Recently, there has been a plethora of blogs and discussions debating various issues around the viability of network virtualization and overlay networks. Can overlay networks scale? Is there a need for a tight coupling between overlay and underlay networks? This blog post is an attempt to bring some clarity in this debate by starting with some background and providing the guidelines for an operationally efficient network virtualization design.

**Background on Overlay Networks**

The core idea of an overlay network is that some form of encapsulation, or indirection, is used to decouple a network service from the underlying infrastructure. Per-service state is restricted at the edge of the network and  the underlying physical infrastructure of the core network  has no or little visibility of the actual services offered. This layering approach enables the core network to scale and evolve independently of the offered services. *So, are overlays a new concept or technology? Or may be an application of a well understood concept to a new new environment?*

Let us take the Internet as an example. The reality is that the Internet itself is nothing more than an “overlay network” on top of a solid optical infrastructure. The majority of paths in the Internet are now formed over a DWDM infrastructure that creates a virtual (wavelength based) topology between routers and utilizes several forms of switching to interconnect routers together. That is, if you are lucky. Because there is also a large amount of paths in the Internet that are still overlaid over SONET/SDH TDM networks that provide TDM paths to interconnect routers. Therefore, pretty much every router path in the Internet is an “overlaid” path.

But the Internet is a wide area network and may be not well understood in an enterprise or local network environment. The situation there is no different. In most cases routers are interconnected together through Ethernet switched networks and they are essentially “overlaid” on top of a layer-2 infrastructure. A router has absolutely no visibility in the layer-2 paths, and if communication between routers is lost because of a loop in the underlying Ethernet network, the routers will never be able to recover. They just declare each other as unreachable. A packet loss in the Ethernet network is not visible in the routers, unless it is correlated through some intelligent mechanism.

One could argue though, that this is still not an IP in IP overlay and it is thus different. Then let us consider the wireless network. Anyone that has little knowledge of 3GPP standards or the majority of large scale WiFi deployments will quickly recognize that pretty much every wireless service is provided through an overlay network of tunnels (called GTP this time). Essentially, traffic from each mobile subscriber is encapsulated in a tunnel and routed to the S-gateway and P-gateway (or SGSN and PGSN) of the wireless network. Mobility and separation of services from the underlying infrastructure mandates the use of tunneling technologies in order to decouple the location and identification of an end-node and allow fast mobility. Therefore, every time you access the web over your mobile device, remember that you are using an overlay network.

One could also argue that both the Internet and the wireless network are mainly consumer services, and business services are never provided in an overlay. It turns out that this not the case either. If we look at the technologies that have supported managed and unmanaged VPNs for the last several decades they are all based on overlays. The whole idea of MPLS L2/L3 VPNs is essentially an overlay network of services on top of an MPLS transport network. The Label Edge Routers (LERs) encapsulate every packet arriving from an enterprise site with two labels. A VPN label identifying essentially the enterprise context, and a transport label that identifies how the packet should be forwarded through the core MPLS network. In a way, this is a “double” overlay. Unless of course enterprise services are still using FrameRelay or DS3 lines that are yet another form of overlay. (Yes, DS3 lines over FrameRelay still exist).

Therefore, from a deployment standpoint, there is an extremely high probability that every time a user, whether consumer or enterprise, is accessing any network service, their packets are routed through one or most likely multiple overlays, that very often are not aware of each other.

**The Three Questions**  
If overlay technologies are so prevalent in networking, then what is “new” with network virtualization in the cloud environment, and what is the scope of the arguments? There are three questions that must be answered:

* *Can overlay networking scale at data center capacities?*
* *Could a correlation between overlay and underlay networks improve the services offered in a data center environment?*
* *If the second question is true, are there open standards that can achieve this, or is there a need for some new protocols or technologies?*

**Can Overlay Networks Scale?**

The main technique of overlay networks is to maintain state information at the edges of the network and “hide” individual  services from the core network. This approach essentially decouples the services from the underlying physical infrastructure.

Let us consider a common data center topology with racks of servers interconnected by a core IP network. The Top-of-Rack switches (ToRs) are configured as routers, where all servers attached to a given ToR belong to the same subnet. In this case, the number of routes that are visible at the core of the network is no more than the number of racks in the data center. It is easy to realize core networks that support thousands of racks with hundreds of thousands of servers with a core switch that supports no more than a couple of thousand routes. And at this time, any decent core switch can support this functionality.

In such a physical infrastructure, overlays are initiated either at the hypervisor or the ToR switch, and the addresses of individual virtual machines or services are never visible at the core the network. Of course, the mapping of packets to overlay tunnels at the edge of the network requires state, and depending on the number of virtual machines  the total number of routes (or address to tunnel mappings) in the data center is at least equal to the total number of virtual machines or overlay end-points. Therefore, the control plane scalability must be addressed as well.

Consider any single server in the data center, and assume that the server is hosting a number of VMs *V1*. Each one of these VMs is in a private network together with a set of other VMs *V2.* Or, in other words every VM in the data center does not require connectivity to every other VM, because if this was the case, there would be no need for network virtualization in the first place. Therefore, the total number of routes that need to be populated in any hypervisor is no more than *V1 x V2*. Even by assuming that the capability of processors increases, and the number of VMs per server increases, the memory requirements for such state is not significant. (As an example, 100 VMs in a server with 100 VMs per network will lead to 10K routes in the hypervisor. Even if the state information was 1K bytes per route, the total memory needed for this state is no more than 10M bytes, that is minuscule  compared to the memory needed to host the 100 VMs). Therefore, the state information per hypervisor is also well contained.

The other potential bottleneck of scalability is the control plane that distributes these routes, but there are several techniques to scale this. One could utilize the same techniques that have been deployed in the Internet and state distribution protocols like BGP to quickly address the scalability of the control plane (see [draft-sb-nvo3-sdn-federation-01](https://tools.ietf.org/html/draft-sb-nvo3-sdn-federation-01) as an example). Alternative techniques based on distributed system algorithms can also be utilized. A separate blog will address the scalability of the control plane and the trade-offs when designing such a system.

The above discussion makes it very clear that not only the scalability of overlay networks is not an issue, but the introduction of overlay networks simplifies the core network as well.

**Is Correlation Needed between Overlay and Underlay?**

As we mentioned earlier, the second question is whether there is a need for a tight coupling between overlay and underlay networks. There are are actually two schools of thought in this question, and the answer will depend on what one tries to achieve. For those already tired with this blog and to minimize the suspense, the answer is “It depends”. Curious readers can follow the details for the explanation.

Again, before answering the question, let us briefly visit how overlays have been structured in other networks, and let us start with optical networks first. The whole idea of the a tight coupling and optimization of the services offered by a DWDM network and the routers, or the *“converged IP/Optical”* network has been literally the holy grail in backbone design for the last 15 years. I still remember the first IETF session discussing about the concepts of GMPLS in 1999. Just last week I was in a discussion with a major backbone provider in North America, and I was told for the \*Nth\* time, that this will never happen. The GMPLS-UNI as defined by IETF is still not being used by the majority of network installations and the teams qualifying and providing optical services are completely different from the packet teams. After 15 years of discussion of IP/Optical convergence, the reality is that the “underlay” optical network is built and managed without considering the “overlay” IP network, and the IP network is perfectly happy to operate in these conditions. Thus, there is a clear proof that a reliable overlay service can be built and operationalized without any coupling. The question is at what cost. An indeed, the cost of 1+1 redundant paths or fast-restoration where the optical network tries to “guess” the services running can often lead to significant control plane complexities and expensive solutions.

Contrary to optical networks though, MPLS VPNs rely in a relatively tight coupling of overlays and L1 paths. (Prefer to not use the term physical paths, since as described in most cases the link between two routers is actually an overlay path). MPLS paths are setup through a careful understanding of the physical topology, and VPN services are overlaid over transport paths with full visibility and correlation between transport and services. The reason for these selections is to allow features like traffic management, fast traffic restoration, etc. The key concept in these networks is that instead of relying on some end-to-end probing mechanisms, they can utilize control plane protocols that will quickly detect impairments and try to restore traffic paths for the overlay network. In reality, the “underlay” or transport network provides a “service” to the overlay, and this service has well defined SLAs in terms of attributes such as quality of service and restoration times.

The reason for the transport path selection in the MPLS world though is that an MPLS network is often a complex graph with several potential bottlenecks, asymmetric and variable latency links. The optimization of transport paths allows some traffic to bypass the shortest path if this path is congested. But would a data center network require a similar function? If an MPLS network needs non-shortest path routes, why not a data center network? Unfortunately, the answer to this question is again “it depends”. (Sorry, but the world is never black and white).

**Data Center Fabrics**

First of all, let us think about the physical topology of a data center. At this time it is well understood that some form of **leaf/spine architecture** or **Clos network** provides a relatively simple method to scale the physical infrastructure and has some important characteristics:

* It scales to relatively large capacities, although not infinite.
* Through simple load balancing, it provides node and link resilience.
* It is easy to manage through any IGP (OSPF, ISIS or even BGP).
* It removes the need for any single node to be highly reliable, thus significantly reducing the cost of the core network nodes.
* It is loop free and does not need to care about STP or any other complex protocol.
* It provides a re-arrangeably non-blocking fabric with equal cost between any two leaf nodes. This is actually one of the most important properties since it allows the placement of applications at the leaf nodes without requiring visibility of the network topology.
* If we consider the universe of “traffic patterns” that are admissible by such a fabric, it actually maximizes the number of traffic patterns that can be supported. Indeed, any traffic pattern that does not over-utilize any leaf node (i.e. any combination of traffic flows from ingress to egress leaf nodes that does not over-utilize an egress node is admissible in this fabric). Notice, that this is unlike rings or non fully meshed architectures that are actually blocking and significantly limit the number of admissible traffic patterns in the network.
* It is impossible to build any fabric that can admit the same number of traffic patterns a Clos network in a “cheaper” way.

Therefore, if we consider a standard leaf/spine architecture, the question that one has to answer is whether any form of traffic engineering, or QoS, at the spine or leaf layers can make any difference in the quality of service or availability of flows. In other words, could we do something intelligent in this fabric to improve network utilization by routing traffic on non-shortest paths? Well, the answer ends up being an astounding NO, provided that the fabric is well engineered. And what we mean by well engineered is that the core link bandwidth, or in other words the bandwidth from leaf nodes to core nodes is actually higher than the bandwidth between the servers and the leaf nodes, or alternatively that some oversubscription is available at the core nodes. In other words, if the server to ToR bandwidth is 10Gbps and the ToR to spine bandwidth is also 10Gbps, and if we assume that servers can create single flows at the 10Gbps rate, then it is easy to create a traffic pattern where randomization or ECMP between the leaf and spine nodes will fail and a given core node will be oversubscribed, even though none of the egress node of the fabric are oversubscribed. As the ratio between access bandwidth and core bandwidth decreases, the probability of such an event decreases, and it can be actually proven that with a certain over-capacity, the network becomes completely non-blocking. (A detailed blog in this design will also follow. Note, that the fluid approximation of such a system is not always matching a packet based system.)

Thus, if the data center fabric is fully non-blocking as in the case of a Clos network, there is no advantage whatsoever of any visibility of actual application flows in the core network, other than potentially through an entropy field in the tunneled packets that will allow seamless load balancing.

**Multi-fabric Interconnects**

Even though a Clos network has all the properties we outlined above, it is true that not all traffic patterns are desirable in every data center scenario. The reality is that applications will most likely use some form of clustering and to a certain extend building a fully non-blocking fabric when not all traffic patterns are important is an overkill.

In addition, a single leaf/spine network is a single availability zone, and very often large cloud providers will require that the network is partitioned to multiple availability zones, with completely independent routing implementations in order to maximize resiliency. Think of it as “share-nothing” availability zones.

In addition, the cost of cabling between leaf and spine nodes can become prohibitively expensive once a certain size is realized. If you assume N leaf nodes and M spine nodes, then one would need NxM cables, and as N becomes large the distance between leaf and spine nodes will also become large, and this will create a requirement for better optics (Single Mode Fiber) that can significantly increase the cost of the infrastructure. Indeed, it is possible, that beyond a certain fabric size, the cost of optics and cables will far exceed the cost of the electronic switches in such an architecture. Thus, for any practical purpose, there is a point where compromises must be made, and the non-blocking assumption must be removed. In these cases, the topology is often transformed to a interconnection of leaf/spine fabrics through more expensive systems that limit the the cross-section bandwidth. The result of such expansion is that some of the basic assumptions of a Clos fabric are not valid any more:

* The number of hops between any two servers is not equal any more.
* The available bandwidth between any two servers is also not equal any more.

In these cases now, an overlay network deployment that completely ignores the properties of the underlay physical infrastructure can lead to congestion and reliability issues with a very high probability. There is a need for some form of intelligence and some correlation between overlay and underlay networks.

To summarize, the above discussion, correlation between overlay and underlay is needed in some situations, when the underlay or physical network infrastructure does not provide a fully non-blocking fabric for the overlay. This can happen either due to bad design of the physical network or most likely because of the need for large scale or increased availability.

**Proprietary or Open Solutions?**

Since the answer to the previous question was “may be”, we might as well try to address the last question. Can the problems be solved with open and standards based and multi-vendor solutions, or is there a need for new technologies? Does a data center operator need to buy the complete solution from a single vendor and get locked in a single solution, or can they choose an open  
path supported by multiple vendors?

The answer is clear. Open networks and existing protocols are more than enough to address these problems. In these section, we will briefly illustrate the methods to achieve this. Subsequent blog posts will introduce additional details.

The root issue with the problem is that the overlay control plane makes forwarding decisions that are not coordinated with the underlay network. A decoupled overlay controller will decide to which tunnel to encapsulate a packet, but not how to route the packet to the destination hypervisor and whether the packet can even be routed there. From an application point of view, there can be two fundamental problems. Either complete lack of connectivity or most likely some performance degradation since in most case, the infrastructure will have enough redundancy to avoid a complete connectivity failure.

The first service that the network has to provide is notify the application about a possible performance degradation. This can be easily done by the overlay controller if it has visibility of the underlying physical infrastructure. And this can be achieved in two steps. First, the overlay controller can peer with the underlying physical network and listen to the corresponding routing protocol messages. This can be done today by a simple IGP listener, and the community is actively working on BGP-LS to extend these capabilities. Second, the overlay controller must be able to collect link utilization and alarm information from the physical nodes in order to detect if a link is over-utilized and some applications are affected by this. This can be done by supporting simple network management interfaces in the underlying topology and this is pretty much standard functionality supported by any physical switch.

Once the overlay controller has visibility of the topology and the link utilization, it can perform a simple correlation analysis and it can detect the applications and/or services that are affected by a failure or congestion. This requires of course an intelligent algorithm that can quickly parse the graph and scale to the thousand of services in a data center.

Last but not least, the overlay controller must react on such congestion or failure and must try to improve the performance for the application. This reaction is the most tricky part, because unlike MPLS networks that allow explicit route paths that bypass the shortest path, a data center overlay network that relies strictly on IP, has no mechanism to bypass these shortest paths.

Fortunately, there are other mechanisms to achieve the same. The controller can rate limit other applications (less important) that are sharing the same paths. It can notify the application or cloud management system and the VMs can be moved to a different location that will alleviate performance. It can inject routes in the topology that will affect the IGP weights and therefore alter the routing behavior. It can create multi-hop overlay paths to bypass congestion. Last but not least, if the core network supports Openflow or some form of Openflow, then the overlay controller can utilize these capabilities to bypass shortest but congested paths.

The above are just some of the tools that an overlay controller can use today to achieve the same or similar performance characteristics in a data center network as a carrier grade MPLS VPN network.

**Summary**

In the quest for an answer to the often repeated question of whether overlay networks scale and whether a tight correlation is needed because overlays and underlays, always look deeper than the marketing messages. There are obviously no silver bullets, and very often the right network design demands a strong correlation. The idea that physical networking can be ignored and treated as a black box can lead to severe issues if the right topology is not deployed or if the demands for scalability make the right topology prohibitively expensive.

In all cases though, the answer is not a lock-in into proprietary solutions and protocols, but the reuse of tools and capabilities that have been well tested in then networking community and are been actively developed and expanded.