

# IBM Research Report

## Designing Modular Overlay Solutions for Network Virtualization

Liane Lewin-Eytan<sup>1</sup>, Katherine Barabash<sup>1</sup>, Rami Cohen<sup>1</sup>,  
Vinit Jain<sup>2</sup>, Anna Levin<sup>1</sup>

<sup>1</sup>IBM Research Division  
Haifa Research Laboratory  
Mt. Carmel 31905  
Haifa, Israel

<sup>2</sup>IBM STG



# Designing Modular Overlay Solutions for Network Virtualization

Liane Lewin-Eytan<sup>†</sup>, Katherine Barabash<sup>†</sup>, Rami Cohen<sup>†</sup>, Vinit Jain<sup>\*</sup> and Anna Levin<sup>†</sup>

<sup>\*</sup>IBM

<sup>†</sup>IBM Haifa Research Lab

**Abstract**—Networking is currently facing important challenges arising from the advent of virtualization and cloud computing. Broad acceptance of these advanced technologies has raised new networking requirements related to the complex configuration, management and control means needed in order to cope with large scale virtual networks of highly dynamic nature supporting multi-tenancy. Coping efficiently with these requirements has brought to the fore the need to virtualize the network by properly decoupling the logical network functionality from the underlying physical infrastructure. An essential method for enabling such a decoupling and achieve a good abstraction of the network is the use of overlays.

In this work, we investigate the main overlay solutions recently proposed and explore their main components, namely: the tunneling protocol, the control plane and the logical view. We observe that while these solutions efficiently extend the connectivity boundaries and solve the imminent scalability concerns, they do not provide a specification of an enhanced logical view. We describe the Distributed Overlay Virtual nEtwork (DOVE) solution, an architecture based on overlay networks that achieves an advanced network abstraction, allowing to define provider-tenant contracts and to provide network services at application level.

## I. INTRODUCTION

Virtualization technologies and cloud computing have been the main enablers for innovation and massive changes in the world of IT. Virtualizing workloads has led to major enhancements in key aspects such as efficient utilization of physical resources and energy saving. As a result, networks in virtualized environments face critical challenges such as the need to support large scale workloads of dynamic nature, spread in any possible location, as well as the need to support multi-tenancy. In order to properly face these requirements, it is crucial that networking technologies achieve a proper separation between the virtual and physical layers. A proper decoupling of these layers and abstraction of the network will enable the design and realization of scalable virtualized multi-tenant data centers, providing services to a very large number of entities over varying physical networking media.

Virtualized networking environments are not adequately supported by existing networking tools. Earlier technologies such as VLAN [1] are too rigid and too tied to the physical infrastructure and thus cannot allow for proper network virtualization where dynamics, flexibility and scale are required both in operation and in configuration. Initial technologies developed to address networking in a virtualized data center fail to provide a proper decoupling, thus increasing both management complexity and cost. For example, the attempts

to extend the physical network all the way to the virtual machines, such as VNTag [2], 802.1Qbg [3], while solve real problems, still expose the VMs to the physical devices and thus do not provide scalable solutions that can cope with multi-tenancy. Evolution of DCNs has emphasized the problems encountered in scaling the number of isolated networks, as well as the problems of managing these networks. Recently, it has become clear that the overlay based approach is the correct answer achieving independency from the physical networking infrastructure [4], [5], [6], [7].

Overlay networks enable the design of modular networking protocols and services in which logical functions are separated from the underlying physical infrastructure. Applications of overlay networks are various, and major examples of such are resilient routing, security, multicast, and mobility. Consequently, overlays play a vital role in advanced networking environments such as cloud computing and virtual data centers. Software Defined Networks [8], [9], [10], referring to an architecture where network services and policies are fully separated from the physical topology, takes virtualized technologies one step further. This abstraction considers the high level network logic as a set of services and policies, decoupling it from the lower level physical connectivity mechanisms used to enable these functions. Logical network functions range from the basic connectivity service to more complex and advanced services such as security, compression, acceleration and QoS. Achieving a full network virtualization layer is feasible only if built upon overlays, so as to provide these services while supporting critical requirements such as scalability, isolation, and ease of operation and configuration in a dynamic multi-tenant environment.

In our previous work [11], the novel networking requirements brought up by the server virtualization were explored, and it was shown how host-based overlays provide first answers to these new requirements. During the past year, multiple vendors have put significant efforts in order to achieve efficient overlay solutions built upon different tunneling protocols [12], [13], [14]. It has become evident that overlays should be an inherent part of the design and deployment of large scale virtual networks, and varying solutions have been proposed. In this work, we initiate a more elaborate discussion, and a better understanding of how should a good modular overlay design look like. A complete overlay solution enabling a rich logical view of the network can be roughly divided into three components (see Figure 1): (1) the logical view and application

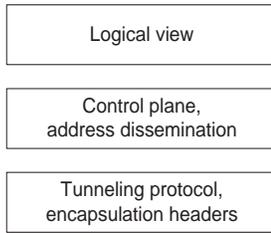


Fig. 1. Modular design of an overlay solution.

layer network services offered to the virtual end points, (2) the control plane and address dissemination method, and (3) the tunneling protocol and encapsulation format. Ideally, these components should be decoupled from each other, in order to achieve a modular design and avoid being restricted to a specific implementation method and/or vendor lock-in.

Recent solutions building upon overlays to achieve benefits of scale in multi-tenant virtual networks are VXLAN [12], NVGRE [13] and STT [14]. All these solutions ensure multi-tenancy support as well as efficient delivery within the underlying network by encapsulating tenants' L2 packets. Each of these methods is characterized by the type of encapsulation it uses, the control plane it relies on, and the logical view it supports. In this paper, we discuss the different components of these solution as well as the level of decoupling they accomplish. We present the Distributed Overlay Virtual Ethernet (DOVE) solution [15], a new architecture based on overlay networks that achieves an advanced network abstraction, allowing to enrich provider-tenant contracts with application level network services. DOVE solution allows the consolidation of multiple abstractly defined networks on large scale commodity physical infrastructures, completely delegating network administration and control to their tenants. **Paper Organization** The remainder of this paper is organized as follows. In Section I-A, we discuss the overlay solution components, as well as the main attributes that should be supported by the tunneling protocol in order to guarantee the quality of the overlay solution. In Section II, we present several important overlay solutions proposed during the past year, explore their main components and discuss the network abstraction they achieve as well as the richness of the logical view they offer. In Section III, we present the Distributed Overlay Virtual Ethernet (DOVE) solution, which has significantly evolved during the past year. Finally, conclusions and future directions are presented in Section IV.

#### A. Overlays and Virtualized Networking Technologies

An overlay network is created on top of an existing network, by generating logical communication links between hosts within the service domain. It is commonly implemented by tunneling, where a *delivery network protocol* encapsulates a *payload protocol*. The payload protocol is initiated from the higher service layer, and is intended to be transferred between endpoints within the service domain. A logical tunnel is created between these endpoints by wrapping the payload with

headers specific to the delivery protocol. Tunneling can operate at different layers of the network stack, while payload can originate either from standard or custom application protocols. The encapsulation can be either simple, assuring only connectivity above the underlying infrastructure, or be more complex and allow different advanced control functions such as explicit routing, QoS, security or traffic engineering. In addition, the tunneling control information can be exchanged either in a conceptually centralized or distributed way. As shown in Figure 1, the main architectural components of overlay-based solutions for network virtualization are as follows.

- 1) The logical view offered to the virtual end points, whether an L2 connection domain, or an extended L3 domain where endpoints can make use of advanced services and policies offered within the domain. The logical view is determined by the use of the layer's headers for control information of the logical tunnel. For example, in case L3 services are used by the end-points across the infrastructure, then appropriate control information should be carried in the respective encapsulation headers.
- 2) The control plane and address dissemination method, referring to the mechanism used to obtain the information needed to create a proper tunnel, as well as the control information that should be carried in the encapsulation headers. The control plane could be fully distributed, or rely on conceptually centralized components.
- 3) The encapsulation format, that is, the specific headers involved in the encapsulation method, along the specific control information carried across the tunnels.

As mentioned, these components should ideally be decoupled from each other in order to achieve a modular design and avoid being restricted to a specific implementation method. However, the tunneling protocol, comprising of the encapsulation format and the layers involved in the creation of the logical tunnels, has an important effect on the quality of the overlay solution achieved. In order to realize an efficient scalable solution providing a rich logical view, the tunneling protocol should support several attributes that can affect the overall quality of the overlay solution.

- The first attributes are interoperability and scalability. The delivery headers added to the payload protocol should be genuine headers used within the underlying infrastructure. This relies on the knowledge of the type of infrastructure used for delivery. In order to obtain full scalability, advanced tunneling methods could be designed to adapt to different underlays.
- Another attribute is the full exploitation of the functionalities offered by the underlying network. In order to achieve that, the encapsulated headers need to support all the relevant fields contributing to these functions. A good example is the exploitation of acceleration capacities of the physical entities. This can be performed by offload of TCP segmentation, requiring the presence of a TCP header within the encapsulation protocol in order to take

advantage of the TSO (TCP Segmentation Offload) and LRO (Large Receive Offload) engines in existing NICs. In some cases, additional functionality is required at the end points of the tunnel during decapsulation, in order to process the control information and possibly replace it in the proper fields of the payload headers. For example, consider the option of offloading checksum calculation for means of acceleration. When sending the packet, IP checksum can be calculated in the network adapter, and be placed in the encapsulation IP header. When the packet is received at the endpoint of the tunnel, in order not to lose the checksum, there's need for a transport header where the respective checksum (which can easily be derived from the IP checksum) can be placed. Another example of exploiting advanced functionalities within the tunnel (however not related to hardware acceleration) is the support of the ECN (Explicit Congestion Notification) option in the IP header to allow end-to-end notification of congestion without dropping packets. To support this function, end-points of the tunnel are required to pass this information through the IP and TCP payload headers. Losslessness is a key feature in advanced data center networks, and means by which it can be achieved as well as performance results are presented in [16], [17].

- The last crucial attribute is extensibility [18]. In order to guarantee the ability to enhance the logical view seen by the virtual endpoints, the encapsulation method used for tunneling should be extendible. This can be achieved by allocating more space for the control data within the headers, or setting a length field along optional header extensions. The richness of the environment that can be presented to the endpoints is a direct function of the control information that can be carried along the packet. While it is clear that the virtual network identification should be carried, other parameters such as policy domain within the virtual network, QoS attributes, traffic engineering parameters etc., would guarantee a richer logical environment. As a major part of the services and policies reflected to the endpoints are activated along the route, this information should be carried within the encapsulation headers.

## II. VIRTUAL NETWORK OVERLAY SOLUTIONS

We present the state-of-the-art main overlay solutions for virtual networks, namely the solutions of the key players in the network virtualization market: (1) VXLAN mainly led by VMWare and Cisco, (2) NVGRE mainly led by Microsoft, Intel and Dell; and (3) STT, Nicira's solution. In section III we present DOVE, IBM's novel networking architecture for network virtualization, focusing on its overlay solution.

### A. VXLAN

Virtual eXtensible Local Area Network (VXLAN) is a framework for overlaying virtualized Layer 2 networks over Layer 3 networks [12]. Published in August 2011, its goal is to address the need for overlay networks within virtualized data

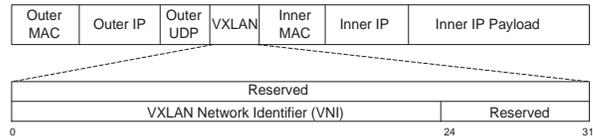


Fig. 2. VXLAN encapsulation frame format.

centers accommodating multiple tenants. Its purpose was to present a proper solution solving the main scalability problems obvious in MAC-in-MAC VLAN based solutions:

- The limited VLAN address space. The current limit of 4094 VLANs is inadequate in face of the growing embrace of virtualization, where data centers might need thousands of VLANs to correctly partition the traffic.
- The inability to support multi-tenancy, both from separation as well as scalability aspects. As different tenants are likely to be hosted over shared network resources, assigning independent MAC addresses and VLAN IDs can potentially lead to duplication on the physical network.
- The limited Layer 2 connectivity and functionality. Layer 2 is not a scalable solution for wide connectivity across large data centers. In addition, using IP for interconnection allows the use of multipathing to achieve resiliency and efficient use of network resources. This is not possible with the Layer 2 approach which uses the Spanning Tree Protocol for a loop free topology.

VXLAN provides an overlay solution for expanding a Layer 2 network over a Layer 3 network. It comprises of a stateless tunneling protocol, where Layer 2 frames are wrapped within Layer 3 packets, after adding a VXLAN header containing a 24-bit VXLAN Network Identifier (VNI). A VNI identifies a "VXLAN segment", called also a "VXLAN overlay network", thus expanding the number of unique virtual segments to 16M. An endpoint of a virtual segment is called a VTEP, and is located within the hypervisor on the hosting server (in software or hardware/physical switch); thus the VXLAN header is known only to VTEPs. The VTEP performs the encapsulation/decapsulation, and is thus responsible for the association between VM's MAC and VTEP's IP (learning from incoming packets), as well the verification of the VNI of received packets.

### Encapsulation Method

The VXLAN encapsulation format is shown in Figure 2. The inner MAC frame is encapsulated with the following headers, by order of encapsulation:

- VXLAN header comprising of flags, reserved bits and the 24-bit VNI.
- Outer UDP header, with valid source and destination ports, and (commonly) zero checksum.
- Outer IP header, with source and destination addresses being the IP addresses of the respective VTEP endpoints of the segment.
- Outer Ethernet header, with destination MAC address being the address of the target VTEP or of an intermediate router within the segment.

## Control Plane

VXLAN has no dedicated control plane, that is, there is no out-of-band mechanism that could be used for a VM to discover other hosts in its segment, their MAC and VTEP IP addresses, or any other relevant connectivity information. Rather, it uses existing Layer 2 mechanisms such as flooding and dynamic MAC learning. Layer 2 broadcast is replaced by IP multicast, by mapping a VXLAN segment to an IP multicast address, limiting the layer 2 broadcast transmissions to the servers hosting VMs in the same VXLAN segment. VTEPs can join or leave multicast groups as needed, using IGMP. Then, ARP is implemented over IP multicast to resolve MAC-to-VTEP mappings. The dependency on IP-multicast has several drawbacks. Ideally, each VNI should be assigned a different multicast group, however, in a large scale deployment of virtual networks, it is likely that several VNIs would be assigned the same group, due to the limited number of multicast groups supported by the network devices. In addition, implementing the control flow by multicast results by multiple floodings within the logical segment boundaries. Any attempt to limit the flooding might interfere with the MAC-to-VTEP learning. Lastly, the scalability of this solution depends on the type of multicast tree used. For example, in case of source base trees, the routing tables would not be able to support the multiple entries needed for large VNIs.

## Logical View & Discussion

By adding the VNI field and wrapping frames in layer 3 packets, VXLAN addressed the main concerns raised: scalability problems due to VLAN space limitations, multitenancy concerns and layer 2 connectivity issues. Following the use of standard IP headers, this encapsulation method benefits from IP address scalability, and provides interoperability across traditional Layer 3 routing boundaries. In addition, functions dependent on the IP & transport headers can be exploited (such as multipathing using ECMP, which distributes the load over different paths using a hash function over the standard 5-tuple). However, the VXLAN encapsulation method does not aim to provide an enhanced logical view of the virtual domain, but rather to efficiently extend the connectivity boundaries and solve the imminent scalability concerns. In fact, it mainly supports Layer 2 within the logical network. As to extensibility, additional control data could be placed within the reserved fields of the VXLAN header, however this option is not mentioned in the specification of this method.

### B. NVGRE

Network Virtualization using Generic Routing Encapsulation (NVGRE) was proposed as “a framework for policy-based, software controlled network virtualization to support multitenancy in public and private clouds using Generic Routing Encapsulation (GRE)” [13], published in September 2011. Similarly to VXLAN, it presents an overlay solution for virtualized Layer 2 networks over an IP infrastructure, and addresses similar problems: limited connectivity and use of resources of Layer 2 (which applies a spanning tree protocol to eliminate loops and thus converges all the flow

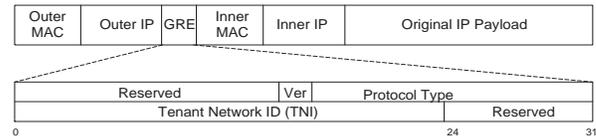


Fig. 3. NVGRE encapsulation frame format.

to specific links); scalability problems of VLAN configuration and limited address space; and the need for a scalable and secure solution supporting multi-tenancy. The key concepts addressed summarize the main goal of the overlay solution for virtualized networks using a Layer 2 logical view:

- Location independent addressing.
- Scalability of the logical networks irrespective of the underlying infrastructure.
- Preservation of Layer 2 semantics, for example, unique addresses despite migration.
- Broadcast isolation despite the large scale deployment and possible migration of workloads.

The two first points relate to the overlay solution decoupling logical and physical infrastructure, while the two last points are specific to the Layer 2 logical view provided by NVGRE (as well as VXLAN).

NVGRE provides a method for tunneling Ethernet frames in IP, using a GRE header which provides space to carry a 24-bit Tenant Network Identifier (TNI). Each TNI represents a logical network, equivalent to a logical Layer 2 broadcast domain. The tunneling mechanism is designed to be stateless, although special issues such as IP fragmentation might require some soft state to be efficiently handled. An NVGRE endpoint is a gateway between the logical and physical domain, and is in charge of the encapsulation/decapsulation of Ethernet frames to and from the GRE tunnel.

### Encapsulation Method

The NVGRE encapsulation format is shown in Figure 3. The inner MAC frame is encapsulated with the following headers, by order of encapsulation:

- GRE header comprising of the common fixed field (flags, version, type), reserved bits and a 32-bit key field, of which the lower 24 bits are used for TNI. The upper 8 bits of the key field are reserved for use by NVGRE endpoints and are currently set to zero.
- Outer IP header, with source and destination addresses referred to as Provider Addresses (PA). Here, there is a policy based option, as in case of several PAs associated with an NVGRE endpoint, the PA choice depends on the policy used for the VM.
- Outer Ethernet header, with destination Ethernet address being the MAC address of the next hop IP address for the destination PA.

### Discussion

NVGRE does not use any dedicated control plane. As to the logical view, it mainly provides a Layer 2 logical domain, while realizing virtual broadcast domains through physical multicast. In addition, it addresses limited policy and routing control by NVGRE endpoint configuration. Policy tables in

an NVGRE endpoint comprises of customer (inner) addresses, TNI and selected PA for the virtual endpoint.

While architecturally NVGRE is very similar to VXLAN, there are some differences related to interoperability and exploitation of existing network functionalities that are worth to be mentioned. On one hand, the use of GRE eases deployment over existing hardware and software stacks, compared to the need of porting a new tunneling format. On the other hand, NVGRE does not use any standard transport protocol (TCP/UDP), and thus functions depending on these headers need to be applied differently. The main example is ECMP hashing for multipathing and better use of bandwidth resources. Another problem acknowledged by NVGRE is IP fragmentation due to over-sized encapsulated frames. A possible future solution could be to use Path MTU Discovery to respectively reduce the virtual network MTU size for IP packets and avoid fragmentation.

### C. STT

Stateless Transport Tunneling Protocol (STT) for network virtualization [14] was published in February 2012, as yet another tunneling encapsulation method for building overlays for virtualized networks. It provides the aforementioned requirements network virtualization places on tunneling protocols (decoupling of logic domain from physical infrastructure with respect to topology, services, machine mobility and addressing, multitenancy support, and unlimited address space of VNs). While addressing all these aspects, a major strength of this method is its efficient use of the capacities of network interface cards to improve performance.

As the previous overlay methods presented, STT is an IP-based encapsulation. According to the STT conceptual model, an STT tunnel operates between a pair of endpoints, whether virtual/physical switches or some other devices, providing a virtual Ethernet link between the endpoints. Frames are encapsulated with an STT frame header carrying the logical information of the virtual domain as well as an outer TCP-like header before the standard IP encapsulation. The TCP header is added for pragmatic reasons, to take advantage of hardware-based TCP Segmentation Offload (TSO) support, but is stateless. Thus, there is no association of TCP-state connection to the tunnel.

#### Segmentation Offload.

As mentioned, STT was designed to use the high performance of the NIC when tunnel endpoints are located in end-systems. A NIC supporting TSO gets a large frame along with metadata for completion of TCP header. It then fragments the frame according to MSS size (for efficiency, equal to MTU), performs checksum operation and applies the appropriate headers to the segments. On the receive side, reassembly can be performed by NICs supporting LRO. Taking advantage of TSO capacities is important in the context of virtualization, as transitions between the guest and the hypervisor as well as between the hypervisor and the NIC are relatively expensive. STT allows packets to be first processed by the virtual switch which can perform different network virtualization functions,

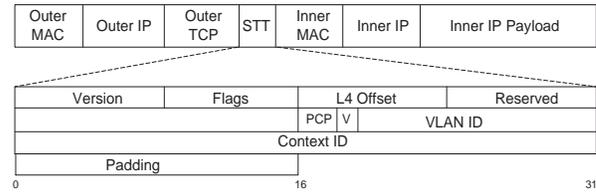


Fig. 4. STT encapsulation frame format.

and then encapsulate the packets adding the TCP-like header. The encapsulated packets can then be segmented in hardware by the NIC. It is worth to note that packet loss can be efficiently handled despite segmentation. Partial frame payload might be delivered to the TCP stack of the VM, which will deal with missing bytes as for regular transmissions.

#### Encapsulation Method

The STT encapsulation format is shown in Figure 4. The inner MAC frame is encapsulated with the following headers, by order of encapsulation:

- STT header comprising of fixed field (flags, version, type), reserved bits, and metadata (such as MSS and VLAN ID) passed for the case where the frame is re-assembled at an STT endpoint and then re-transmitted on another physical interface. The logical information is carried within the 64-bit context field, which is a generalized form of network identifier, allowing room for further extension.
- The TCP-like header used for TSO.
- Outer IP and Ethernet headers.

#### Discussion

STT does not provide any specification as to the control plane used to manage the tunnels. As to the logical view, STT allocates more space to logical data, without specifying its interpretation, thus providing more freedom for software processing. This method can be considered as fitted for soft switching with hardware acceleration. Additional features allowing efficient processing in software are the use of redundant fields in the header for more efficient lookup and padding to improve byte-alignment on 32-bit boundaries.

Clearly, the most important functionality exploited by this method is hardware acceleration, however other functions such as multipathing could be exploited as well. As distribution of traffic over multiple path is performed as a function of the source port, this can be achieved by using a method for calculating the port which provide a random distribution over the port numbers range, for example by applying a random hash on ports and addresses of the TCP-like header.

With respect to interoperability, an STT packet appears exactly the same as a TCP-IP packet, and thus can be transmitted over any IP underlying infrastructure, assuming no TCP processing occurs on the way. As no TCP state is maintained along the header, undesired behavior might occur in case STT packets pass through middle boxes that process TCP.

### III. DOVE OVERLAY SOLUTION

#### A. Modular Overlay Design

A good modular design of an overlay solution should separate the three main components: encapsulation format, control plane and logical view. After all, why should any of these components be restricted by the specification of the others? For example, the same logical view could be provided by several ways of implementing the control plane, while an encapsulation method should not be tied to any explicit control plane or logical view. Both VXLAN and NVGRE focus on the encapsulation method, while considering the essence of the logical view as an extended Layer 2 logical domain. Alternatively, STT is merely an optimized encapsulation method, and does not address the control plane or logical view (but provides room for logical view extensions).

We view the encapsulation format as just one of the elements that is part of a design of a full overlay solution. In DOVE, tunneling format is decoupled from the logical view offered by the solution, and only defines the way frames are wrapped to be transferred by the underlying infrastructure. Moreover, a logical view offered to a virtualized network could be applied using most extendible encapsulation methods.

#### B. Logical View

The Distributed Overlay Virtual nEtnetwork is a novel networking architecture that provides a clean abstraction fully separating the logical function from the underlying physical infrastructure [15]. It allows to create a network virtualization layer for deploying, controlling, and managing multiple independent and isolated network applications over a shared physical network infrastructure. DOVE meets the set of requirements presented by large scale deployment of virtualized networking technologies: scalability, multi-tenancy support, highly dynamic workloads support, exploitation of the underlying infrastructure hardware appliances, etc. However, its major strength, differentiating it from existing overlay solutions, is the logical view it provides to the endpoints. Unlike other overlay solutions, DOVE is not limited to L2 emulation (e.g. Ethernet), but considers it as a mean to carry data rather than as a service.

DOVE's abstraction captures the high level functionality required for network virtualization. The basic function is clearly to provide connectivity between endpoints. However, in today's rich IT environments, connectivity alone is not enough and much more is expected from the networking services. For example, traffic may be required to pass through network appliances (called also "middleboxes") such as firewalls, intrusion detection systems, encryption and compression engines, etc. DOVE's logical view was thus defined based on the observation that application level networking requirements are best described in terms of the connectivity between endpoints and the policies associated with it. In DOVE, a policy is a set of measures characterizing the connectivity between communicating entities (source and destination). These measures can be, among others, access control, QoS, security, or

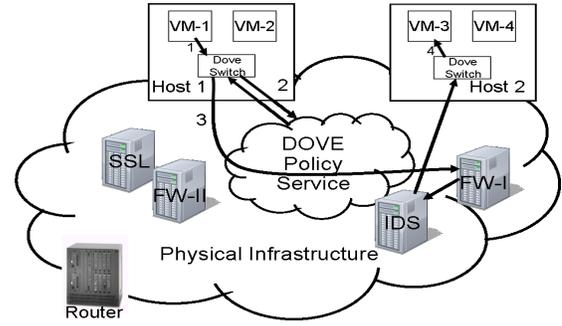


Fig. 5. DOVE control flow following transmission between endpoints hosted by two different dSwitches: Step 1 Packet interception by dSwitch hosting the sending endpoint; Step 2 Local policy lookup and, if needed, acquiring it from the DPS; Step 3 Packet encapsulation and sending through a set of appliances towards the destination dSwitch; Step 4 Packet decapsulation and delivery to the destination endpoint.

performance criteria. More generally, policy measures can be defined by a sequence of policy actions that must be applied when forwarding data between the source and the destination endpoints. Note that although the logical connectivity and policy specification is decoupled from the underlying infrastructure, its actual deployment depends on the physical networking technology and topology used, as specified policy actions must be backed up by specific infrastructure capacities.

#### C. Architecture and Control Plane

In DOVE, data traffic is handled by distributed data plane entities called *DOVE Switches* (dSwitch) while control is achieved through a control plane engine called *DOVE Policy Service* (DPS). The dSwitch entities are in charge of connectivity and policy enforcement in a DOVE environment, and get the respective control information from the DPS (if not available in local cache). Typically, a dSwitch is located on a physical server, and serves the virtual machines hosted by this server. All the traffic sent and received by a DOVE endpoint (VM) must traverse its hosting dSwitch. The DPS maintains the logical view of the network, as described in the previous section, and keeps it updated following any change of virtual network configuration. In addition, it maintains the correlation between the logical view and the physical infrastructure, including physical location of virtual machines, location of network appliances, etc. Based on this information, DPS maps connectivity requests sent from a dSwitch to packet sending instructions. DOVE's control flow is depicted in Figure 5.

**The dSwitch.** The edge of DOVE's overlay network comprises of the dSwitches, which transfer data between them using logical tunnels. A dSwitch is thus responsible of the encapsulation/decapsulation of the packets, delivering data packets using DOVE's tunneling protocol. Data is transmitted subject to the policy actions that should be performed between the specific endpoints. For example, in case of middleboxes, these actions can be enforced by controlling the data path, requiring the data to pass through the physical appliances that apply the necessitated policies. A dSwitch maintains policy

information related to its hosted endpoints. This information can be obtained from the DPS, and then stored in the dSwitch local cache using timed entries. In addition, it sends control information to other dSwitches as well as to the DPS. A main example is for reporting the location and virtual address information of hosted endpoints. Endpoint location updates should be communicated upon every detected change (for example, following endpoint migration).

**The DPS.** The DOVE Policy Service is a critical component serving the entire DOVE environment, and thus must be highly available, resilient, and scalable. The DPS maintains all the information required to resolve policy requests issued by dSwitches, and thus maintains all the information regarding existing virtual networks as well as their correlation with the physical infrastructure. This information comprises of the policy actions and rules (and mapping of these actions to the physical infrastructure), and the endpoints specification (the physical information of their dSwitch, as well as their virtual L2/L3 addresses).

#### D. Encapsulation Method

DOVE's design is not restricted to a specific encapsulation method. Rather, it could use any method relying on L2/L3 connectivity of the underlying infrastructure. In fact the two parameters that should be present in the encapsulation header are the virtual network ID and optionally a policy specifier defined by a domain ID. Virtual network ID along the endpoint virtual address uniquely identifies the endpoint, due to address space isolation between virtual networks. These fields, along others which might be needed to support policy extensions, could be put in any suitable place within the encapsulation header, according to the specific method used. In case of VXLAN encapsulation, these parameters would be placed in the VXLAN, for example by dividing the 24-bit VNI. In another case, if using the optimized STT method, these parameters would be carried within the 64-bit context field.

#### IV. CONCLUSIONS

In this work, we have investigated overlay solutions for network virtualization throughout their architectural components. Although overlays were previously proposed as an efficient solution addressing the requirements raised by the server virtualization, significant evolvments have taken place during the past year, allowing a more elaborated analysis. We presented our view of a modular overlay design, where decoupling should be achieved between the main components: the tunneling protocol, the control plane and the logical view. Furthermore, we studied the tunneling protocol, as being the lower level component supporting the functionality of the upper components, and examined the attributes it should support to assure the quality of the complete overlay solution.

We investigated the main methods recently proposed, namely VXLAN, NVGRE and STT, and discussed the nature of their control plane and the logical view they offer. While these solutions efficiently extend the connectivity boundaries and solve the imminent scalability concerns, they do not

provide specification of an enhanced logical view. However, we note that reserved fields within their headers could be used for logical view extensions in the future.

Finally, we described the architecture of our solution, Distributed Overlay Virtual nEtnetwork (DOVE). DOVE was designed focusing on the upper components, in order to provide a logical view that supports provider-tenant contracts with application level network services. DOVE's architecture has significantly evolved during the past year, and we thus provided an overview of its main components, and explained how modularity is achieved. Specifically, we presented the tunneling format as decoupled from the logical view offered, stressing that the later could be applied using most extendible encapsulation methods.

Although it is now clear that overlays provide an efficient solution for large scale, dynamic and highly virtualized networking environments, there are still many significant challenges ahead. A prominent example is the incorporation of the overlay components within an enhanced and complete SDN architecture, enabling a richer customized abstraction of the network.

#### REFERENCES

- [1] *IEEE 802.1q: VLAN*, 2005.
- [2] "Cisco vn-link: Virtualization-aware networking," 2009.
- [3] "Ieee 802.1qbg - edge virtual bridging," 2011.
- [4] T. Narten, M. Sridharan, D. Dutt, D. Black, and L. Kreeger, "Problem statement: Overlays for network virtualization," 2012.
- [5] M. Lasserre, F. Balus, T. Morin, N. Bitar, Y. Rekhter, and Y. Ikejiri, "Framework for dc network virtualization," 2012.
- [6] L. Jin and B. Khasnabish, "Architecture of psn independent overlay network," 2012.
- [7] L. Kreeger, D. Dutt, T. Narten, D. Black, and M. Sridharan, "Network virtualization overlay control protocol requirements," 2012.
- [8] P. Pan and T. Nadeau, "Software-defined network (sdn) problem statement and use cases for data center applications," 2011.
- [9] M. Casado, T. Koponen, R. Ramanathan, and S. Shenker, "Virtualizing the network forwarding plane," in *Proceedings of the Workshop on Programmable Routers for Extensible Services of Tomorrow*, 2010.
- [10] B. Pfaff, J. Pettit, T. Koponen, K. Amidon, M. Casado, and S. Shenker, "Extending networking into the virtualization layer," in *HotNets*, 2009.
- [11] K. Barabash, R. Cohen, D. Hadas, V. Jain, R. Recio, and B. Rochwerger, "A case for overlays in dcn virtualization," in *Proceedings of the 3rd Workshop on Data Center - Converged and Virtual Ethernet Switching, Dc-CaVES*, 2011, pp. 30–37.
- [12] M. Mahalingam, D. Dutt, K. Duda, P. Agarwal, L. Kreeger, T. S. M. Bursell, and C. Wright, "Vxlan: A framework for overlaying virtualized layer 2 networks over layer 3 networks," 2012.
- [13] M. Sridharan, K. Duda, I. Ganga, A. Greenberg, G. Lin, M. Pearson, P. Thaler, C. Tumuluri, N. Venkataramiah, and Y. Wang, "Nvgre: Network virtualization using generic routing encapsulation," 2011.
- [14] B. Davie and J. Gross, "A stateless transport tunneling protocol for network virtualization," 2012.
- [15] R. Cohen, K. Barabash, V. Jain, R. Recio, and B. Rochwerger, "Dove: Distributed overlay virtual network architecture," 2012.
- [16] A. S. Anghel, R. Birke, D. Crisan, and M. Gusat, "Partition/aggregate in commodity 10g ethernet software-defined networking," in *IEEE 13th Conference on High Performance Switching and Routing (HPSR)*, 2012.
- [17] R. Birke, D. Crisan, K. Barabash, A. Levin, C. DeCusatis, C. Minkenberg, and M. Gusat, "Cross-layer flow and congestion control for datacenter networks," in *Proceedings of the 3rd Workshop on Data Center - Converged and Virtual Ethernet Switching, Dc-CaVES*, 2011, pp. 44–62.
- [18] S. Bradner, B. Carpenter, and T. Narten, "Procedures for protocol extensions and variations," 2006.