

I. Introduction

This paper presents research into virtual networking offered as part of a cloud service. It follows a model of virtual resource interconnect, where resources are dynamically deployed and migrated in a cloud. One such model is put forward by the RESERVOIR project [1]. Under RESERVOIR, *service applications* owned by customer service providers consume a set of dedicated compute, network, and storage resources. These service applications are deployed on top of a seamlessly infinite reservoir of virtual resources, that are uniformly available across a federation of infrastructure providers and dynamically allocated to serve the needs of the service applications.

RESERVOIR provides a technology foundation for an Internet-scale data center where resources and services are transparently and flexibly provisioned and managed like utilities [2]. The virtual resources are completely separated from the physical infrastructure that hosts them at any particular point in time and may migrate as a whole or in parts between physical hosts and sites. The RESERVOIR project vision of **Service and Resource Migration without Barriers** offers unlimited application scalability.

In order for a service application to be deployed as part of a cloud, resources need to be fully isolated, where the physical characteristics of the resource are abstracted away. As an example, compute resources in the cloud are encapsulated in the concept of a Virtual Execution Environment (VEE). VEEs abstract away the host's physical characteristics such as the CPU and memory. VEEs are also fully isolated and enable sharing of the host. Similarly, service applications in the cloud need dedicated data network resources that can be encapsulated and separated from the physical network and host resources. Such virtual network services should (1) be fully isolated; (2) enable sharing of hosts, network devices, and physical connections; (3) be able to hide the network-related physical characteristics such as link throughputs, location of hosts, and so forth.

This paper first describes the RESERVOIR service principles and derives a set of requirements from the cloud virtual network service. Next, the paper investigates the formation of a scalable service that follows the requirements, and presents a novel solution designed to conform to the RESERVOIR service principles. Unlike state-of-the-art solutions, the presented solution can be offered by a federation of independent infrastructure providers and allows the migration of complete service applications as well as individual virtual resources. Section II describes the RESERVOIR networking service model, service principles, and network service requirements. Section III discuss the state of the art. Section IV presents a novel overlay network design in view of the RESERVOIR network service requirements. Section V describes an implementation of the presented solution.

II. RESERVOIR Networking Model and Requirements

RESERVOIR is a complete framework for building an Internet scale distributed data center in which customer service providers can operate numerous service applications deployed across a federation of Infrastructure Providers. The network service model used by RESERVOIR makes a distinction between two main virtual data network services offered to service applications:

1) *Virtual Application Network (VAN)* — a service allowing application VEEs to communicate among themselves.
2) *Virtual Internet Access (VIA)* — a service allowing VEEs to communicate with entities external to the service application.

The federated network service model used by RESERVOIR includes multiple VANs per service application. As shown in Fig. 1, each VEE may be connected to zero or more VANs (each referred to as a *VAN instance*). An attachment to a VAN instance is referred to as a *VAN node*. A section of a VAN instance which may serve multiple VAN nodes is referred to

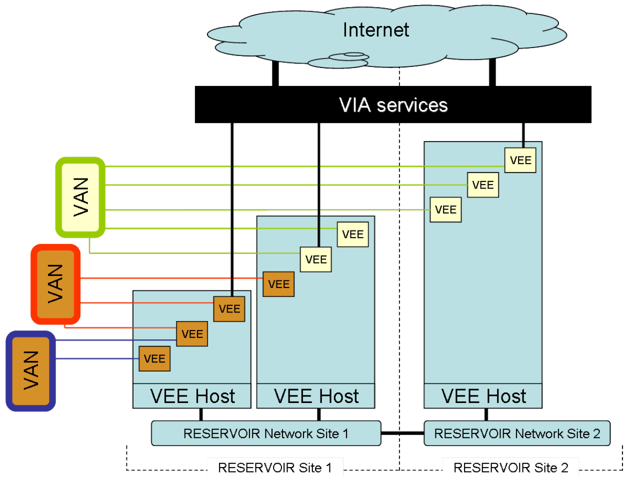


Fig. 1. Service model for federated network services. The RESERVOIR system hosts two service applications: one is composed of five VEEs, one VAN, and two interfaces to entities external to the application (using the VIA service); the second application is composed of four VEEs, two VANs, and one external interface.

here as a *VAN segment*. VEEs connected to the same VAN communicate with each other over a private network. The primary focus of this paper is the VAN service. The VIA service allowing VEEs to connect to entities external to the service application is left for further study. Note that different VEEs that connect to the same VAN may be hosted on the same host or on a collection of physical hosts. The hosts in the collection may reside at different sites of the federation. All VEEs, including those with VIA interfaces, might be subject to migration as discussed later.

A. Driving Principles and Requirements

As part of the RESERVOIR project, a set of basic principles were identified to enable Internet scale distributed data centers. In this section, these principles are briefly described and then used to derive the requirements for cloud virtual network services.

Separation - There should be full separation between the infrastructure provider and its users in order to reduce the mutual dependency. From the networking point of view, this separation principle translates into the following requirements from a VAN service:

Infrastructure agnostic: VAN nodes should be kept unaware of their physical location and the type of underlying physical networking infrastructure. Further, VAN nodes should be kept agnostic to migration and hence cannot be reconfigured or suffer from harmful loss of connectivity due to migration.

Address space independence: VAN nodes should use an address space that is independent from the one used by the infrastructure, avoiding mutual dependency.

Black box: The infrastructure provider may not assume anything about the service application internals. Therefore, virtual machines need to be treated as black boxes and offer services via a common and well established networking interface.

Isolation - Isolation of virtual services allows possibly competing service applications to securely share the resources of the infrastructure provider. Isolated VAN services need to be offered side-by-side while sharing network resources of the infrastructure provider. Offering an isolated network service by an infrastructure provider requires that all source and destination VAN nodes, as well as any intermediate network infrastructure, be completely under the control of the infrastructure provider. No aspect of the VAN implementation may be under control of the application or the VEEs. The infrastructure provider should adhere to several requirements:

Source identifiable: Any traffic entering the VAN should be strictly source identifiable and recognized as entering from a VAN node or VAN segment that was attached to that VAN instance.

Traffic tagging: Since the service application address space of different applications may collide, and since different applications may share the same physical resources, traffic entering the VAN should be tagged as belonging to a specific VAN instance. Such a tag should be carried alongside the traffic and considered during certain traffic forwarding decisions.

Destination identifiable: Traffic tagged as belonging to a VAN instance may be received by a VAN node or a VAN segment only if the specified VAN node or segment is attached to the specified VAN instance.

Service independence: The VAN service should give a service application the impression that it is using a dedicated network. The behavior of one application should not directly affect the service given to another application that shares the same infrastructure. Specifically, misbehavior of VAN nodes belonging to one VAN instance, such as packet flooding, should not affect the service given to other VAN nodes of other VAN instances.

Elasticity - The infrastructure provider should offer an elastic and extendable environment allowing customer service providers to adjust the size of their service application on demand. A VAN service needs to enable application elasticity making changes to both the number of application VEEs and the resources available per each VEE:

Dynamic sizing: In order to adjust deployed service applications to real-world dynamics, customer service providers should be allowed to add VAN nodes and remove VAN nodes from the network service without affecting the service offered to other members of the VAN.

Scalability: In order for service applications to enjoy RESERVOIR's elasticity, VANs need to be scalable. Putting together a scalable VAN service entails limiting the complexity of the information stored per device and per site while ensuring that the configuration complexity does not depend on the network size. Scalability challenges include: scalability of the number of administrative domains, hosts, and VAN nodes per VAN; scalability of the number of hosts, VANs, and VAN nodes per administrative domain; scalability of the number of VANs and VAN nodes per host.

Federation - Enabling resource migration without barriers across a federation of possibly competing infrastructure providers, requires that an interchangeable VAN service is offered by all administrative domains:

Distributed deployment: VANs may cross administrative domain boundaries, and their VAN nodes should remain unaware of being placed at remote sites.

Migratability in parts and as a whole: A VAN service should efficiently support a state in which some or all of its VAN nodes migrate while leaving no trace at the origin.

Administrative privacy: A VAN service crossing administrative domains must protect the privacy of federated infrastructure providers such that the respective administrations will not need to reveal information pertaining to their internal infrastructure to other possibly competing administrations. In particular, the service should not require that administrations share the identities or addresses of all hosts servicing the VAN service.

Administrative security: VANs crossing administrative domains must not endanger the security of any site supporting the VANs. A site should be protected against a breach occurring at other sites serving the same VANs. For example, instability in the network infrastructure of one provider should not affect the network infrastructure of other providers. A security breach at a site may translate into a security breach of any of the service applications served by the site. When an application is co-served by a group of sites, an application infringement may affect all virtual resources serving the application in one or more of the participating sites. Thus, it is required that the VAN service prevent a security breach from propagating between federated administrations, between applications, or from an application to an administration.

Administrative independence: A VAN service crossing administrative domains should not lessen the independence of the participating sites to take any inner site administrative decisions without prior coordination with other administrations. In order to allow each administration to maintain its independence, the VAN service should: (a) allow an administration to make changes to the physical hosts or network without coordination; (b) allow independent placement decisions per administration; and (c) allow independent manifestation of VAN services initiated at different sites, including setting up service application unique identifiers such as VAN-related identifiers, network addresses for end-nodes, etc.

Non-functional requirements - Beyond the four sets of functional requirements noted above, a VAN service needs to ensure automation for the addition or removal of VAN nodes from the virtual network, and for the migration of VAN nodes within the virtual network. In addition, the network service should be able to efficiently minimize the overhead per packet, optimize packet routes, minimize the operations and overhead required for the setup of routes, and so forth. Another important requirement from VANs is resiliency. VANs should offer a reliable service, by reducing the chances for failure and introducing measures to overcome illegal states.

B. Overlay Networks

Following the separation principle and the requirement to decouple the network service from the network infrastructure, the need to create an overlay network is clear. An overlay network is a network built on top of another network and serves to separate the service domain from the underlying infrastructure domain. Other alternatives such as mobile IP solutions [3], [4] present partial separation [5] and will be discussed later in this paper.

While creating an overlay network, a distinction should be made between the *overlay network service* offered to virtual entities such as VAN nodes and the *underlay network infrastructure*, which comprises the provider's physical network infrastructure connecting the physical hosts. The term 'underlay' is used here to refer to the infrastructure network that implements the delivery of packets while serving the connectivity needs of the overlay network.

The separation principle requires that VEEs be serviced as black boxes, where VEEs are served using their existing networking interface. Today's most common interface is the use of IP packets encapsulated within Ethernet frames. Hence, this paper focuses on an overlay network that services either Ethernet MAC frames or IP packets. The discussion is limited to methods that allow nodes to migrate without changing their respective addresses such as Ethernet overlay networks, Mobile IP [3], [4] and Peer-to-Peer [6].

III. State of the Art

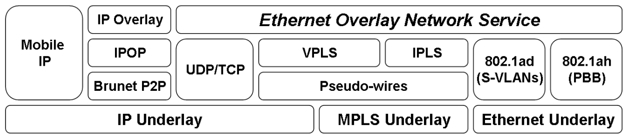


Fig. 2. Overlay network service for Ethernet or IP with different options for underlay network infrastructure

Different protocols, technologies, and research address methods for supporting overlay Ethernet and IP networks (see Fig. 2) including (1) technologies used by network service providers seeking to offer Ethernet as a Service (EaaS) [7]–[9], (2) research investigating the introduction of a virtualized network to serve migratable virtual machines [10]–[14] and (3) mobile IP [3], [4] technology. This section discusses how solutions described in the literature cope with the requirements derived from the RESERVOIR principles, as summarized in Table I. Methods used to forward packets at the overlay network are summarized in Fig. 3.

A. EaaS Offered by Network Service Providers

Ethernet as a Service (EaaS), offered by a network provider, allows enterprise customers to connect LAN fragments at multiple sites into a joint Ethernet LAN (see the Metropolitan Ethernet Forum [15]). Several standards have been developed to overlay Ethernet by network providers.

TABLE I
RESERVOIR PREREQUISITES FOR CLOUD SERVICES VS. STATE OF THE ART

Requirements	IPOP	VIOLIN	VNET	MobileIP	I/VPLS	802.1ad	802.1ah	EoIP (1)
Separation								
Infrastr. Agnostic	(2)	✓	✓	✓	✓	✓	✓	✓
Address Indep.	✓	✓	✓	(3)	✓	(3)	✓	✓
Black Box	✓	✓	✓	✓	✓	✓	✓	✓
Isolation								
Src. Identifiable	✓	✓	✓	✓	✓	✓	✓	✓
Traffic Tagging	✓	(4)	(4)	✓	✓	✓	✓	(4)
Dest. Identifiable	✓	(4)	(4)	✓	✓	✓	✓	(4)
Service Indep.	✓	(4)	(4)	✓	✓	✓	✓	(4)
Elasticity								
Dynamic Sizing	✓	✓	✓	✓	✓	✓	✓	✓
Scalability	(7)	(6)	(6)	✓	(5)	(5)	(5)	(14)
Federation								
Distr. Deployment	✓	✓	✓	✓	✓	✓	✓	✓
Migratability	✓	(8)	(8)	(8)	✓	✓	✓	✓
Adm. Privacy	(9)	(9)	(9)	(9)	(9)	(9)	(9)	(9)
Adm. Security	(10)	(10)	(10)	(10)	(10)	(10)	(10)	(10)
Adm. Indep.	✓	✓	✓	✓	✓	(11)	(11)	✓
Non-functional requirements								
Automation	✓	✓	✓	✓	✓	(12)	✓	(16)
Efficiency	(13)	13	✓	✓	✓	✓	✓	(15)
Resiliency	✓	(6)	(6)	✓	✓	✓	✓	✓

1) Ethernet over IP (EoIP): Virtual bridges at the overlay network using Transparent bridging or non-hierarchical routing protocol alternatives (Rbridges, SEIZE) with IP as the underlay network
2) IPOP may require a long recovery time after a migration
3) Address space of overlay must conform to that of the Underlay
4) Untagged forwarding
5) Supports connecting a limited number of fragments
6) A centralized solution
7) Shown to scale to about 1000 nodes
8) Does not support a complete migration
9) Information flow between administrators revealing inner structure
10) No hierarchy therefore unsecure
11) Administrations are forced to use Ethernet infrastructure
12) Uses static routing
13) Routes are non-direct
14) In case of transparent bridging: Ethernet scalability as discussed above
15) In case of transparent bridging: RSTP inefficiencies
16) Requires extensions for dynamic routing

IEEE first introduced the 802.1ad [7] standard, where provider edge bridges add a set of provider service VLANs (S-VLANs) to each incoming frame. The S-VLANs are then used by backbone S-VLAN-aware bridges owned by the provider to ensure that customer traffic is sent only to other sites connected to the same S-VLAN. IEEE 802.1ad suffers from a lack of separation between customer and provider domains, since the provider bridges take routing decisions using customer-specific internal addresses. Additionally, the use of 12-bit VLAN IDs limits the solution's scalability.

IEEE later introduced the 802.1ah standard, also known as Provider Backbone Bridges (PBB) [8], where provider edge bridges encapsulate customer traffic with a provider Ethernet header, a provider VLAN tag, and a service identifier. The added provider headers therefore separate the address schemes of the provider and the customer, and eliminate 802.1ad scalability limitations. IEEE 802.1ah moves all knowledge of the customer to provider edge bridges. The backbone core bridges remain ignorant of customer addresses. This allows the core bridges to make use of standard L2 bridging, while maintaining smaller and faster routing tables. PBB encapsulation may also enable the use of a provider multicast to serve the customer's broadcast, unknown, and multicast traffic as suggested by [16]. IEEE 802.1ay [17] suggests using PBB with static routing.

Virtual Private LAN Service (VPLS) [18], [19], or its IP-only variation (IPLS) [20], is an alternative standard for EaaS suggested by the IETF Layer-2 Virtual Private Networks (L2VPN) workgroup [9]. Under VPLS, Provider Edge (PE) devices may auto-discover each other using BGP [19], and may signal other PE devices to establish point-to-point pseudowires [18], [19], created over either a Multiprotocol Label Switching (MPLS) backbone or an IP backbone. Using pseudowires is well suited for unicast traffic, but less so for broadcast, unknown, and multicast traffic. Frames that need to be sent to multiple destinations are transmitted multiple times, once per destination. The harm from such behavior is minimal when connecting a few well-dispersed LAN fragments. Yet, using pseudowires to connect a large number of nodes, many of which may be co-located at the same site, results in considerable inefficiency. While servicing an incoming customer Ethernet frame, the PE device encapsulates the frame and transfers it via the pseudowires to a PE located at a different site, where it may be de-encapsulated and forwarded to the local LAN. Like IEEE 802.1ah, VPLS separates the customer address scheme from the network address scheme and moves all knowledge of the customer to provider edge devices. This enables the use of existing backbones (MPLS and IP) that remain ignorant of customer addresses. Therefore, both the IEEE 802.1ah and VPLS approach to EaaS use smart edges connected by an ordinary interconnect backbone.

The IEEE 802.1ah solution and the IETF VPLS solution both use a most efficient variant of overlay forwarding where the source edge sends virtual packets directly to the destination edge (see Fig. 3B).

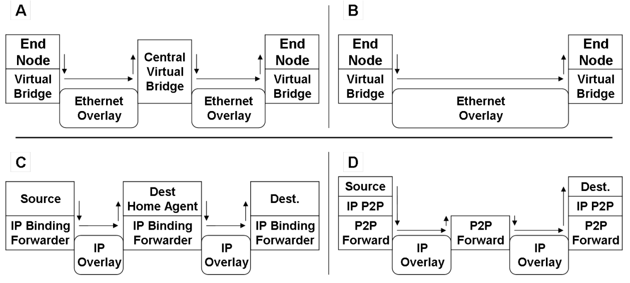


Fig. 3. Forwarding in the overlay (A) Ethernet's hop-by-hop bridging (B) Ethernet's edge-to-edge bridging (C) Mobile IP (D) IP edge-to-edge forwarding using a P2P service

B. EaaS Serving Virtual Machines

Several research groups have investigated the problem area of constructing an overlay network to serve virtual machines. Project Vine [10] uses overlay networks to virtually connect static nodes that may be located in different domains. Project VIOLIN, Virtual Internetworking on OverLay Infrastructure (see [11], [12]) use a star configuration. All nodes register to a vSwitch, which maintains a centralized route table and forwards traffic between the end nodes. The use of a centralized vSwitch forces co-located end nodes to communicate via a possibly distant vSwitch. The VNET project uses a similar approach [14] and extends it by allowing the virtual switch to take heuristic routing decisions and ask peers to establish adaptive direct connections. VNET therefore improves the solution efficiency, but remains dependent upon the centralized switch. Note, however, that both solutions suffer from scalability and resiliency issues while using a centralized vSwitch service. Another concern with the use of vSwitch emerges when considering the *migratability in parts and as a whole* prerequisite derived from the RESERVOIR federation principle that would prohibit a statically located vSwitch. Also, the *administrative privacy*, *administrative security*, and *administrative independence* prerequisites prevent a situation in which a vSwitch located in one administration serves end nodes located in other administrations. At the overlay network, VIOLIN and VNET uses Hop-by-hop forwarding (see Fig. 3A), where the source sends the packet to the destination via an intermediate node.

A different approach for servicing virtual machines is to use peer-to-peer networks such as Brunet P2P network [6], [13], [21], [22] where dispersed peers form a logical virtual ring. Each node joins the ring at a virtual location based on its unique identifier, without taking into account its physical location. Shortcuts may then be dynamically created between nodes to improve the transport efficiency along the ring. IP over P2P (IPOP) [6] suggests transporting IP packets using the Brunet P2P service, and mapping IP addresses to the P2P unique identifier using a hash function. A clear advantage of such mapping is that the translation function between the destination IP address and the respective Brunet P2P address is achieved locally at the sender. Yet, the actual packet forwarding in the P2P layer is achieved via intermediate nodes of the P2P network (See Fig. 3D). Based on IPOP, WOW [13] suggests serving dispersed virtual machines, making use of Brunet's attractive plug-and-play features to support virtual machine migration. Although Brunet P2P does not rely on a centralized switch, it suffers from issues similar to those discussed for Violin and VNET. RESERVOIR's federation-related prerequisites would prevent a situation in which a P2P node of one administration serves a P2P node located in other administrations.

All such methods establish UDP or TCP sessions between virtual network components and feed traffic sourced from virtual machines into UDP or TCP services. Therefore, all of the above methods make use of a standard IP network as the underlay network infrastructure.

C. Ethernet and Transparent Bridges

Using Ethernet as an overlay network service introduces certain concerns when considering RESERVOIR's scalability and federation-related prerequisites. In its traditional role, Ethernet scales to several hundred end nodes [23]–[25] but is limited by flooding mechanisms, broadcast service, MAC table size, and the use of Rapid Spanning Tree Protocol (RSTP). Several recent papers addressed the Ethernet scalability issues (Rbridges [26], [27], SEIZE [24] and [23]). However, the research does not specifically address issues of virtual Ethernet services or requirements derived from the *federation* principle such as *administrative privacy* and *administrative security*.

D. Mobile IP

Mobile IPv6 [3] and mobile IPv4 [4] separate the role of an IP address as a destination identifier from its role to suggest the route to the destination. Two IP addresses are therefore used at each mobile end-node. The first IP address is not location-based and is used for end-node identification in the overlay network. The second IP address is location-based and is used by the underlay infrastructure network for traditional routing. As shown in Fig. 3C, Mobile IP also uses two modes of operation. In the first mode, overlay traffic is forwarded between end nodes via an agent located in the destination end node home network. In the second mode, end nodes can optionally exchange routing information to allow overlay traffic to be sent directly between them. Mobile IP's dependence on the destination's home network introduces concerns related to RESERVOIR prerequisites such as: *scalability*, *resiliency*, *migratability in parts and as a whole*, *administrative privacy*, *administrative security*, and *administrative independence*.

IV. Virtual Application Networks

This section describes a VAN service designed to fulfill the RESERVOIR requirements. The suggested solution utilizes an overlay network between the hosts, continuing the work of Vine ,VIOLIN and VNET in order to ensure *address space separation* and establish VAN nodes that are *infrastructure agnostic*. By using Ethernet overlay as a host service rather than a function of the VM, the solution also conforms to the *black box* and *dynamic sizing* requirements.

To ensure *administrative independence*, the solution requires no more than simple connectivity from the underlay network. The solution *efficiency* is addressed by using *edge-to-edge forwarding* in the overlay network within each site. The requirements derived from the *isolation* principle including: *source identifiable*, *tagged forwarding*, *destination identifiable*, and *service independence* are addressed by controlling the inbound and outbound VANs and by tagging the frames along the path from ingress to egress.

It is common for the Ethernet overlay service to be constructed flat, without hierarchy [7], [8], [10], [11], [14], [18], [19]. This paper suggests that a hierarchical network service is required to support requirements such as *administrative privacy*, *administrative security*, *distributed deployment*, and to improve *scalability*. The solution is completely distributed in order to address *scalability*, *distributed deployment* and *migratability in parts and as a whole*. Protocols between hosts are utilized to enable *automation* and *resiliency*.

A. Hierarchical Ethernet as an Overlay Service

The suggested hierarchical overlay service is described in Fig. 4 and Fig. 5.

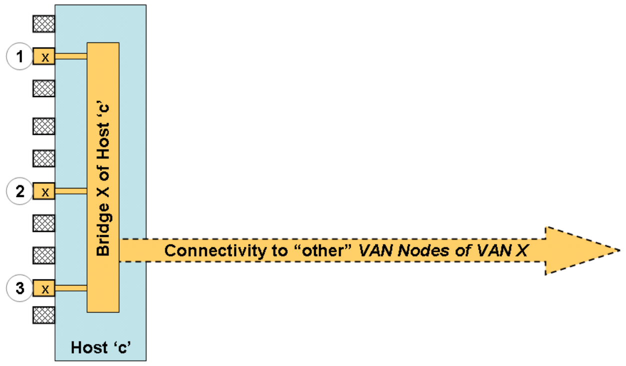


Fig. 4. An Edge Bridge is established to serve all local VAN Nodes. The X bridge instance serves only VAN Nodes that connect to VAN X while other VAN Nodes are served by other bridge instances.

First level: Each host may serve multiple VEEs via multiple local virtual bridges at the host. A single *edge bridge* instance is established per VAN at any host that serves VAN nodes of that VAN. The decision of which VAN node needs to connect to what VAN is an administrative decision with security considerations. Edge bridges should enforce the *source identifiable*, *traffic tagging*, and *destination identifiable* requirements. Edge bridges therefore have complete knowledge of the local VAN nodes served by the bridge.

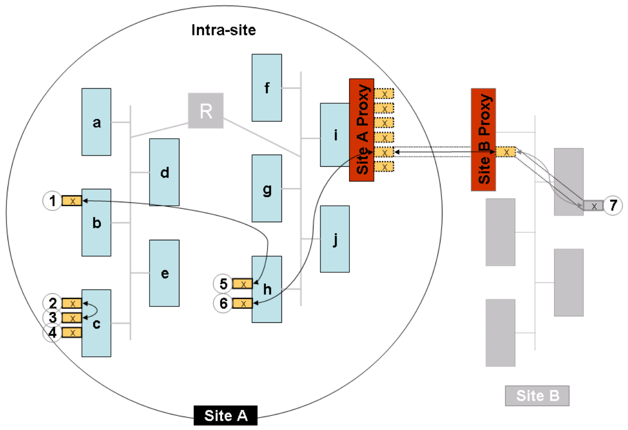


Fig. 5. The Ethernet overlay service at Site 'A' has at least the following levels: (1) Intra-host communication such as forwarding between VEE-2 and VEE-3. (2) Intra-site communication such as VEE-1 at host 'b' to VEE-5 at host 'h'. (3) Inter-site communication, such as VEE-6 at host 'h' to VEE-7 that is proxied by host 'i'. Host 'i' then communicates with the Site 'B' proxy to reach VEE-7. The Ethernet overlay service of Site 'A' is decoupled from that of Site 'B'. Site 'A' administration is unaware and unaffected by the internals of Site 'B'.

Second level: *Edge-to-edge forwarding* is used between edge bridges within the administration. It is assumed here that hosts are trustable allowing hosts of the same administration to signal a traffic tag as part of the forwarded packet. Such an assumption introduces risk, since a compromised host translates into a compromised site. Further consideration is required into mitigating site risks by reducing the risk from a compromised host. It is further assumed that hosts of the same administration may directly communicate with each other regardless of the underlay network topology. Since a single hop is used between all edge bridges, bridges are not required to use the Spanning Tree Protocol, thus overcoming some of the Ethernet *scalability* and *efficiency* concerns. Edge bridges need only to serve traffic between their locally connected VAN nodes and a subset of the non-local VAN nodes. The edge bridge is therefore not required to maintain a forwarding table to all site end-nodes, thus overcoming yet another Ethernet *scalability* concern.

Higher levels: Inter-site forwarding enables connectivity between a site VAN fragment and a fragment of the VAN located at a remote site. It is achieved using a pair of edge bridges with extended functionality, which were named as *VAN proxies*. The establishment of VAN proxy relationships between administrations is an administrative decision with security considerations. Proxies need only serve an identifiable remote proxy. Route topology between proxies are left as an administrative decision. Issues such as fault tolerance, load balancing, auto-discovery and handshaking protocols between proxies are beyond the scope of this paper.

Within the local site, the local proxy behaves as an edge bridge serving traffic to/from VAN nodes of the foreign administration. Between the local and remote sites, the local proxy behaves as an edge bridge serving traffic to/from VAN nodes of the local administration. This approach has three advantages: (1) Only a few connections are required between the administrations, representing a controllable security risk. Further, when connecting to a compromised site via a proxy, only VAN instances which are shared with the compromised site are affected. Other VAN Instances are not compromised. (2) The administrations do not share private information concerning site internals. (3) The proxies are being used to decouple the address space of the administrations and therefore ensure *administrative independence*. Proxies need only to serve traffic between a subset of local VAN nodes that actively communicate with a subset of non-local VAN nodes. Therefore, proxies are not required to maintain a forwarding table to all local or non-local end-nodes.

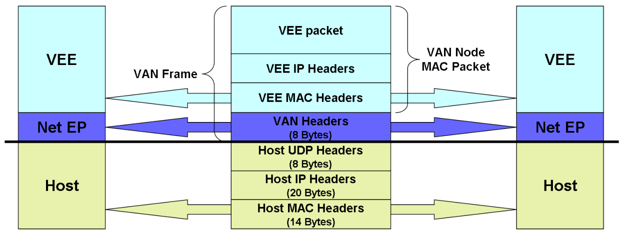


Fig. 6. The default encapsulation used by the VANs. VAN frames are encapsulated into a UDP transport.

B. Using the Underlay Network

In order to support the requirement for *administrative independence*, the service offered should be decoupled from the underlay network decision. In view of the *tagged forwarding* requirement, Ethernet frames forwarded between edge bridges should be tagged regardless of the underlay network used.

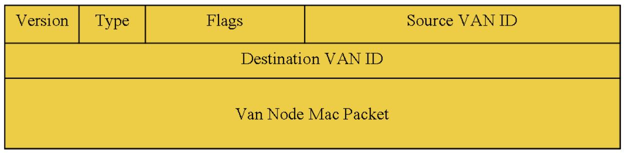


Fig. 7. VAN headers add support for tagged forwarding and enable isolation.

Different tags should be used between the same hosts when serving different VAN instances. A common example is for the Ethernet frame to be first encapsulated with VAN headers that include the appropriate VAN tag and then re-encapsulated with the host IP and Ethernet headers before being transported to a peer edge bridge at a different host. Similar tagging is also used between site proxies. Once leaving the proxy, the packet is encapsulated with VAN headers and then re-encapsulated using a standard secured network service before being transported to the peer administration.

Fig. 6 shows the default encapsulation for VANs. Other encapsulations such as direct encapsulation over IP, and direct encapsulation over MAC are also envisioned. Fig. 7 shows sample VAN headers being used by a current VAN prototype to carry the VAN tags.

C. Protocol

A broadcast domain is constructed between the VAN edge bridges. For intra-site traffic, multicasts may be used to establish part of or the entire broadcast domain, depending on the underlay network. Domains without multicast support and traffic traveling between proxies can use multi-unicast instead. The broadcast domain is used to serve any application broadcasts and for flooding unknown traffic. Ethernet makes use of flooding without feedback where learning occurs when a response from the destination is sent back to the sender. As a result, sender traffic flooding is unbounded and would stop only if and when the destination responds. The suggested VAN solution uses a learning signal between edge bridges such that a receiving edge bridge signals the sending edge bridge of the correct route. The route is then learned by the sending edge bridge.

Ethernet further uses timeout-based unlearning by bridges in order to support network dynamics. As a result, flooding is unnecessarily repeated. Because VAN edge bridges are independent from virtual machines, and routes are established between edge bridges, routes no longer depend on the virtual machine state. Therefore, as part of the suggested VAN solution, VAN edge bridges which receive wrongly routed traffic send unlearn signals to the VAN edge bridge of the sender. A health mechanism between VAN edge bridges that share a route allows the system to overcome VAN edge bridge (i.e., host) failures. As a result, unknown traffic is reduced compared to current Ethernet traffic levels. Further, building upon the said signals and health mechanism between edge bridges, an ARP proxy at the VAN edge bridges may serve local virtual machines without the need for periodical learning of the ARP entries across the underlay network. This serves to reduce the respective user broadcast traffic incurred by the broadcast service.

D. Scalability

The Ethernet scalability concerns presented above are mitigated because a) none of the edge bridges are required to learn all VAN nodes b) the spanning tree protocol is not used, and c) the amount of user broadcasts and unknown traffic is reduced.

The improved scalability offered by our proposed system takes specific advantage of the cloud computing separation between host edge bridges and virtual machines, and between virtual overlay traffic and the underlay traffic. Instead of traditional Ethernet mechanisms that require all bridges to learn all served nodes, the protocol presented here suggests that edge bridges serving local VAN nodes are required to obtain only information concerning active traffic streams to/from these local VAN nodes. As a result, no edge bridge is required to maintain a complete view of the entire network.

Another scalability concern relates to the underlay network. The use of encapsulation greatly reduced the complexity of the underlay network because it no longer needs to directly serve a large number of virtual machine interfaces. Instead, each host can be represented by a single address regardless of the number of virtual machines it serves. Furthermore, the dynamics of cloud computing including virtual machine migrations, elasticity of virtual applications, and any unsteadiness of the deployed virtual applications no longer require modifications to the underlay network. As a result, the use of encapsulation allows the underlay network to scale more easily.

Although VANs mitigate many of the scalability concerns with other suggested virtual networking solutions, further study that is beyond the scope of the this paper is required to evaluate VAN scalability limits including the number of VAN nodes that can be supported per VAN instance, per host, and per site.

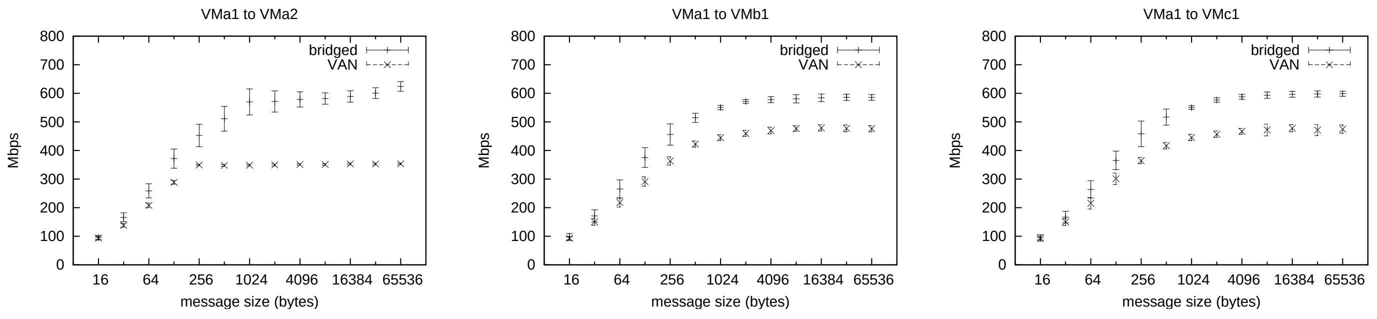
V. Implementation

In order to study the behavior of the RESERVOIR VAN framework, a single threaded user space daemon was developed under Linux to support the protocols and methods described. The implemented daemon is used in the RESERVOIR testbed and integrated as part of the RESERVOIR technology demonstration.

Validation environment

As part of the VAN daemon validation, traffic performance of the implemented daemon was compared to that of the Linux kernel bridge. Additionally tests were conducted to examine the system behavior following route changes occurring during VEE migrations.

The experimental setup when conducting traffic performance measurements consisted of three IBM System X3400 servers with two quad core Intel(R) Xeon(R) CPUs E5420 @ 2.50 GHz with 16 GB RAM. Shared storage was provided by a fourth machine with a similar configuration. All of the



(a) VMs inside the same host (b) VMs of hosts in same subnet (c) VMs in hosts of different subnets

Fig. 8. Network throughput between virtual machines in different configurations

systems were running Fedora release 10 (Cambridge) with kernels updated to 2.6.27, KVM-84 and a modified version of libvirt 0.5.1. The original libvirt package was modified to allow target VAN specification in the VM definition XML, and to connect virtual network interfaces of the created VMs to the specified VAN instance.

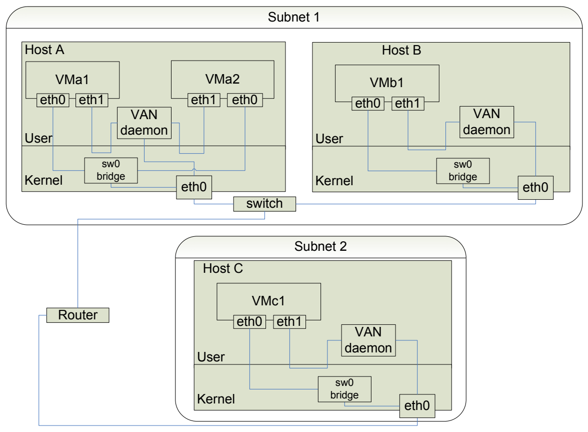


Fig. 9. Testing setup

Fig. 9 show the network setup used where two of the servers were located in the same subnet, while the third was placed in a different subnet. Each host was configured to contain a few VMs. Each VM included two virtual network interfaces. One VM virtual network interface was connected via a tap device to the standard Linux kernel software bridge as was instantiated by KVM initialization scripts. A second virtual network interface was connected via a tap device to a user space daemon that implemented the VAN protocols described above.

Validation tests

Throughput: Throughputs were measured using netperf version 2.4.3 with following command line:

```
netperf -H <target host> -l 30 -c -C --
-s 256k -S 256k -m <message size>
```

For each message size 40 tests were run for 30 seconds each and their results were then averaged.

TABLE II
NETWORK LATENCY BETWEEN VMS (MS)

source	dest	conf	min	avg	max	mdev
VMa1	VMa2	bridged	0.732	0.746	0.834	0.039
		VAN	0.755	0.768	0.850	0.033
VMa1	VMb1	bridged	0.847	0.869	0.946	0.040
		VAN	0.919	0.961	1.012	0.038
VMa1	VMc1	bridged	0.896	1.028	1.165	0.073
		VAN	0.921	0.969	1.631	0.163

The throughput test results are summarized in Fig. 8.

Latency: Network latency was measured using ping with following command line:

```
ping -q -i 0.2 -c 20 <target IP>
```

The results summarized in Table II show that the system under test had no significant impact on latency.

Migration: Impact on migration downtime incurred by VAN was measured for VEE migrations between hosts in same subnet and between hosts in different subnets. In both cases connectivity between VEES was restored roughly 100ms after migration.

Validation results analysis

Serving VMs at different hosts on the same subnet, VANs incurred 18% reduction for large packets (averaged for TCP packets larger than 16KB) and 12% for small packets (averaged for TCP packets smaller than 64B) compared to the standard bridging implementation. Serving VMs at different hosts with different subnets, VANs incurred 21% and 13% respectively. Serving VMs at the same host, VANs incurred 42% and 18% respectively.

The proof of concept uses a user space daemon that performs the VAN protocols, encapsulation, de-encapsulation and packet forwarding. The use of a user space daemon although provides for a comfortable implementation environment is non optimized for performance. Major performance pitfalls at the host/hypervisor side of the tested implementation when running on a multi-core machine and compared to the Linux kernel bridge implementation occur since traffic goes via the host kernel to the user-space daemon and from there back to host kernel. As a result transmission and reception includes two extra mode switches, two extra packet copies, one of which is a cold-cache copy. Indeed, when compared to the performance of the kernel based bridge, the VAN daemon performs less favorably. Yet, the observed difference may be mitigated by introducing a more optimal implementation that would transfer the functionality of the user space daemon to the kernel. It is left for future research to consider what is required to allow hosts serving VMs with VANs to take advantage of high speed interfaces such as 10Gbps Ethernet.

VI. Conclusions

Building scalable clouds to accommodate complete applications introduces new challenges in the area of virtual networking. The RESERVOIR project defines principals and requirements for a future Internet scale data center where virtual applications can be deployed, grow (or shrink), and migrate without barriers.

This paper describes the RESERVOIR service principles and derives a set of requirements from the cloud virtual network service. The requirements are then compared with current technologies to show that certain gaps exist. Next, the paper proposes a design for a new network service referred to as a VAN and explains how it follows the RESERVOIR service principles and meets the dervied requirements.

The presented solution can be offered by a federation of independent infrastructure providers and allows migration without barriers of complete VAN instances as well as individual VAN Nodes. The solution addresses scalability concerns both at the overlay network and at the underlay network. Yet, more research is required to evaluate the limits of the proposed design.

As a proof of concept, the VAN technology was implemented inside a user space daemon under Linux. The daemon performs the VAN protocols, encapsulation, de-encapsulation, and packet forwarding. This first implementation of the VAN technology is deployed as part of the RESERVOIR testbed.

VANs offer to service applications dedicated data network resources that are encapsulated and separated from the physical network and host resources. The offered virtual network service (1) is fully isolated; (2) enable sharing of hosts, network devices, and physical connections; (3) hides network-related physical characteristics such as link throughputs, location of hosts, and so forth.

The research described here suggests that separating the virtual network service from the physical one, achieving isolation, enabling elasticity, maintaining acceptable traffic performance, and supporting scalable federated infrastructure services is all feasible while using VANs. VAN edge bridges may be used to virtualize network resources similarly to the way hypervisors virtualize host compute resources such as CPU and Memory and can therefore be considered an extension of the host hypervisor.

Acknowledgment

The research that led to these results is partially supported by the European Communitys Seventh Framework Programme (FP7/2001-2013) under grant agreement no. 215605. Special gratitude to Ben-Ami Yassour, David Breitgand, Dean Lorenz, Julian Satran and Leah Shalev for reviewing this paper and providing valuable inputs, and to Irit Loy and Kenneth Nagin for their insights into the RESERVOIR vision and way of thinking.

References

[1] The RESEROIR project homepage. [Online]. Available: http://www.reservoir-fp7.eu
[2] B. Rochwerger *et al.*, "The reservoir model and architecture for open federated cloud computing," *IBM Journal of Research and Development*, vol. 4, no. 53, 2009, to appear.
[3] D. Johnson *et al.*, "Ip mobility support in ipv6," RFC 3775, Jun. 2004.
[4] C. Perkins, "Ip mobility support for ipv4," RFC 3344, Aug. 2002.
[5] E. Silvera *et al.*, "'ip mobility to support life migration of vms across subnets," *IBM Research Technical Paper, H-0264*, Nov. 2008.
[6] A. Ganguly *et al.*, "Ip over p2p: Enabling self-configuring virtual ip networks for grid computing," in *IPDPS*, Apr. 2006.
[7] *Provider Bridges*, http://www.ieee802.org/1/pages/802.1ad.html, IEEE Std. 802.1Qad.
[8] *Provider Backbone Bridges*, http://www.ieee802.org/1/pages/802.1ah.html, IEEE Std. 802.1Qah.
[9] PWE3 IETF Workgroup web site. [Online]. Available: http://ietf.org/html.charters/pwe3-charter.html
[10] M. Tsugawa *et al.*, "virtual network (vine) architecture for grid computing," in *IPDPS*, Jun. 2006.
[11] X. Jiang *et al.*, "Violin: Virtual internetworking on overlay infrastructure," in *ISPA*, 2004, pp. 937–946.
[12] P. Ruth *et al.*, "Virtual distributed environments in a shared infrastructure," *IEEE Computer*, vol. 38, no. 5, pp. 63–69, 2005.
[13] A. Ganguly *et al.*, "Wow: Self-organizing wide area overlay networks of virtual workstations," in *HPDC*, 2006, pp. 30–42.
[14] A. Sundararaj *et al.*, "Dynamic topology adaptation of virtual networks of virtual machines," *Proc. Seventh Workshop on Languages, Compilers and Run-time Support for Scalable Systems (LCR)*, Oct. 2004.
[15] R. Santitoro. (2003) Metro ethernet services - a technical overview. metro-ethernet-services.pdf. [Online]. Available: http://www.metroethernetforum.org/PDF_Documents/
[16] A. Elangovan, "Efficient multicasting and broadcasting in layer 2 provider backbone networks," *IEEE Commun. Mag.*, vol. 43, no. 11, pp. 166–70, Nov. 2005.
[17] *Provider Backbone Bridge Traffic Engineering*, http://www.ieee802.org/1/pages/802.1ay.html, IEEE Std. 802.1Qay.
[18] M. Lasserre *et al.*, "Virtual private lan service (vpls) using label distribution protocol (ldp) signaling," RFC 4762, Jan. 2007.
[19] K. Kompella *et al.*, "Virtual private lan service (vpls) using bgp for auto-discovery and signaling," RFC 4761, Jan. 2007.
[20] H. Shah *et al.*, "IP-Only LAN Service (IPLS)," IETF Draft, Feb. 2008.
[21] P. Boykin *et al.* Brunet software library. [Online]. Available: http://brunet.ee.ucla.edu/brunet
[22] P. O. Boykin *et al.*, "A symphony conducted by brunet," *arxiv.org*, Sep. 2007.
[23] A. Myers *et al.*, "Rethinking the service model: scaling ethernet to a million nodes," *HotNets-III*, Nov. 2004.
[24] C. Kim *et al.*, "Revisiting ethernet: plug-and-play made scalable and efficient," in *LANMAN*, Aug. 2007.
[25] H. Jang *et al.*, "Hierarchical broadcast ring architecture for high-speed ethernet networks," in *INFOCOM*, 2006.
[26] R. J. Perlman, "Rbridges: Transparent routing," in *INFOCOM*, 2004.
[27] R. Perlman *et al.*, "Rbridges: Base protocol specification," IETF Draft, Jan. 2009.