Apollo: Anchor Stack 1.0. Overview

Document Version: 1.0. March 13, 2015.

Audience

This is an overview document intended to give a sense of the high level architecture and general direction of Anchor Stack development. The audience for this document includes all stakeholders in the SDN vertical ecosystem. This includes:

- Operators who will commit to include Anchor Stack (in whole or in part) in trials for production deployments.
- Switch vendors who will integrate their OpenFlow switches with Anchor Stack.
- Controller vendors who will integrate their controllers to run in Anchor Stack.
- Application developers who will create new applications to run on top of Anchor Stack.

Introduction

The Anchor Stack project seeks to develop and distribute an integrated, vertical collection of open-source SDN components which, together, provide the core to build out deployable SDN Solutions in the following target segments - Small, Medium, Large Enterprise, Data Center, Cloud Service Provider, Cloud Enterprise, Service Provider Core, Wireless Service Provider, Service Provider Edge, Verticals (Education, Healthcare...) etc... ..

For a more detailed architectural description and integration guide, please see the Anchor Stack Architecture and Integration document.

The Apollo release of Anchor Stack (addressed by this document) will focus on routing functionality with BGP peering as the primary application. As such, it is not exclusively tailored to Data Center (DC) or WAN deployments, but can be used in both scenarios with additional work. See the Operator Use Case Section below for more information.

Motivation and Goals

The current state of SDN technology suffers from two significant gaps that interfere with the development of a vibrant ecosystem. First, there is a large gap in the *integration* of the elements that are needed to build an SDN stack. While there may be multiple choices at some layers, there are missing pieces and poor or no integration.

Second, there is a gap in *interoperability*. This exists both at a product level, where existing products from different vendors have limited compatibility, and at a protocol level, where interfaces between the layers are either over or under specified. For example, the number of versions of OpenFlow currently defined, and the complexity of determining the particulars of one version, make it difficult to connect an arbitrary switch and controller. On the other hand, the interface for writing applications on top of a controller platform is mostly under specified, making it difficult to write a portable application.

The Anchor Stack project attempts to address these challenges, not by working in the specification space, but by instantiating and integrating a set of production quality components. Their successful integration will then allow alternative component instances (switches, controllers, applications) to be integrated into the stack. This integration will be driven by the exact top-down requirements needed to provide the functions and interfaces exercised by Anchor Stack applications. Thus Anchor Stack will initially be minimally generic with the expectation that future releases will provide abstractions as needed.

Most importantly we wish to work closely with network operators on deployable use-cases, so that they could download near production quality code from one location, and trial functioning software defined networks on real hardware. Thus, Anchor Stack will be the first *full open source SDN distribution* akin to a Linux distribution. We believe that with operator input, requirements and a deployment scenario in mind, Anchor Stack can be both be useful as distributed, and provide the basis for future extensions and alternative distributions whose focus is on requirements different from those of the original Anchor Stack.

Guiding Principles

We mark the following principles for the project.

1. The project is organized as a unified open source effort developing and maintaining a single **open source SDN distribution.**
2. Where possible, **adopt existing open source components** such as ONOS, ODL, OVS, SwitchLight and those from OCP.
3. Where necessary, create, adapt or commission **missing components** such as OF-Config or TTPs.
4. Be **inclusive** of multiple hardware vendors and control plane solutions.
5. Work hand-in-hand with operators on **deployable use-cases**, leading to trials on real hardware.
6. Work in close **collaboration** with different open-source SDN projects resulting in a broader solution. An important example is the development of common NB-API code.

The Anchor Stack project is not attempting to provide a complete, production quality, off the shelf solution to any specific deployment use case. Rather, it is focused on developing and making available production quality components that can be integrated into a variety of deployments.

Anchor Stack 1.0 Overview

Operator Use Case

Google's production network R&D team wishes to deploy a BGP peering router using SDN technologies in a production safe way. Such an SDN router would behave like any normal router by advertising routes and forwarding IP packets. It would peer with traditional (non-SDN) routers and forward real traffic in a deployment that spans multiple continents. Google's hope is that such a deployment would help operators worldwide understand SDN technologies better and gain confidence in them, by presenting the operators with an opportunity to interact with a networking component they are intimately familiar with - a peering router.

In addition, by focusing on an actual deployment scenario instead of a lab trial or demo prototype, this use case draws on requirements that would otherwise be glossed over or ignored. The intention is therefore to also help open-source SDN efforts move closer to production quality.

Figure 1 below shows the deployment scenario. The peering router delivered by this project would initially be deployed in two ASes - one in Australia (AARNet) and then other in the US (ESNet). The SDN routers would be connected by transcontinental L2 circuits. They would also peer with traditional routers within their AS. Over time, more ASes can be peered with the ones shown below.

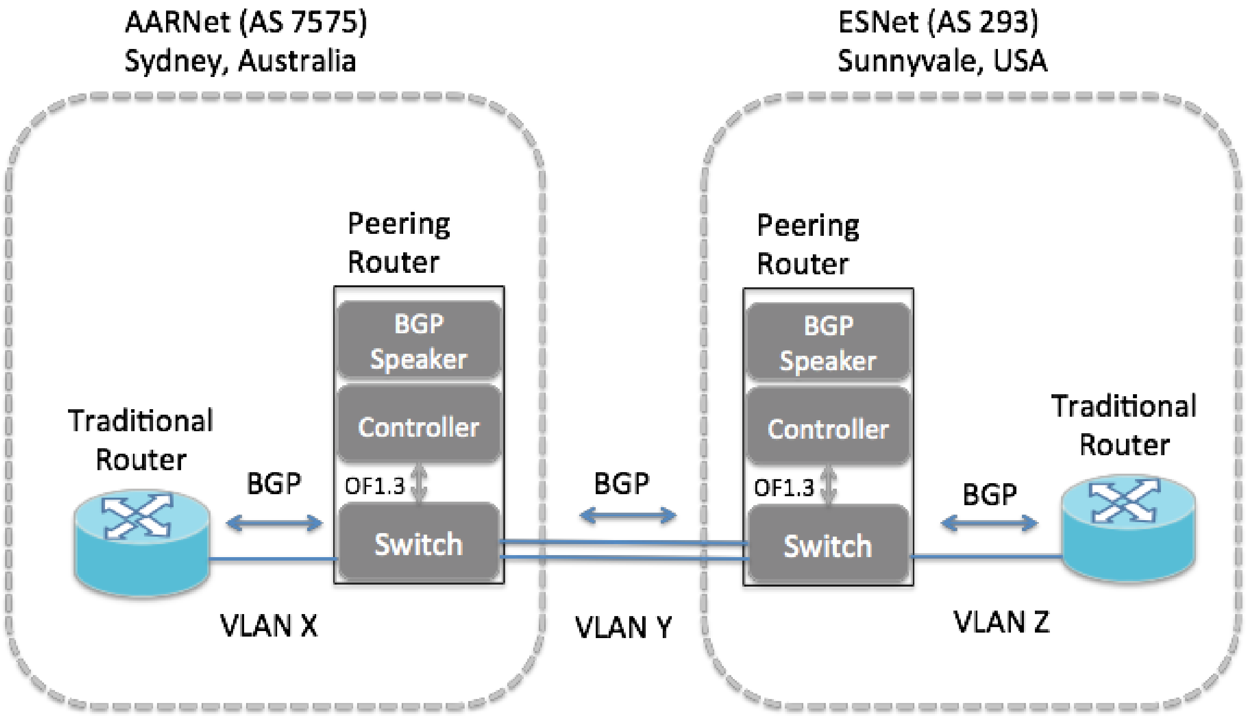


Figure 1: Anchor Stack Apollo Deployment Scenario

A peering router allows SDN technologies to interact with traditional networks seamlessly. But it also presents the opportunity to expand the work in many different ways, as shown in Figure 2. For example, the controller can control several switches in physically different locations to create a distributed router. Or the switches can be arranged in a Clos formation to create a scale-out router eg. 192 X 10G ports. Finally the switches can themselves individually behave as independent routers in a private-AS. All of these ideas shown in Figure 2 represent possible extensions to Anchor Stack Apollo.

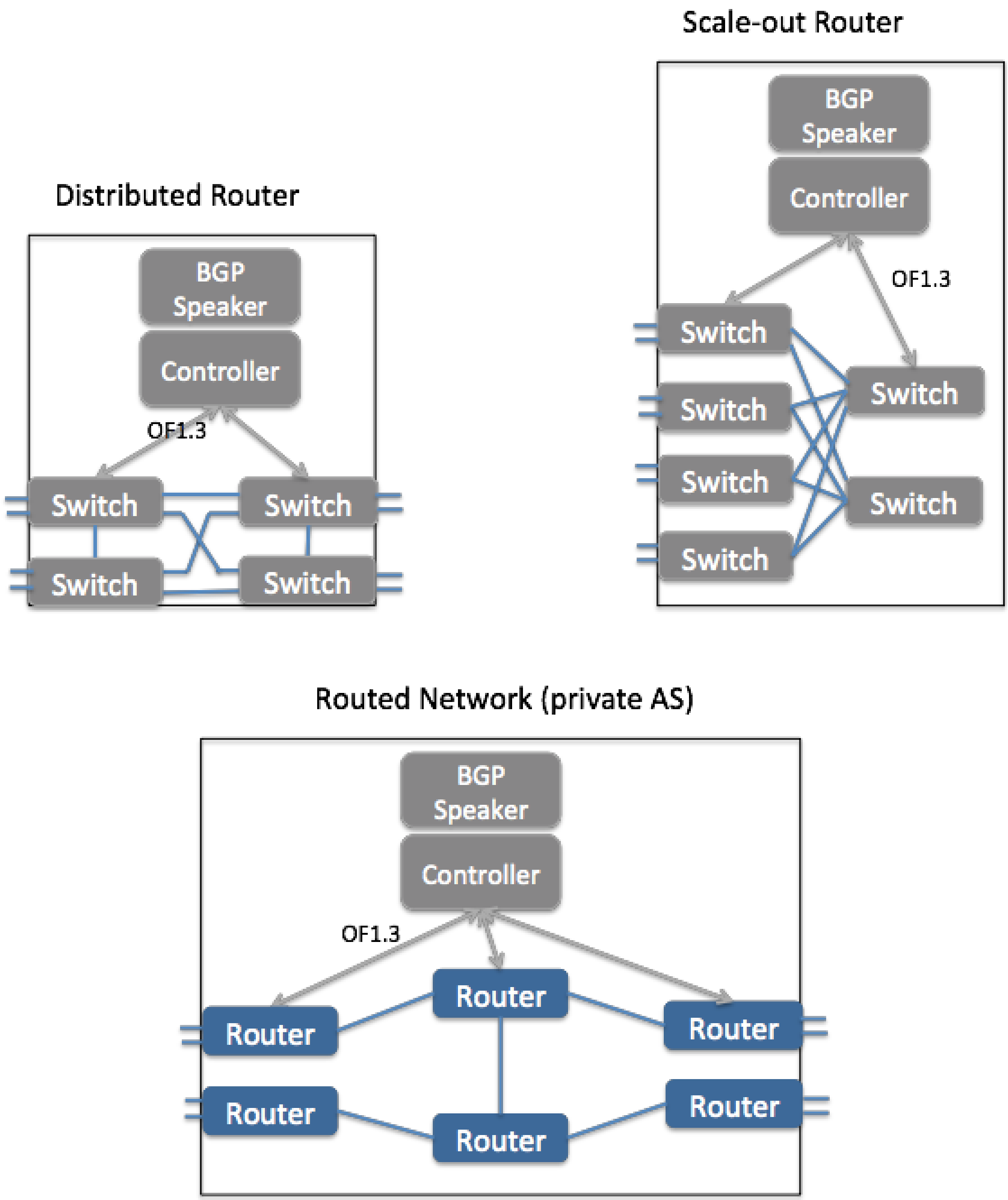


Figure 2: Extensibility Scenarios

Peering Router Requirements

- E-BGP peering with multiple BGP peers
- 25K routes in FIB
- Resolve next-hop IPs for routes received by BGP, and forward traffic based on IP destination/prefix to the next hop.
- ARP & ICMP handling

- Vlan handling for vlan-ids assigned to switch interfaces
- Drop the following traffic
    - traffic with dst MAC that does not correspond to interface MAC addresses or router-loopback interface MAC address
    - traffic on VLANs other than ones configured for the interfaces
    - untagged traffic
    - all Ethernet traffic that is not ARP and not IPv4 (for example, this will drop IPv6 traffic)
    - all IPv4 traffic that does not match in the FIB and is not destined to the Router IP addresses (interface or loopback)
    - Multicast traffic

Anchor Stack Apollo Reference Architecture

See Anchor Stack Architecture and Integration Guide for detailed information about the architecture and component integration for Apollo. Below is a diagram providing an overview of the stack architecture.

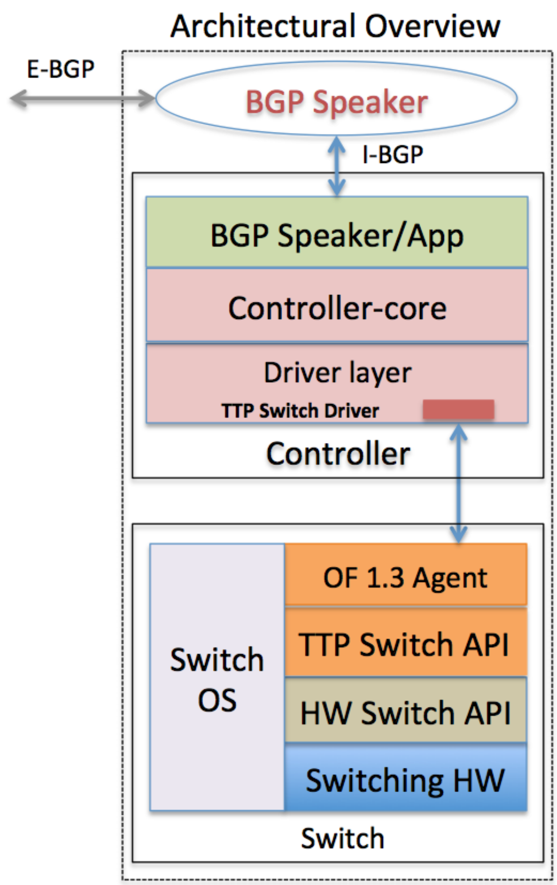


Figure 3: Anchor Stack Apollo Architecture

This reference stack includes the following that together create a Peering Router:

- An open source BGP implementation like Quagga or Bird
- An open source OpenFlow Controller like ONOS, Ryu or ODL
- An open source Driver layer for the controller that allows the implementation of hardware-pipeline specific drivers for hardware from different vendors
- The use of OpenFlow v1.3 and Typed Table Patterns (TTPs) in both the controller driver and the underlying switch
- An OpenFlow 1.3 agent in the switch and the associated 'gluework' required to tie the agent to the driver for the switching hardware
- Switch ASIC driver: Written to the forwarding hardware. Provides a hardware abstraction interface.
- The switching ASIC: Does the actual packet switching. Provides a register interface for its control.
- "The rest" (shown as "Hardware/OS/Other" in Figure 3): This includes the physical hardware and operating system. This subcomponent is responsible for maintaining the environment appropriate for the forwarding element including the operating system context.

Data Plane Switch Component

The data plane switch component for Anchor Stack includes the data plane switching mechanism (hardware or software), the operating system and software stack necessary to control the switch, and all software necessary to control and manage components necessary for the operation of the switch (fans, power supplies, optical interface components, etc.) The Apollo release of Anchor Stack uses OpenFlow 1.3, so the switch must expose this interface. A TTP describes the pipeline configuration for the switch and determines the design of the controller-side driver for the switch.

Anchor Stack will leverage and integrate open-source components wherever possible. At the same time, it will be inclusive of proprietary components via open interfaces. One example of this is the use of OF-DPA which is an open interface for the switching hardware that gives access to features normally exposed only under NDA.

Another example is the inclusion of proprietary vendor equipment from OEM vendors. In such cases the open interface is the OpenFlow protocol (v1.3) and the underlying Typed Table Pattern (TTP) it exposes. The data planes for the peering router stack can be both proprietary or open, as long as they expose an open pipeline description. See the Anchor Stack Architecture and Integration Guide for more details.

SDN Controller Component

The controller must support OpenFlow 1.3 with Typed Table Patterns (TTPs) as a SouthBound interface, and it must support I-BGP as a NorthBound interface. Details are described in the Anchor Stack Architecture and Integration Guide document including descriptions of the TTPs used for the Apollo release.

As shown in Figure 3, Anchor Stack Apollo, which focuses on the BGP peering application, assumes that the controller is divided into a core platform and an I-BGP speaker application. This interface connects with an external BGP speaker such as Quagga allowing connectivity to the BGP control plane. Again, see the Integration guide mentioned above for more details.

Orchestration Component

Apollo will not support an Orchestration Component (such as OpenConfig) in its initial release. This may be a candidate project for community support.

Roadmap

Under development.

- ONOS for BGP peering application and Apollo release
- Switch targets: Starting with Whitebox and Corsa, integrating TBD ASAP
- ODL integration plan
- Use Cases for ODL deployment
- More use case plans

Validation

Give reference to documentation for testing.

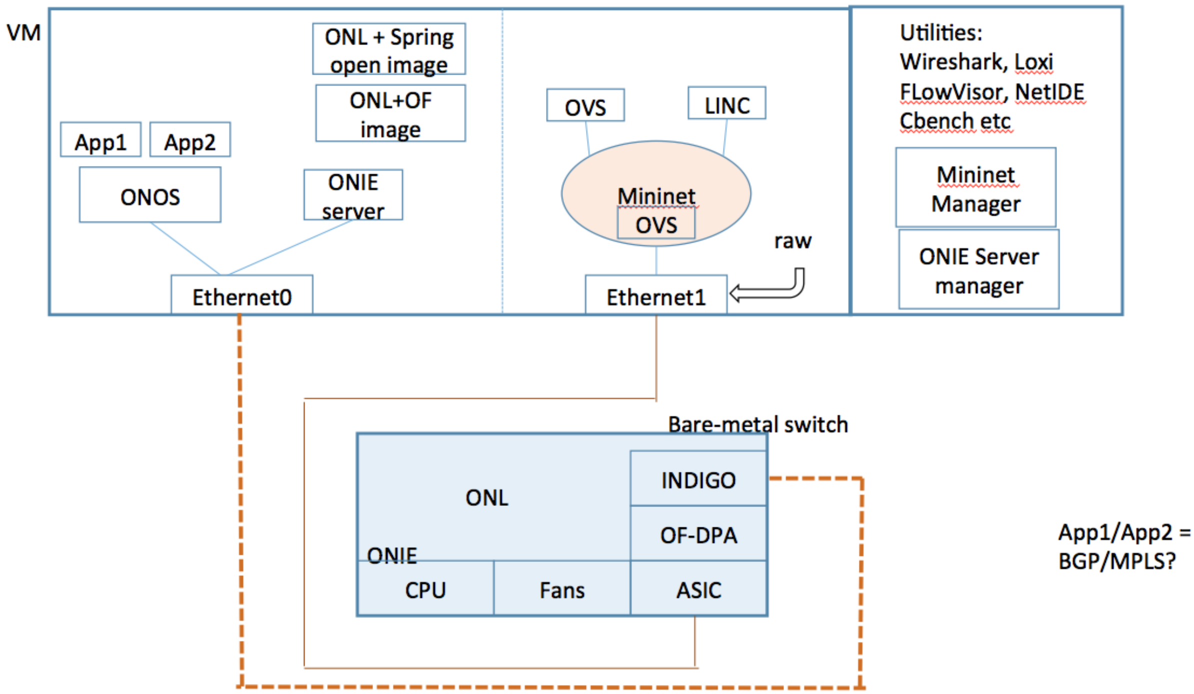
Demonstration

TBD: Describe demo setup and logisitics.

Distribution

While most of the components of Anchor Stack are available independently, a distribution package will be created centered around a VM containing the controller, local-BGP application, and a virtual switch instance. Details TBD.

Below is a diagram showing a possible deployment of the Anchor Stack distribution.



Document History

Release	Date	Notes
1.0	3/13/15	First release