IPVLAN – The beginning

Mahesh Bandewar, Eric Dumazet

Google
Mountain View, CA, USA
maheshb@google.com edumazet@google.com

Abstract

The commonly used method to connect namespaces to the outside world without going through the forwarding set up on the host used to be the macvlan. This setup is simple and efficient except when the next-hop devices apply policies barring host to act like a layer2-switching device. This is especially problematic where the connected next-hop, e.g. switch is expecting frames from a specific mac for a given port. In a situation like this the macvlan setup does not work. The host will either have to fall-back to non-efficient forwarding methods or something else. IPvlan was designed to address this specific need along with few other mentioned in next few sections. This paper attempts to describe these use cases and highlights differences with macvlan devices and briefly talk about future enhancements planned.

Keywords

IPvlan, Macvlan, Layer2 Switch, ARP, Broadcast, Multicast, IPv4, IPv6

Introduction

This paper attempts to describe the motivation behind developing the IPvlan virtual device driver. It highlights the challenges faced, describes various use cases and briefly touches on the future enhancements planned.

The need

- Macvlan setup: In many modern setup the fabric that connects hosts together have security policy which does not allow hosts to emit frames apart from the mac-address (L2) assigned to it. This ensures that the switch port the host is connected to always receives packets coming from the legitimate host and there is no L2 level spoofing involved. Even though this prohibits the L2 level spoofing issue, it poses host a problem for using macvlan device type of solution to use when multi namespace setup is required. The basic requirement for the macvlan based setup is to have every virtual device use one of the mac-addresses reserved by the NIC or possibly randomly generate one for each of the virtual devices. Assigning each such virtual device to namespace and then making the underlying physical device to act like a switch to mux and demux packets is a common setup. But this also means each ns is not going to TX L2 frames with the random L2 address that is assigned to each of these virtual devices. This would make the connected switch drop the packets thinking these are spoofed and hence disrupting the connectivity.

- Promiscuous mode: One other issue of using multiple L2 addresses on the NIC is that each of such NIC can handle certain no of unicast and multicast L2 addresses. Once the limit is reached, the NIC will be forced to operate in promiscuous mode. This could have deterring performance impact on host if the connected switch hardware forwards more than necessary traffic its way (especially multicast traffic).

- Forwarding alternative: In a restrictive environment like above where switches don't allow frames other than the NICs' L2 address, the other alternative that can be used does not involve macvlans. In this setup device pairs like virtual-Ethernets can be used and the default namespace can be put into the forwarding mode so that the packets coming out and are needing to send to these namespaces be forwarded from and to within the default namespace effectively making the default namespace a forwarding instance. Though this solution works on a functional basis, the performance / packet rate expected from this setup is is much lesser since every packet that is going in or out is processed 2+ times on the network stack (2x Ingress + Egress or 2x Egress + Ingress). This is a huge cost to pay for.

- NAT: Another possible alternative is to use some sort of NAT-ing on the host so that connected switch always sees the same L2 from the host but this approach also suffers from the same performance pitfall as mentioned in the forwarding approach.

Development Challenges and Choices

Integrate changes into macvlan

Initially when started working on the solution to solve the above problem, making changes into the existing solution (i.e. macvlan) was the plausible choice. The basic idea in macvlan is to use L2 to find the right virtual device in the hash-table. The same logic needed to be extended to use L3 for these “new mode(s)” to work. When it comes to IP (L3), a device could have multiple IPs at the same time. This was proving difficult to accommodate while maintaining the current driver structure of macvlan without compromising it in any way.

Broadcast / Multicast tweaks

One other problem that macvlan faces is handling of broadcast / multicast traffic. It does good job now with the implementation of multicast filters and deferring the RX traffic to work-queues. However it has to handle the traffic which is deterrent to the performance of the device. Since ipvlan has L3 knowledge and can be used to handle broadcast / multicast in an efficient manner. If IPv4 is not enabled, processing of broadcast can be completely turned off and this eliminates large chunk of overhead on the driver while multicast filters can handle the multicast traffic with some optimization.

Communication with the host (default-ns)

Similar to the macvlan devices, the traffic to and from the master device cannot be sent to and from slaves. In the macvlan setup the problem can be solved if the host is connected to a switch that allows hair-pin mode. However, not many switches support this mode.

Packets from the slave interfaces will reach the master interface (mostly in the default-ns) but the replies can’t reach the slave interfaces since the bridge on the master is transparent in the TX mode. This causes TX replies from the master to leave the host and will be lost resulting in broken connectivity.

This can be worked-around by assigning one of the virtual devices to the host and eliminating the configuration on the master interface, as shown in Fig1.

If the above is not possible in some setups, there is another way this can be achieved is by injecting specialty routes in both name-spaces (masters’ as well as slaves’ name-space). This is too hacky as well as not really scalable in the way it is currently implemented. I’m not sure if there is a cleaner way of implementing this.

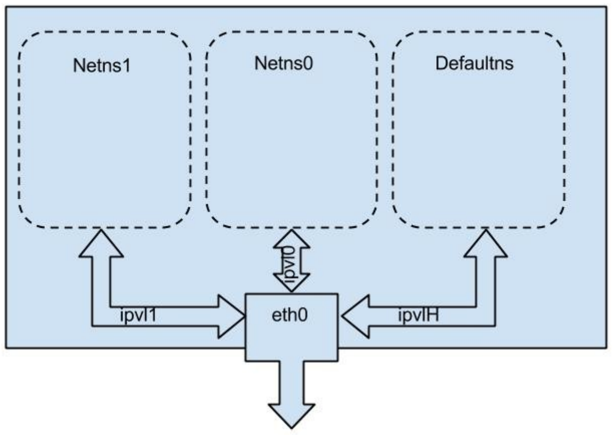

Fig1: Host setup making all traffic go though the bridge on the master device

Typical Use Cases

L2 Mode

This mode is equivalent to the bridge mode in macvlan. The usual deployment case is the slaves are assigned to the namespaces and packets are completely processed on stack instance attached to the namespace. This is true for both ingress as well as egress. This is setup is more suited for the independent as well as configuration-trusted namespaces. This setup is similar to connecting multiple hosts to a switch where each host can function independently and relies on the switch to provide connectivity.

L3 Mode

This is more restrictive mode. The usual deployment case is same as above except that most of the networking related decisions are taken in the namespace where master is attached. This includes any encapsulation / routing policy that is specific to the address assigned to a namespace. Currently the address manipulation on the virtual device from it’s attached namespace is allowed, but even that would be restricted with future enhancement.

Future Enhancements

Deferring multicast / broadcast traffic

In the current IPvlan implementation; there is no concept of deferred work and all traffic is processed as and when it appears. With multicast / broadcast work-queue implementation, the unicast packets will get preferential treatment boosting throughput and improving latencies.

ARP filter

Broadcast processing is a big burden in IPv4 enabled environment especially when there are large no of virtual devices / slaves present on a master device. Every packet has to be duplicated n-times when there are n slaves. This burden can be lightened with tricks like ARP-filter. The packet can be probed and it can be dropped directly when none of the slaves are interested in it and can be forwarded to only the one who will process it. This reduces the need to make n-copies of the packets for every ARP broadcast.

Macvtap type of enhancements

Macvtap is a character device that largely follows tun/tap ioctls and can be used by Kvm/Qemu or any other other hypervisor that supports tun/tap devices. A similar device can be envisioned to work with IPvlan. The mac-address will be inherited by the slave device while the IPvlan can learn about L3 on slave and use that to switch packets to and from the device.

Xmit offload to hardware

If the underlying (master) device supports L2 hardware acceleration, then the TX path can be offloaded to the hardware which can be more efficient than the standard TX path.

Special L3 mode

When master is configured to use as bridge while each of the name-spaces gets individual slave links. In a setup like this L2 mode is easier to configure but this is not possible in the L3 mode. All the traffic that is not unicast cannot be delivered to any link and hence this special mode is required. In this mode, one of the slave link will be nominated to receive this traffic and that slave link can be assigned to the name-space (default-ns) which processes all other traffic.

Configuration Lock

There are deployment situations where the name-spaces are not trusted and there is a need to lock-down configuration from salves' namespace. This can be achieved by simply allowing only the namespace where master belongs can make changes.

User-space utility support (CRIU / Docker)

Docker containers make effective use of namespaces and face the same connectivity performance issues when it comes to the network namespaces. CRIU tool is used for checkpoint / restore operations in container migrations setups. IPvlan benefits can be extended to these offering, but since IPvlan is relatively new, these tools do not include support yet.

Authors Biographies

Eric Dumazet, and Mahesh Bandewar both work as Software Engineers at Google.