Virtual Router (vRouter)

Jonathan Hart
Open Networking Lab

January 5, 2016

Introduction

In legacy CO environments, a Broadband Network Gateway (BNG) device aggregates subscriber connections and routes traffic to and from the core network. In the disaggregated CORD architecture we no longer have a BNG device as the gateway to the CO, and instead we implement the necessary functionality as a virtual router (vRouter) service. The vRouter service does not intended to implement all functionality that exists in traditional BNG devices, however it does implement the functionality necessary to provide internet access to CORD in a disaggregated fashion.

The purpose of the vRouter service is to be the gateway between the CORD POD and the upstream network and to provide internet access to the subscribers and services within CORD. Logically, it is the final service in the chain that a user's traffic traverses before exiting the CORD system. Physically, it is the interface between CORD and the provider's upstream network. The vRouter service provides Internet-as-a-service to other services within the CO. The vRouter is implemented as a network control application running on ONOS.

vRouter Requirements

- Perform L3 unicast routing to/from CO; participate in dynamic routing protocols
- Multicast signaling and forwarding
- Apply QoS policies
- Perform NAT functionality

AT&T Field Trial

As the vRouter is the interface between CORD and the upstream provider network, the exact protocols and functions that need to be supported by the vRouter service will vary according to the requirements of each CORD deployment. The current vRouter service focuses on a field trial of CORD with AT&T, which needs to support OSPF and iBGP communication to upstream routers. Other deployments may require a different set of routing protocols, or none at all depending on the provider's upstream network.

- Exchange routing information with upstream routers via OSPF and iBGP – learn routes from upstream, advertise subnets used within the CORD POD towards the upstream.
- Support PIM-SSM for multicast signalling
- Set DSCP bits appropriately for QoS.

Design

The design of the vRouter service is split into two main parts that are relatively independent from one another – control plane and data plane.

The vRouter has some dataplane devices which it is in control of, and from the external point of view those devices appear as though they are a single router, supporting a set of routing protocols.

Control Plane

The main functionality of the vRouter concerns speaking routing protocols with external routers. In order to avoid having to implement routing protocols within an ONOS application, we've elected to use an existing open source routing stack called Quagga. Quagga supports a wide range of routing protocols, which allows the vRouter to support these protocols without having to reimplement them. At this point the vRouter is considering environments with a relatively small number of routes, so there aren't the performance concerns that come with internet-scale.

Quagga will be configured to communicate with the upstream routers – in the field trial case this will be using OSPF and iBGP.

Quagga defines an interface called the FIB Push Interface (FPI) which enables it to push routes to an external entity. We use this interface to communicate routes from Quagga to ONOS. The ONOS vRouter app acts as the Forwarding Plane Manager (FPM), and is able to receive and decode routes from Quagga. The vRouter app is then able to use these routes to program the dataplane accordingly.

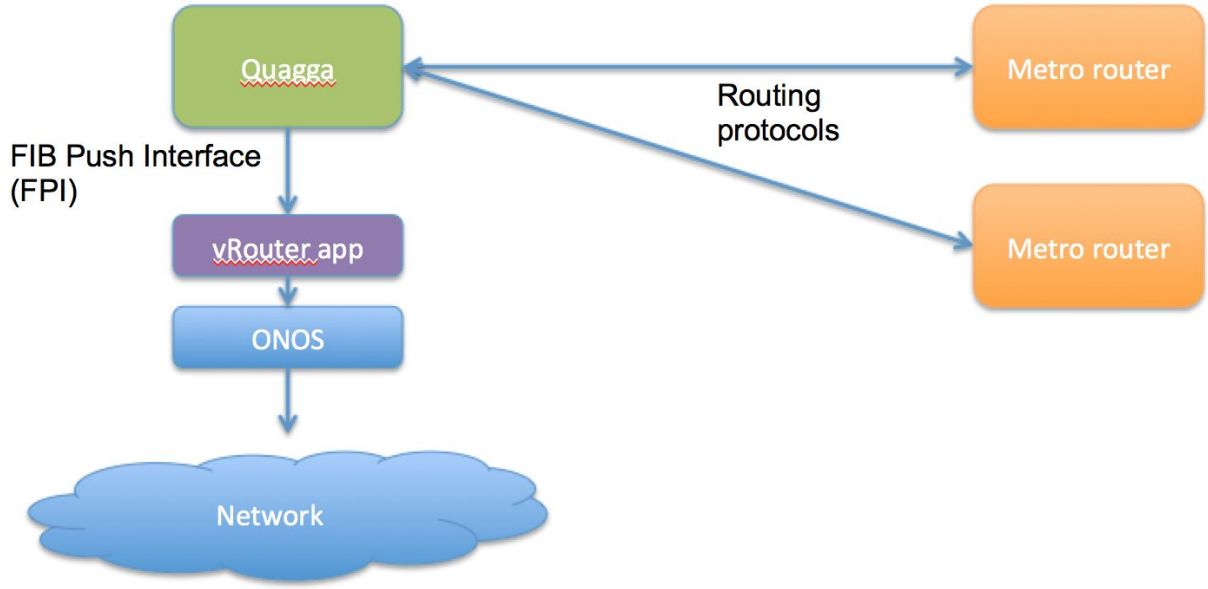

Figure 1: Control plane architecture of the vRouter

The design of the control plane is similar to the approach used in the ONOS SDN-IP application. The main difference here is that we need to support more than just BGP peering, because we need to support an IGP as well. SDN-IP uses an iBGP connection between Quagga and ONOS, whereas with vRouter we use the FPM interface.

Control Traffic

In order for Quagga to be able to communicate with upstream routers, routing control traffic has to flow between Quagga server and the external router. Before any routes are exchanged, the first task of the vRouter app is to program the dataplane to allow this traffic to flow. The Quagga server is connected to a port on the vRouter datapane, and incoming/outgoing routing packets are directed to/from that port. This bypasses the usual routing function of the vRouter because it concerns control plane traffic that is destined for the router itself.

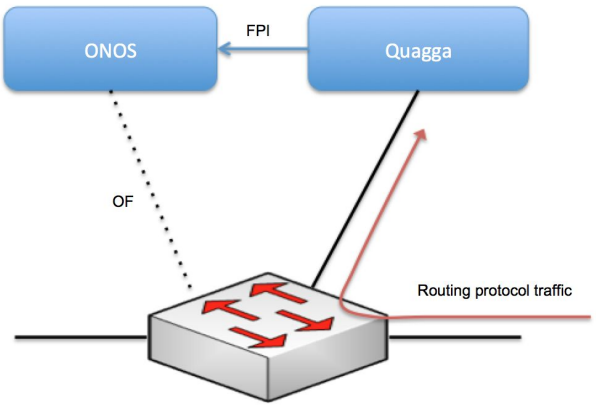

Figure 2: Routing control traffic is handled by redirecting to towards the Quagga instance connected on the dataplane

Multicast

The vRouter also needs to support PIM-SSM for multicast signalling to the upstream. PIM-SSM will not be handled by Quagga, rather there will be a dedicated ONOS app for managing PIM messages. The PIM app will send PIM join messages upstream when new multicast groups need to be requested. The PIM app will learn of the need for new multicast groups from other control applications within ONOS (ultimately this comes from IGMP snooping at the access devices). The will not be any protocol-based multicast signalling on the dataplane between the fabric and the vRouter - this all happens within ONOS.

Data Plane

We've chosen to use dedicated switches for dataplane of the vRouter. While it may technically be possible to implement the vRouter functionality in the fabric hardware switches, this would be a more complex proposition to explore. In light of this we've chosen to use dedicated switches for the vRouter in order to ensure that we are able to build the vRouter within the required time frame. We also need to ensure that the vRouter can support NAT and QoS, which are not supported by the current fabric switches. In the future it would be interesting to investigate implementing the vRouter dataplane in the fabric switches.

Figure 3: The vRouter dataplane elements sit between the fabric and the upstream metro routers.

Implementation

The implementation of the vRouter app in ONOS will leverage existing code from other ONOS routing applications.

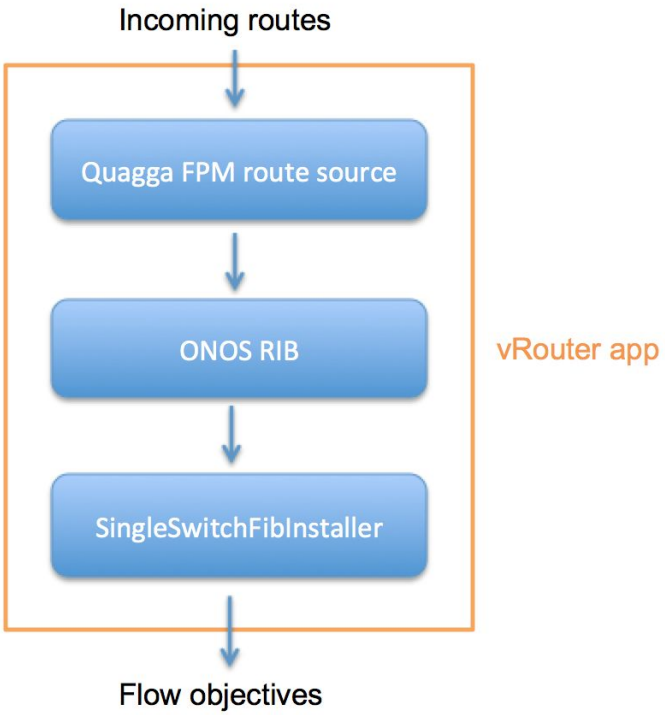

Figure 4: Components that make up the vRouter ONOS app

The architecture of the routing function is shown in Figure 4. Incoming routes from Quagga are received by a Forwarding Plane Manager (FPM) component. This component knows how to decode routes from the external Netlink protocol, and it pushes those routes into the ONOS RIB. The ONOS RIB works to resolve the MAC address of the next hop, and once it has this information it pushes a FIB update to FIB installer component.

Initially for the FIB installer we will reuse a SingleSwitchFibInstaller component which was developed for the ONOS BgpRouter application. This component is designed to install routes into a single switch, thereby turning that switch into an IP router. The component will generate FlowObjectives and submit them to ONOS.