Integration of EVPN in Kubernetes

Potter, F.J.M.
MSc Security and Network Engineering
University of Amsterdam
frank.potter@os3.nl

Supervisor: De Groot, A.
Cumulus Networks
attilla@cumulusnetworks.com

October 16, 2019

Abstract

Recent development in data center architecture requires distributing of tenants over the available hardware. This optimizes resource utilization and thus cuts back costs. However, some tenants could be malicious and therefore hard multi-tenancy is required. Current Container Network Interfaces (CNIs) in Kubernetes use network policies to circumvent this. However, in an orchestrated containerized environment the scalability of those network policies can become an issue. As an alternative this research introduces EVPN as CNI in Kubernetes, that does not rely on network policies to provide hard multi-tenancy.

In order for EVPN to provide for hard multi-tenancy without network policies, an architecture based on the Kubernetes implementation was created. This architecture made use of a Virtual Routing and Forwarding (VRF) and a virtual bridge per tenant. The VRF only contained the routes of containers that belong to the tenant, such that the tenant can only reach its own containers. Beside the L3 isolation provided by the VRF, a virtual bridge prevents malicious tenants from sniffing L2 traffic and spoofing the anycast SVI to become a MITM.

The architecture was tested in a Proof of Concept and showed that inter-host connectivity and multi-tenancy could be offered by EVPN without the use of network policies. Allowing EVPN to be used as an alternative CNI in Kubernetes that does not rely on network policies.

Keywords — EVPN, Multi-tenancy, Calico, Cilium, VRF, CNI
1 Introduction

Network Virtualization Overlays (NVOs) are used to provide logical separation over a shared physical network infrastructure [1]. Examples of NVOs are Virtual Extensible Local Area Network (VXLAN) and Network Virtualization using Generic Routing Encapsulation (NVGRE). NVOs were created as a response to the increasing network demand from server virtualization and are more and more used in data center environments. Server virtualization allowed for a physical infrastructure to be shared between multiple tenants [2]. Multi-tenancy can be interesting for smaller tenants that only want a few containers that are spread over a few hosts for redundancy. Smaller tenants do not want dedicated hosts, but rather share those hosts with other tenants to allow for tight packing of containers, which could optimize resource utilization and thereby reduce costs [2]. Since the other tenants are unknown and might be potentially malicious, tenants want complete isolation from other tenants (i.e. hard multi-tenancy) [3]. NVOs make use of network policies to provide hard multi-tenancy. Network policies give control over the communication between containers and prevent one tenant’s traffic to reach a different tenant [3]. However, in an orchestrated containerized environment, where one server can host hundreds of containers, the scalability of those network policies can become an issue [4].

Ethernet Virtual Private Network (EVPN), that can be added to existing NVOs, can provide for multi-tenancy by logically separating the communication between pods of different tenants [5] [6] [7]. Therefore, it can prevent tenants from communicating in the first place, rather than blocking the undesired traffic later. Meaning that EVPN does not rely on network policies and the scaling issues of those network policies in a dynamic container environment. EVPN has been adopted by Free Range Routing (FRR) in version 4.0, that was released in March 2018. FRR is an IP routing protocol suite for Linux and Unix platforms and allows for EVPN to be integrated on the host. This paper focuses on the analysis of EVPN on the host and how to integrate EVPN into an orchestrated containerized environment. The container orchestrator that is used in this research is Kubernetes, since Kubernetes was found to be the most actively used container orchestrator in the industry [8]. In order to evaluate EVPN as a multi-tenancy solution, we investigate the following research question:

"How can EVPN provide for multi-tenancy in a Kubernetes containerized environment?"

Two sub questions were derived from this question:

1. "What are the advantages of EVPN as a solution to implement multi-tenancy in a Kubernetes orchestrated container environment?"

2. "How can the recent improvements in FRR allow for EVPN to be integrated into the Kubernetes container orchestration platform?"
2 Related work

2.1 Network Virtualization Overlays

In order to describe the problem statement of the increasing demand for server virtualization in data centers and the lack of the data center networks to support this increasing demand, Narten, et al., [2] created RFC 7364. Server virtualization provides numerous benefits, including higher utilization, increased security, reduced user downtime and reduced power usage. However, the underlying network also needs to support this increasing demand for server virtualization. According to Narten, et al., data center networks can overcome this increasing demand by using a NVO that provides multi-tenancy. RFC 7364 describes traffic isolation as the main requirement for multi-tenancy. Another requirement that Narten, et al., give is address space isolation. Address space isolation means that different tenants can use the same address space within isolated virtual networks. Address space isolation becomes possible with traffic isolation, whereas the address space is only visible to the tenants within the same isolated virtual network [2].

RFC 7364 provides a good description of NVOs as a solution to the increasing demand for server virtualization. However, the main focus of RFC 7364 is on Virtual Machine (VM) virtualization instead of container virtualization. Nevertheless, we still rely on the terms and definitions specified in RFC 7364 in our paper, since these terms and definitions are also relevant in container virtualization and container networking.

2.2 EVPN as a control plane NVO

RFC 8365 [9] describes EVPN as a control plane NVO solution to address the requirements of a multi-tenant data center. The control plane is used to provide the management of traffic flows between NVO endpoints [1]. Whereas the traffic flows between NVO endpoints itself are considered as the data plane. The data plane enables the forwarding of traffic in the network and the control plane can provide logic to the data plane. Sajassi, et al., [9] specify the isolation of network traffic per tenant, the support for a large number of tenants, extending L2 connectivity and MAC mobility as the key requirements of EVPN as a control plane NVO solution. Furthermore, Sajassi, et al., state that the scaling properties of the control plane for a NVO are extremely important.
2.3 Container networking

Makowski and Grosso [10] conducted a study into the evaluation of virtualization and traffic filtering methods for container networks. They claim that container networks in essence are NVOs where traffic between containers is encapsulated and sent through a tunnel. Furthermore, they claim that multi-tenancy is a major feature for modern NVOs. Therefore, Makowski and Grosso compared Identifier Locator Addressing (ILA) and EVPN based on multi-tenancy and ease-of-use. From this comparison it was found that EVPN was more flexible than ILA in terms of multi-tenancy and that EVPN was a more mature solution. Moreover, they state that ILA would offer a lightweight way of creating virtual networks that spans across multiple administrative domains, but requires more development effort before it could be used in a production environment. Lastly, Makowski and Grosso state that even though EVPN and ILA are capable of creating multi-tenancy in an orchestrated containerized environment, they both lack integration with a container orchestrator (e.g. Docker Swarm or Kubernetes). However, Makowski and Grosso state that EVPN can be adopted in a container network without much development effort [10]. Therefore, in our research we extend their study by proposing an architecture and Proof of Concept (PoC) of EVPN in container networking.

3 Background

In the background we describe Kubernetes and the networking rules it needs to comply with. We also discuss the most important CNIs that offer multi-tenancy as a comparison for EVPN. After discussing those CNIs, we go in more detail on how EVPN is added to BGP and the route types it provides.

3.1 Kubernetes

Kubernetes was created in 2015 as an open source cluster manager for container platforms, such as Linux Containers (LXC), Docker, etc. [11]. Kubernetes manages container platforms by having a cluster with a master and one or more nodes. The master can schedule and scale containerized applications through an Application Programming Interface (API) that communicates with the Kubelet agent on each node, as shown in Figure 1. Each node can have a pod, which is a group of one or more containers. Each pod shares resources such as storage volumes, network namespace and container specific information [12].
Since containers in the same pod share the same network namespace they can communicate with each other via localhost. Containers in separate pods are isolated from each other, unless pods use a virtual bridge to allow containers to communicate outside their pods. However, the virtual bridge does not provide inter-host communication. To allow for inter-host communication, routing functionality is needed. This could be done manually on each host, but this does not scale [12]. Therefore, inter-host communication in Kubernetes is often provided through a Container Networking Interface (CNI). There are currently 25 different CNIs in Kubernetes, each with a different focus. Nonetheless, each CNI needs to comply to the Kubernetes networking rules [14], which are:

1. Pods on the same node must be able to communicate with other pods without the use of Network Address Translation (NAT).
2. All agents (e.g. system daemons, kubelet) running on a particular node can communicate with all the pods running on the same node.
3. Pods that use the host network must be able to contact all other pods on all other nodes without using NAT.

From the 25 CNIs 11 comply to those rules by using a NVO. The NVO is a virtual network that is built on top of the physical infrastructure (i.e. underlay) that creates tunnels between hosts by using encapsulation. However, the use of encapsulation will increase the packet size and due to the default Maximum Transmission Unit (MTU) size, the total number of frames that need to be transferred will increase. This leads to overhead in the network [15]. To overcome this the MTU size should be increased [12].
3.2 Calico

Calico is a CNI that offers multi-tenancy with the use of network policies. Calico runs a calico agent on each node in a Kubernetes cluster, which consists of the BIRD Internet Routing Daemon (BIRD) and Felix agent. BIRD is a BGP routing daemon that is used to propagate routes between nodes. The Felix agent can write to the kernel’s routing table and manipulate the iptables of the node. When a pod is created, the Felix agent assigns an IP address to the pod and writes this IP address to the kernel’s routing table. Then the BIRD agent advertises this new pod to other BIRD agents in the cluster. However, in larger deployments iBGP advertisements can become a limiting factor since a full mesh is used by default. Meaning the number of iBGP connections increase with $N * (N - 1) / 2$ for each added peer. In order to reduce this connection increase the BIRD BGP daemon can run as a BGP Route Reflector on some of the BIRD agents [12] [16].

In order for Felix and BIRD to function properly, it relies on Etcd. Etcd is a distributed Key-Value (KV) store that is used to store and share information between the different components that Calico needs to build a network [16].

On top of BGP Calico can provide a NVO. Calico does this by creating a tunnel between hosts and makes use of IP-in-IP (IPIP), as shown in Figure 2.

IPIP encapsulates the IP address of the container as the inner header and the IP address of the node as the outer header. Calico uses BGP to the host, which allows for L3 multi-tenancy. Nonetheless, Calico needs network policies to provide tenant isolation. Calico does so through the Kubernetes network policy API that makes use of the standard Linux network filtering technology iptables [12]. The disadvantage of Calico is that iptables does not scale well [4]. Another disadvantage is that IPIP is not a frequently used payload, so the underlying hardware needs to support it. The advantage of IPIP is that it only adds a 20 bytes IP header [12].

Figure 2: Calico multi-tenancy [12]
3.3 Cilium

Cilium is a CNI that provides multi-tenancy with the use of network policies. Cilium addresses the network policy management that has become a challenge in orchestrated container environments, where containers can have short lifetimes. To address these shortcomings Cilium uses extended Berkeley Packet Filter (eBPF) instead of iptables [4]. The eBPF is a Linux kernel bytecode interpreter created to filter network packets in kernel space [4]. Developers can write a program for eBPF, load it into memory, and then run it when certain events happen. Thus, with eBPF one can filter packets in a more granular and fine-grained manner [4].

Cilium also focuses on labels as identity of containers, rather than IP addresses that can change frequently. Labels are KV pairs that are attached to objects such as pods or containers and are intended for users to identify attributes. Cilium keeps track of these labels and their identity by mapping them in a KV store. These labels allow Cilium to isolate pods by limiting access to certain pods based on their identity and apply L7 policy enforcement on both ingress and egress traffic [12].

Cilium uses VXLAN as a NVO to allow for L2 multi-tenancy. An example of L2 multi-tenancy with VXLAN as done by Cilium is shown in Figure 3.

![Figure 3: Cilium multi-tenancy](image)

In L2 multi-tenancy a VLAN is connected to a Virtual Network Identifier (VNI). In this example Tenant1 has VM1 on Server1 and VM3 on Server2, which both are in VLAN 10 that is connected to VNI 1000. Tenant2 has VM2 on Server1 and VM4 on Server2, which both are in VLAN 20 and is connected to VNI 2000. The VNIs span across the Virtual Tunnel End Points (VTEPs) to provide the overlay [17]. Note that the tenants share the same bridge, meaning L2 traffic can be sniffed. Cilium uses the earlier mentioned labels for pod isolation to prevent this.
The disadvantage of VXLAN as overlay is that it adds 50-54 bytes of overhead to the frame (Figure 4). This overhead needs to be considered with the MTU size (standard 1500 bytes), since it can cause fragmentation. Fragmentation puts extra load on the CPU and should therefore be avoided [12].

![VXLAN Frame Format](image)

Figure 4: VXLAN Frame Format [18]

### 3.4 EVPN in BGP

EVPN is added as an extended community to MultiProtocol-BGP (MP-BGP) and was introduced as a new BGP Network Layer Reachability Information (NLRI), which is the EVPN NLRI as specified in RFC 7432 [19]. The EVPN NLRI makes use of the MP-BGP extensions Address Family Identifier (AFI) and Subsequent Address Family Identifier (SAFI) that have been defined for EVPN. For EVPN the AFI is 25 (i.e. Layer 2 VPN) and the SAFI is 70 (i.e. EVPN) [19]. In the EVPN NLRI a Route Type (RT) can be specified. RFC 7432 describes the first four RTs and a draft of the IETF describes the later added RT-5 in version 4 of the draft [20]. The five RTs are depicted in Table 1.

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RT-1</td>
<td>Ethernet Auto-discovery Route</td>
</tr>
<tr>
<td>RT-2</td>
<td>MAC/IP Advertisement Route</td>
</tr>
<tr>
<td>RT-3</td>
<td>Inclusive Multicast Ethernet Tag Route</td>
</tr>
<tr>
<td>RT-4</td>
<td>Ethernet Segment Route</td>
</tr>
<tr>
<td>RT-5</td>
<td>IP Prefix Route</td>
</tr>
</tbody>
</table>

Table 1: EVPN Route Types [19], [20]
There are two RTs that are used throughout this research, which are RT-2 and RT-5. The RT-2 advertisement route, as shown in Table 2, can be used to advertise MAC/IP bindings that are learnt locally [21]. In a RT-2 advertisement, the EVPN Instance (EVI) is used to connect multiple nodes over an underlay, and the Route Distinguisher (RD) is used to specify the EVI the update is meant for. The Ethernet Segment Identifier (ESI) field is used in multihoming situations, where the VTEP informs the other EVIs there are multiple paths to reach the advertised MAC/IP address and the MultiProtocol Label Switching (MPLS) label fields are used for the L2VNI and L3VNI [22]. Whereas the L2VNI is used for bridging between similar VNIs and the L3VNI is used for routing between different VNIs [21].

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RD</td>
<td>RD of an EVI</td>
</tr>
<tr>
<td>ESI</td>
<td>ESI of a connection to a peer</td>
</tr>
<tr>
<td>Ethernet Tag ID</td>
<td>VLAN ID</td>
</tr>
<tr>
<td>MAC Address</td>
<td>Length of host MAC</td>
</tr>
<tr>
<td>MAC</td>
<td>Host MAC</td>
</tr>
<tr>
<td>IP Address</td>
<td>Mask length of host IP</td>
</tr>
<tr>
<td>IP</td>
<td>Host IP</td>
</tr>
<tr>
<td>MPLS Label1</td>
<td>Layer 2 VNI</td>
</tr>
<tr>
<td>MPLS Label2</td>
<td>Layer 3 VNI</td>
</tr>
</tbody>
</table>

Table 2: EVPN RT-2 [19]

The RT-5 or IP Prefix Route advertisement, as shown in Table 3, is used to advertise prefix routes in order to aggregate routes (e.g. /24 routes instead of /32 host routes). The RT-5 also consist of a RD, ESI, Ethernet Tag and have the same means as in the RT-2. Furthermore it consists of an IP Prefix length and the IP prefix to be advertised. It also contains the gateway IP address, which is only needed if there is no L3VNI, and therefore is often set to zero. The MPLS label is the L3VNI used.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RD</td>
<td>RD of an EVPN Instance (EVI)</td>
</tr>
<tr>
<td>ESI</td>
<td>ESI of a connection to a peer</td>
</tr>
<tr>
<td>Ethernet Tag ID</td>
<td>VLAN ID</td>
</tr>
<tr>
<td>IP Prefix Length</td>
<td>Mask length of host IP</td>
</tr>
<tr>
<td>IP Prefix</td>
<td>Host IP</td>
</tr>
<tr>
<td>Gateway IP address</td>
<td>Underlay VTEP IP</td>
</tr>
<tr>
<td>MPLS Label</td>
<td>L3VNI label</td>
</tr>
</tbody>
</table>

Table 3: EVPN RT-5 [20]
4 Methodology

Section 2.3 from the related work describes that inter-host communication is a difficult task in dynamic container networks and section 3.1 of the background describes that there exist 25 CNIs to target this task, each with a different focus. Since Makowski and Grosso [10] specify that multi-tenancy is a major feature for modern NVOs, our focus was to achieve multi-tenancy in a Kubernetes orchestrated container environment with the use of EVPN as an alternative CNI.

4.1 Requirements for multi-tenancy

In order to achieve multi-tenancy with the use of EVPN, the requirements that tenants have for multi-tenancy need to be specified. RFC 7364 [2] describes that the main expectations tenants have in a multi-tenancy data center is that a certain level of security and privacy is met in order to separate their resources from other tenants. An example of those expectations is traffic isolation and address space isolation, as mentioned earlier in section 2.1 of the related work, whereas the traffic is isolated between tenants without restricting address allocation.

Other requirements are dynamic provisioning of network resources, MAC mobility, decoupling of logical and physical configuration, the scalability of the forwarding table sizes, and optimal forwarding [2]. Dynamic provisioning of network resources, such as routing, is required in order to allow for quick changes in tenant systems and services. MAC mobility and the decoupling of logical and physical configuration is required for the movement of logical systems between physical systems in order to achieve high utilization and allow for live migrations. The scalability of the forwarding table is also an important requirement, since L2 networks in the data center can become huge when they stretch over multiple L3 networks. Last but not least, optimal forwarding requires the handling of east-west traffic within the data center, but also north-south traffic from the clients on the internet to the data center. Therefore, the best paths need to be chosen and networking resources must be efficiently used to improve the utilization of network resources [2].

4.2 Approach for EVPN as multi-tenancy solution

From these multi-tenancy requirements, we created an architecture. From the architecture, the design choices and their advantages were described, in order to answer our first subquestion "What are the advantages of EVPN as a solution to implement multi-tenancy in a Kubernetes orchestrated container environment?". Then, the integration of EVPN in Kubernetes was tested on a virtual environment as a PoC. The virtual environment that was used is shown in Figure 5. In this figure a full mesh between the spine and leaf switches is used. Note that the servers communicate directly over L3, whereas in the earlier VXLAN as provided by Cilium, the servers would use L2. This PoC is used to answer the second subquestion "How can the recent improvements in FRR allow for EVPN to be integrated into the Kubernetes container orchestration platform?".
4.3 Experiments

To test whether our architecture was integrated as intended we performed two experiments. The first experiment is used to test the connectivity between containers of tenants on different nodes through the overlay. From this experiment we could see the traffic flow with EVPN. The second experiment is used to test multi-tenancy, where multiple tenants share the node and therefore require isolation between tenants while maintaining connectivity between their containers on the different nodes. Both experiments are visually represented in section 6.2 and section 6.3.

4.4 Experiment 1: Connectivity and traffic flow

The first experiment consisted of two tenants that both have containers spread over four nodes (i.e. one container on each node). However, the nodes do not host both containers from the tenants. Meaning a node is either of Tenant1 or Tenant2. The tenants must have connectivity between their containers. Whereas the tenants do not share the node, but share the network.

4.5 Experiment 2: Multi-tenancy and isolation

The second experiment consisted of three tenants that have containers spread over three nodes (i.e. three tenants per node, nine containers in total), and therefore share the network and the node. The tenants must have connectivity to their containers, but the connectivity must also be separated from other tenants to provide isolation. The tenants must also not be able to intercept traffic of other tenants.
5 Architecture of EVPN for multi-tenancy

In order to allow for EVPN as a multi-tenancy solution, based on the multi-tenancy requirements, the design depicted in Figure 6 was created. The design choices to meet those requirements and the advantages are specified below.

![Figure 6: EVPN architecture for multi-tenancy](image)

5.1 Traffic isolation and address space isolation

To comply with the main requirements, traffic isolation and address space isolation, we made use of an IP-VRF (i.e. L3VNI) and a virtual bridge for each tenant. The IP-VRF provides L3 traffic isolation by creating a separate virtual routing table per tenant. This VRF table only consists of the routes to the containers belonging to the tenant, such that the tenant can only reach its own containers. Beside the L3 isolation, a virtual bridge for each tenant was used to prevent a malicious tenant from sniffing L2 traffic and potentially advertising the anycast Switched Virtual Interface (SVI) MAC address. The anycast SVI acts as the default gateway, which might be spoofed in order for a malicious tenant to become a Man-In-The-Middle (MITM). The use of a separate VRF table per tenant also provides address space isolation. Since the VRF only contains the tenant specific IP addresses, a different tenant can reuse the same IP addressing. However, Kubernetes requires every pod to have a unique IP address. Therefore, there is no use for address reuse.
5.2 Dynamic provisioning

The dynamic provisioning of network resources is provided through the use of FRR and EVPN RT-2. Once a container IP address is imported in the corresponding VRF table, which will be described in detail in section 6.1, FRR advertises the MAC/IP binding through an EVPN RT-2 to the same EVPN VRF instance on different nodes. Tenants on different nodes can then learn the MAC/IP binding of the container and add it to its corresponding tenant VRF table. In our design we decided to advertise containers as a host route with EVPN RT-2, due to the focus of multi-tenancy for smaller tenants in the data center. Larger tenants are assumed to prefer dedicated nodes rather than shared nodes with multi-tenancy. However, when larger tenants decide to become part of a multi-tenancy data center environment, the use of RT-5 Prefix route advertisements would be more efficient. This would reduce the size of the VRF table of that tenant, by aggregating routes. However, the effect of aggregating routes depends on the number of containers per node.

In addition to the dynamic provisioning of network resources, fast convergence of the learned routes is also an important factor. EVPN has the advantage that it can quickly reroute traffic to circumvent failed links, failed network devices or when a container is migrated to a different node [23]. Since the forwarding of the data plane is dictated by the EVPN control plane, traffic destined for the endpoint can be quickly redirected. However, this could also create issues when a MAC addressing is flapping between VTEPs, e.g. due to network instability or duplicate MAC addresses. To prevent constant updating, EVPN has a counter (default 180 seconds). If the same MAC has moved more than 5 times before the counter has expired, EVPN will stop sending updates for that particular MAC and creates a syslog message. This prevents the network and the other VTEPs from also becoming unstable [21].

5.3 MAC mobility

MAC mobility and the decoupling of logical and physical configuration is also provided through the use of EVPN RT-2. When the node (i.e. VTEP) learns about a MAC address it advertises this directly, rather than having to relearn the MAC/IP address and the corresponding VTEP through the data plane. Furthermore, an anycast SVI is used to prevent the container from relearning the IP address of the default gateway. The SVI does this by sharing a virtual IP address that is used by the tenant across different nodes.

Beside MAC mobility, control plane MAC learning also allows for a higher network efficiency. As mentioned in section 3.4, control plane MAC learning is done by advertising locally learned MAC addresses to the other VTEPs, as soon as a MAC address is learned locally. This advertisement message is sent as an EVPN RT-2 to all remote VTEPs and contains the MAC/IP binding, VNI and corresponding VTEP. Control plane MAC learning is less resource consuming than having to learn the MAC addresses through the data plane, which makes use of...
the flood-and-learn approach. A flood-and-learn approach such as ARP flooding generates a lot of unnecessary traffic on the network, since every VTEP is learning for themselves. The use of EVPN RT-2 can eliminate this flooding approach and therefore improve the network efficiency [5].

The disadvantage of control plane MAC learning is that remote VTEPs that have no interest in the RT-2 advertisement still get the advertisement and have to discard it. VTEPs can have no interest in the RT-2 advertisement if it does not host the VNI that is advertised. As a result, unnecessary traffic is sent to the VTEPs that are not interested in the advertisement. The reason that all VTEPs receive these advertisements is that otherwise the spine switches need to keep track of the VTEPs that are interested in these advertisements [5]. Beside the unnecessary computation to provide this tracking, it also scales better when the spines do not have to track VNIs. The default implementation of EVPN is not to keep track of the VTEP VNIs [5]. We kept it that way.

5.4 The scalability of the forwarding table sizes

The scalability of the forwarding table sizes is provided by splitting the forwarding table into smaller forwarding tables per tenant with the use of VRF. Furthermore, L3 networks are tunneled over a L3 network, instead of L2 networks tunneled over a L3 network. Meaning the L2 networks are not stretched and remain small. Kubernetes can manage a total of 300,000 containers spread over a maximum of 5,000 nodes [12]. Assuming a larger tenant has 5,000 containers, e.g. two container on each node, than the forwarding table of that tenant contains 4,998 routes (not counting the containers on the node itself). However, the two host routes for the containers can be aggregated to RT-5 Prefix routes for each node, reducing the number of routes to 2,499. Furthermore, the other tenants forwarding tables can remain small. Whereas with other CNIs that use network policies, all the nodes share the forwarding table, meaning the forwarding table can consist of thousands of routes and thousands of network policies are required to prevent the communication between tenants.

5.5 Optimal forwarding

The requirement for optimal forwarding for east-west traffic is provided through the use of Equal-Cost Multi-Path routing (ECMP) in EVPN. ECMP is specified as a requirement of EVPN in section 4.2 of RFC 7209 [23]. ECMP allows for the use of same cost paths to a destination and can provide for load balancing between those equal cost paths [23]. Especially in multi-homed scenarios as in our virtual test environment, were the node is connected to two leaf switches, the leverage of ECMP for load balancing purposes can increase the efficient use of network resources.
The north-south traffic can be achieved by mapping the tenant VRFs to a VRF-Lite or global routing table on the border leaf switches (as shown in Figure 7) that are connected to edge routers and provide external connectivity. These border leaf switches can use route leaking to "leak" routes from the tenant VRFs to the global routing table of the border leafs in order to advertise those routes to the outside world. Routing leaking becomes possible with the use of EVPN, since every node and switch are eBGP peers [24] (as shown in Figure 6).

![Figure 7: North-south traffic through the Border Leafs and Border Edge Routers.](image-url)

In addition to ECMP, EVPN also allows for multihoming. Data planes such as VXLAN do not provide for multihoming. As mentioned earlier in section 3.4, EVPN allows for Ethernet Segment (ES) multihoming. An ES is a device or network that is connected to one or more links [9]. All-active multihoming allows for flow-based load balancing in order to fully utilize all the links. However, multihoming can also cause loops. To avoid loops, EVPN makes use of split-horizon, such that packets leaving one multihomed link are not sent back to one of the other multihomed links. A MPLS label is used by EVPN to support split-horizon filtering. This label contains the ESI. When a NVE is attempting to forward a multi-destination frame, it checks the label and if the ESI corresponds to the ESI of the outgoing interface, it will not be forwarded to that interface. Furthermore, EVPN selects a Designated Forwarder (DF) so that only one link is allowed to send Broadcast, unknown Unicast and Multicast (BUM) traffic. This prevents a multihomed device or network from delivering duplicate BUM traffic into the network. The election of a DF can be done for each VNI, allowing for load sharing [9].
6 Results

From the created architecture and the advantages of EVPN as a multi-tenancy solution, the next step was to integrate EVPN as CNI in Kubernetes. In this chapter we describe the integration of EVPN in Kubernetes and present the results of the conducted experiments.

6.1 Integration of EVPN in Kubernetes

For the integration of EVPN in Kubernetes as CNI the IP addresses of the containers had to be added to the VRF table of the corresponding tenant. Then these IP addresses need to be advertised to other tenant VRF tables on different nodes. In order to get the container IP addresses of a tenant and advertise them to the VRF tables of that tenant on different nodes, we used the process as shown in Figure 8.

![Integration of EVPN in Kubernetes flowchart](image)

Figure 8: Integration of EVPN in Kubernetes flowchart

When the Kubernetes Master schedules a container to the Kubelet agent on the node, the Kubelet agent creates the container with Docker. The creation or deletion of a container triggers an event, which the Cumulus Routing on the Host Docker Advertisement daemon (CRoHDAd) listens to. The original CRoHDAd checks for the IP address of the container and adds or deletes the host route to a default routing table. We adjusted the CRoHDAd (see Github [25]), such that based on the labels provided in the Kubernetes deployment of the container, CRoHDAd adds or deletes the host route to the corresponding VRF table. When the host routes are added to the tenant VRF, they get imported to FRR and advertised by BGP-EVPN.
6.2 Experiment 1: Connectivity and traffic flow

For the connectivity and traffic flow experiment we used Server2 and Server6 as Tenant1 and Server4 and Server8 as Tenant2, as shown in Figure 9. Note that the last octet of the IP address of Container1 on Server2 is higher than the other containers. This is because Server2 was used a lot for the container creation to test the adjusted CRoHDAd, since Kubernetes uses flexible IP address allocation. Kube-dns can be used to provide a mapping between a container IP address and the naming of the container.

Figure 9: Experiment 1 - Inter-host connectivity between containers of the same tenant as if they were on the same host.

From the created architecture and the adjusted CRoHDAd, it was possible to create connectivity between Container1 and Container3 of Tenant1 through the use of VRF. Container2 and Container4 of Tenant2 also had connectivity through a different VRF table. Since each tenant had its own VRF table there was no cross connectivity between the containers of the different tenants (e.g. Container1 of Tenant1 to Container2 of Tenant2).
Beside the connectivity between the containers of each tenant, we also looked into the traffic flow between Container4 on Server8 and Container2 on Server4 of Tenant2. The results of the traffic flow are shown in Figure 10.

Figure 10: Experiment 1 - Traffic flow from Container4 on Server8 of Tenant2 towards Container2 on Server4 of Tenant2.

From this Traffic flow we can see that only the Source MAC (SMAC) and Destination MAC (DMAC) change throughout the traffic flow. Whereas the SMAC and DMAC change in the ethernet frame, until the traffic is encapsulated through the underlay with VXLAN. When the traffic is encapsulated with a VXLAN header from the Tenant2 SVI on Server8 (i.e. SVI T2S8) towards the Tenant2 SVI on Server4 (i.e. SVI T2S4), the SMAC and DMAC of the VXLAN header change, while the ethernet frame SMAC and DMAC remain untouched. When the Tenant2 SVI on Server4 obtains the encapsulated ethernet frame, it decapsulates it and forwards the ethernet frame to the bridge on Server4. In this traffic flow Leaf3, Spine2 and Leaf1 were chosen for the path through the underlay. However, other paths might also be chosen. For the SVIs on Server8 and Server4 it appears if they are directly connected. Also note that the SMAC and DMAC at Tenant2 SVI on Server8 change in both the ethernet frame as in the VXLAN header. This is because it first changes the ethernet frame and then encapsulate it with VXLAN.
6.3 Experiment 2: Multi-tenancy and isolation

In experiment 2 the multi-tenancy and isolation were tested. For this experiment Server3, Server5, and Server7 were used. As shown in Figure 11 each server was shared by three tenants, that had their own bridge, SVI and VRF.

From the conducted experiment it was possible to create a VRF for each tenant on the same node that contained only the routes from that tenant (i.e VRF Tenant1 contains the routes of Container1, Container4 and Container7). The global routing table only contained the underlay IP addresses to reach the other nodes, and inter-host connectivity was provided between containers of the same tenant. Furthermore, there was no connectivity possible between containers of different nodes, and no traffic could be intercepted by containers on different bridges that belonged to a different tenant. Meaning that there was inter-host connectivity between the tenant its containers, while complete network separation was provided between different tenants.
7 Discussion

In line with the hypothesis, the conducted experiments showed that inter-host connectivity and multi-tenancy isolation could be provided without the use of network policies. However, the experiments were conducted in a small test environment without real traffic. Therefore, EVPN with VRFs and virtual bridges to provide tenant isolation in Kubernetes should be thoroughly tested in a larger environment. Since it is unknown what the effect is of a large number of VRFs and virtual bridges on a node.

In addition, EVPN was only tested in a Kubernetes orchestrated containerized environment. Meaning the results may vary in different orchestrated containerized environment, such as Docker Swarm.

Furthermore, the results from the conducted experiments are based on the design choices made in the created architecture. However, multiple architectures might be possible to provide traffic isolation without the use for network policies in EVPN.

8 Conclusion

In this research, the inter-host connectivity and multi-tenancy of EVPN without the need for network policies was investigated using a virtual test environment. The focus of this research was based on network separation between tenants on the same node, such that smaller tenants could share a node with different tenants to provide higher resource utilization and thus reduce costs.

In order to provide multi-tenancy for EVPN in Kubernetes, an architecture was created. This architecture focused on isolation through the use of VRFs and a separated virtual bridge per tenant. Beside traffic isolation and address space isolation, this architecture also focused on the requirements dynamic provisioning, MAC mobility, scalability of the forwarding table, and optimal forwarding. Furthermore, EVPN as CNI in Kubernetes has the advantage that it could provide for more network efficiency with the use of control plane MAC learning and multihoming. Last but not least, EVPN also allows for fast convergence by rerouting traffic through the dictating of the control plane.

Based on the results, it was found that an adjusted version of CRoHDAd could be used to inject routes into a tenant VRF table. The injected routes could then be advertised by FRR to VRF tables of that tenant on different nodes. The results also showed that inter-host connectivity could be provided between tenants on different nodes, as if the containers were hosted on the same node. Furthermore, it was shown that multi-tenancy could be provided by EVPN without the need for network policies, but with the use of VRFs and separated virtual bridges to provide isolation.
9 Future work

In this investigation, we did not test the performance of the created architecture in comparison with the current CNIs that provide multi-tenancy such as Calico or Cilium. Like Cilium, our architecture made use of VXLAN as NVO. VXLAN creates an overhead of 50-54 bytes to the frame, which has an impact on the network performance and needs to be considered with the MTU size. Otherwise fragmentation can occur, that puts extra load on the CPU. However, in contrast with Calico and Cilium, EVPN routing decisions can be made without having to check network policies, which might lead to a performance increase. In order to broaden this research, a performance comparison of EVPN with different multi-tenancy CNIs that also provide multi-tenancy could be conducted.

In addition, an EVPN CNI plugin could be written, such that EVPN can directly be used as a CNI, instead of using CRoHDAd to inject the host routes to the tenant VRF table in order to advertise the routes of the tenant VRF across the nodes.

10 Acknowledgements

We would like to show our gratitude to Attila de Groot, system engineer at Cumulus Networks, for providing us the opportunity to explore EVPN and Kubernetes, but also for the virtual test environment to conduct our PoC and the support during the writing of this paper.
References


