

# Network Architecture

- [Introduction](#)
- [Physical Networking Architecture Considerations](#)
- [Layer 2, IP and BGP Architecture](#)
  - [Rover](#)
  - [Unicycle with SR-IOV datapath](#)
  - [Unicycle with OVS-DPDK datapath](#)
- [LAG Considerations](#)
- [DHCP, HTTP PXE and PXE Booting](#)
  - [Regional Controller Deployment](#)
  - [Rover Deployment](#)
  - [Unicycle Pod Deployment](#)
- [IP Address Plan](#)
- [BGP Networking](#)

## Introduction

## Physical Networking Architecture Considerations

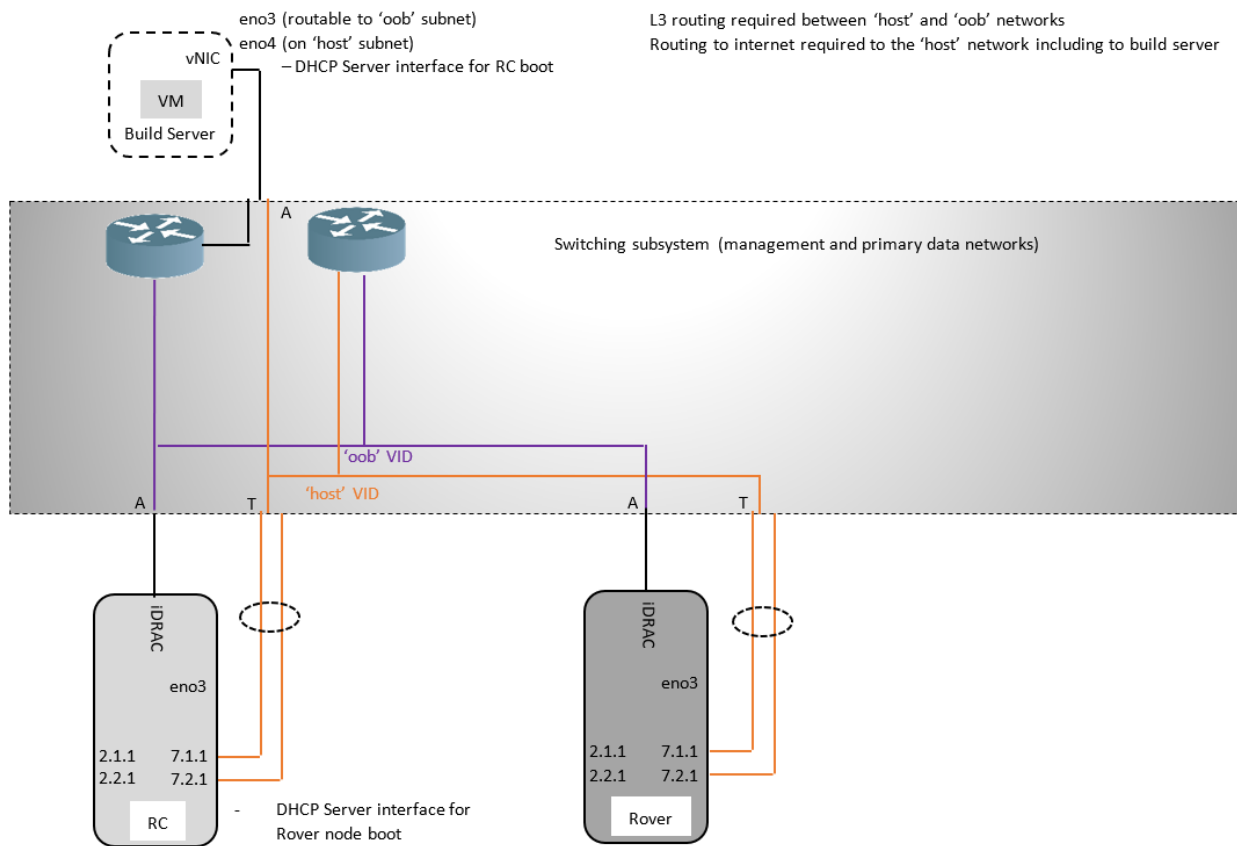
The Network Cloud blueprints deploy Rover and Unicycle pods at multiple edge locations from a Regional Controller. As such the networking architecture spans the WAN between the RC and edge pods as well as the edge sites themselves.

Clearly it is not desirable to span L2 broadcast domains over the WAN nor is it desirable to span L2 domains between edge sites. Thus the networking architecture allows for deployment using local isolated L2 domains at each edge site as well as support for IP connectivity only across the WAN.

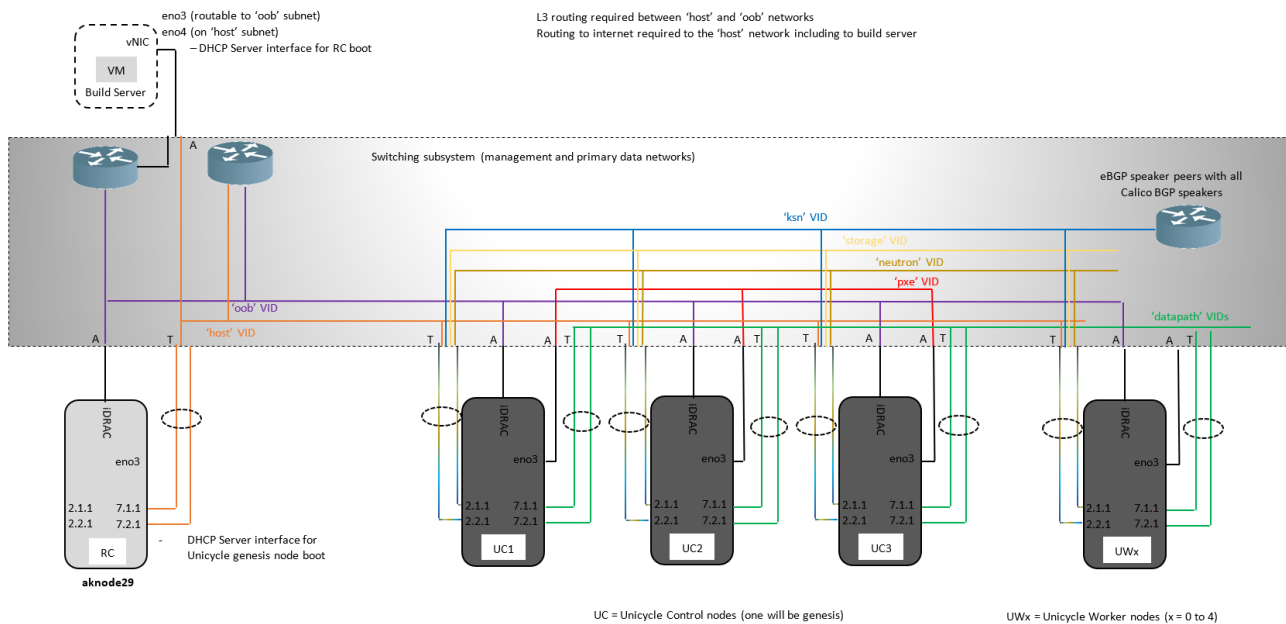
## Layer 2, IP and BGP Architecture

The diagrams below show the physical, layer 2, IP and BGP architecture for the Network Cloud family of blueprints in R1.

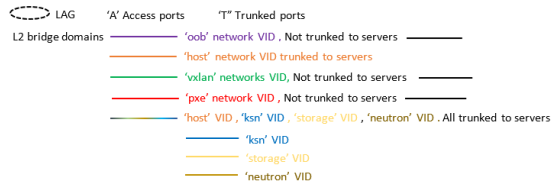
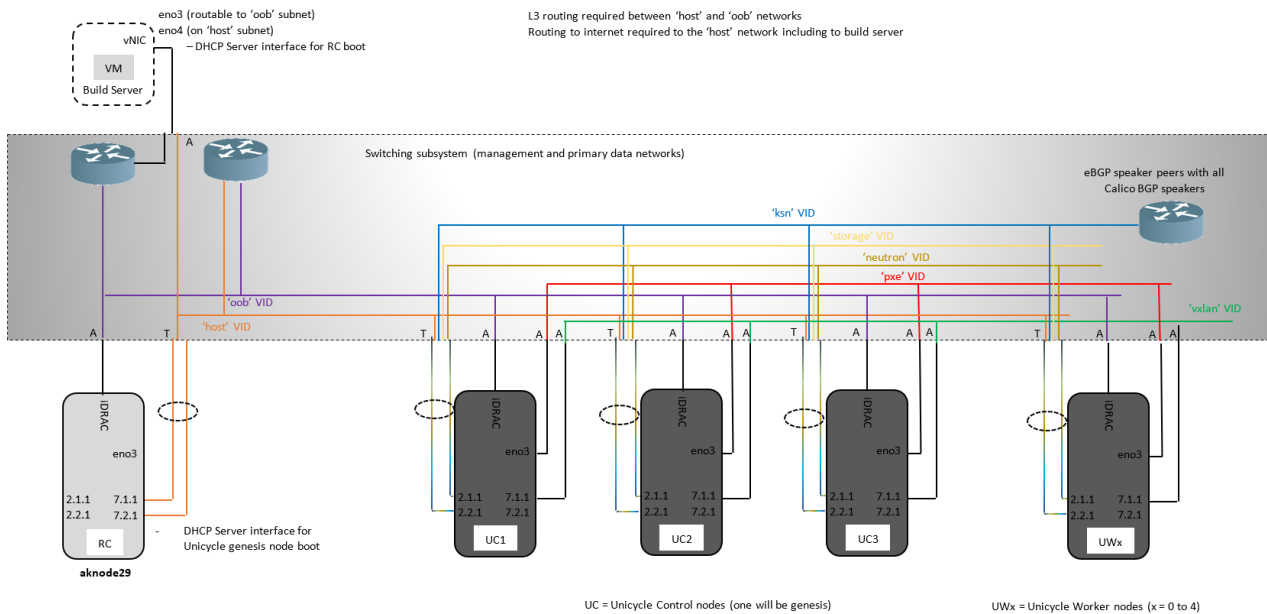
### Rover



## Unicycle with SR-IOV datapath



## Unicycle with OVS-DPDK datapath



## LAG Considerations

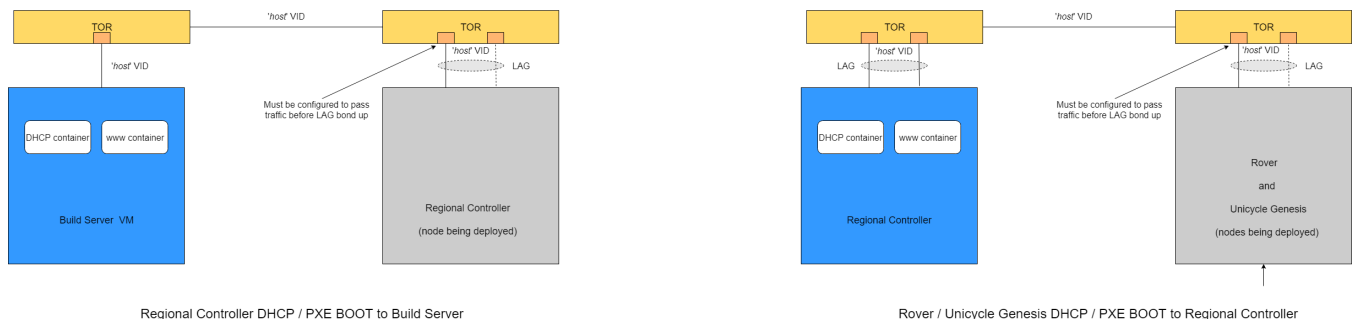
The RC (when installed on a bare metal server), Rover and Unicycle Genesis nodes all boot via their VLAN tagged 'host' interfaces which are pre-provisioned on the serving TOR switches with LAG bonding. Since booting occurs before the linux kernel can bring up its LAC-P signaling the TOR switches must be configured to pass traffic on their primary (first) link before the LAG bundle is up.

The switch configuration for the network bond for the 'host' interfaces must be set to bring up the bond on the first interface prior to lacp completing negotiation.

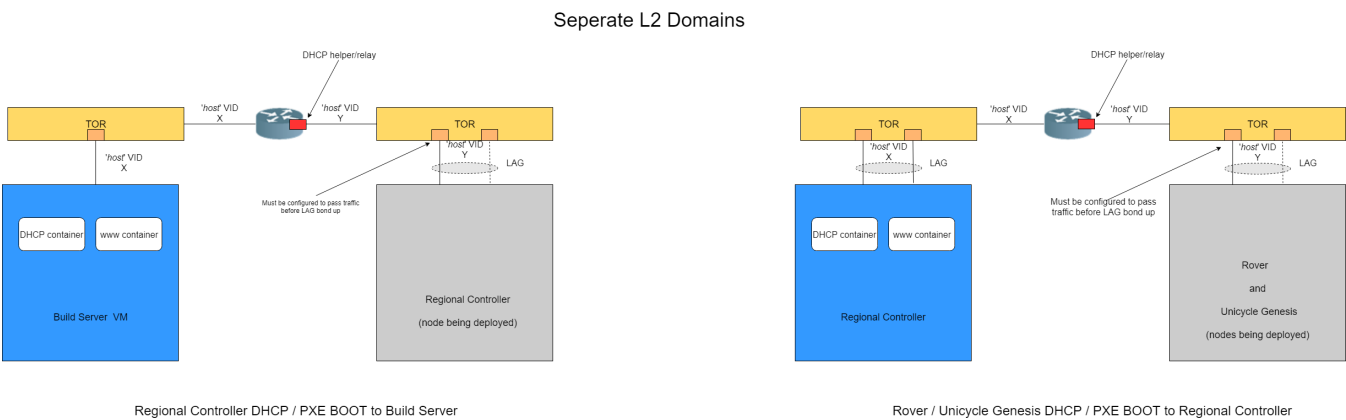
- For Junos OS this option is typically called **force-up** and should be set on the first interface in the bond.
- For Arista, this option is typically called **lacp fallback**.
- Please ensure any TOR switch that is to be used supports this functionality and then refer to your network switch documentation to determine the correct configuration.

The 'host' network interfaces on all DHCP clients and servers should be located on the same L2 network so that the DHCP Request/Reply messages broadcast from the client reach the server and vice versa.

### Same L2 Domains



Alternatively DHCP helper/relay functionality may be implemented on the TOR to which the DHCP clients are attached to allow inter subnet DHCP operation between DHCP client and server.



## DHCP, HTTP PXE and PXE Booting

Network Cloud family blueprint deployments involve a number of different DHCP servers and HTTP PXE and non HTTP PXE booting procedures during the automatic deployment process.

### Regional Controller Deployment

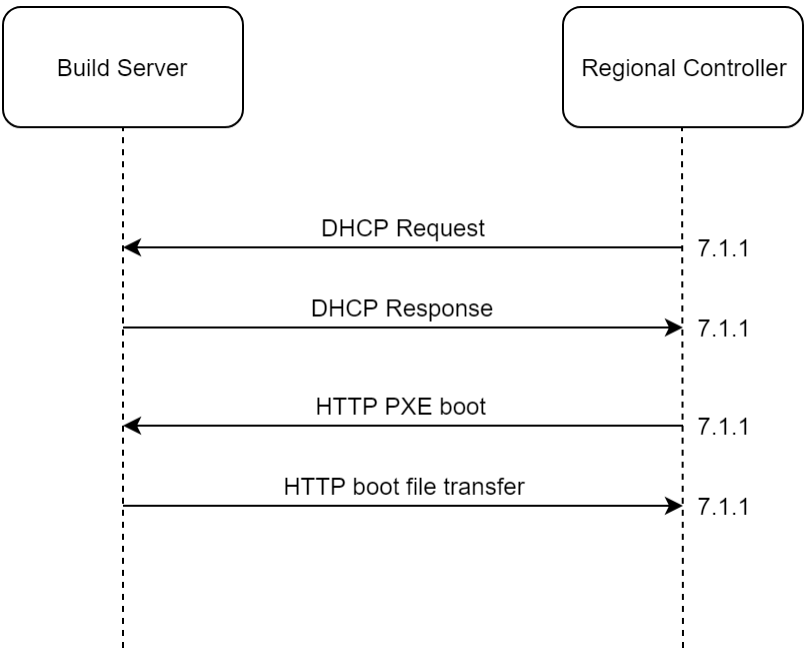
When installed on a bare metal server, the RC is configured by the Redfish API calls issued by the Build Server to make DHCP Requests and then HTTP PXE boot over its vlan tagged 'host' network interface. (When the RC is installed in a VM the BIOS setting procedures are not relevant and since the operating system installation is considered a pre-requisite the RC operating system installation steps are not preformed).

The Build Server acts as the DHCP and HTTP PXE boot server for the RC via its own 'host' network interface.

The 'host' network must provide connectivity for these processes between the RC and the Build Server.

In R1 the 'host' network has been verified as a single L2 broadcast domain.

The RC DHCP/HTTP PXE process flow is summarized below.



### Rover Deployment

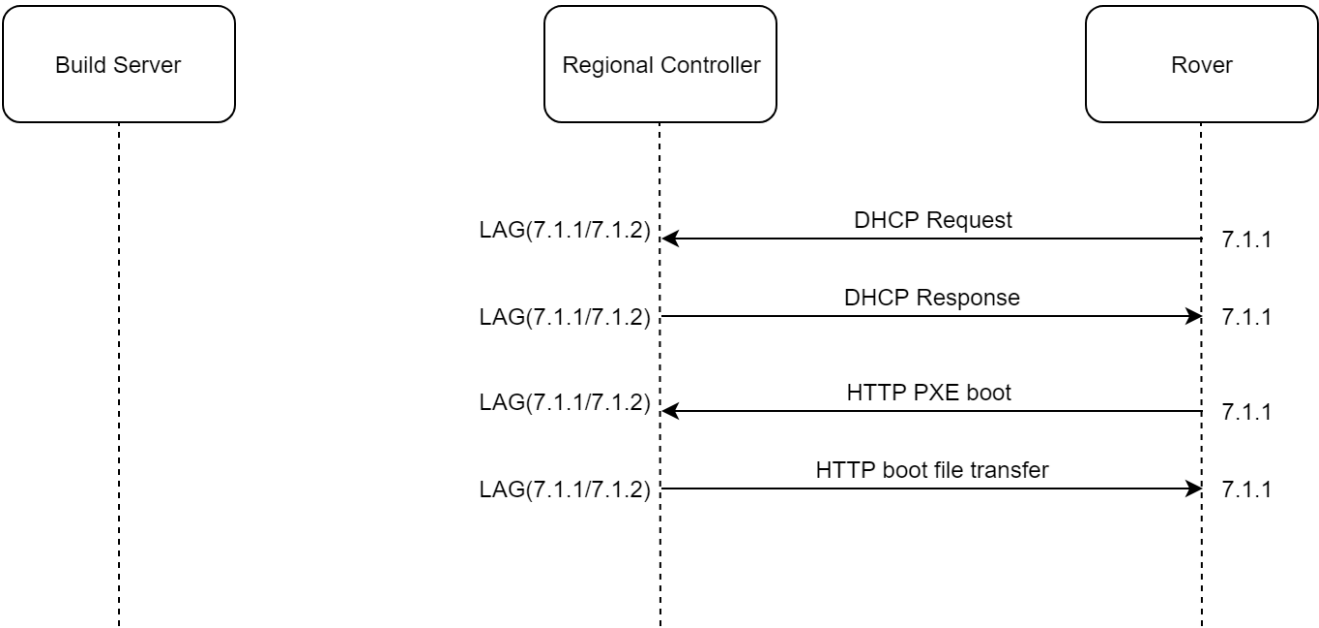
The Rover server is configured by the Redfish API calls issued by the RC to make DHCP Requests and then HTTP PXE boot over its vlan tagged '*host*' network interface.

The RC acts as the DHCP and HTTP PXE boot server for the Rover server via its own tagged '*host*' network interface.

The '*host*' network must provide connectivity across the WAN between the Rover edge site and the RC in a more centralized location.

In R1 the '*host*' network has been verified as a single L2 broadcast domain.

The Rover DHCP/HTTP PXE process flow is summarized below.



It is possible to split the '*host*' network spanning the WAN into multiple routed L2 domains using DHCP helper/relay functionality on the Rover's TOR but this is unverified in R1.

## Unicycle Pod Deployment

A Unicycle pod's Genesis node is configured by the Redfish API calls issued by the RC to make DHCP Requests and then HTTP PXE boot over its vlan tagged '*host*' network interface.

The RC acts as the DHCP and HTTP PXE boot server for the Unicycle Genesis node via its own tagged '*host*' network interface.

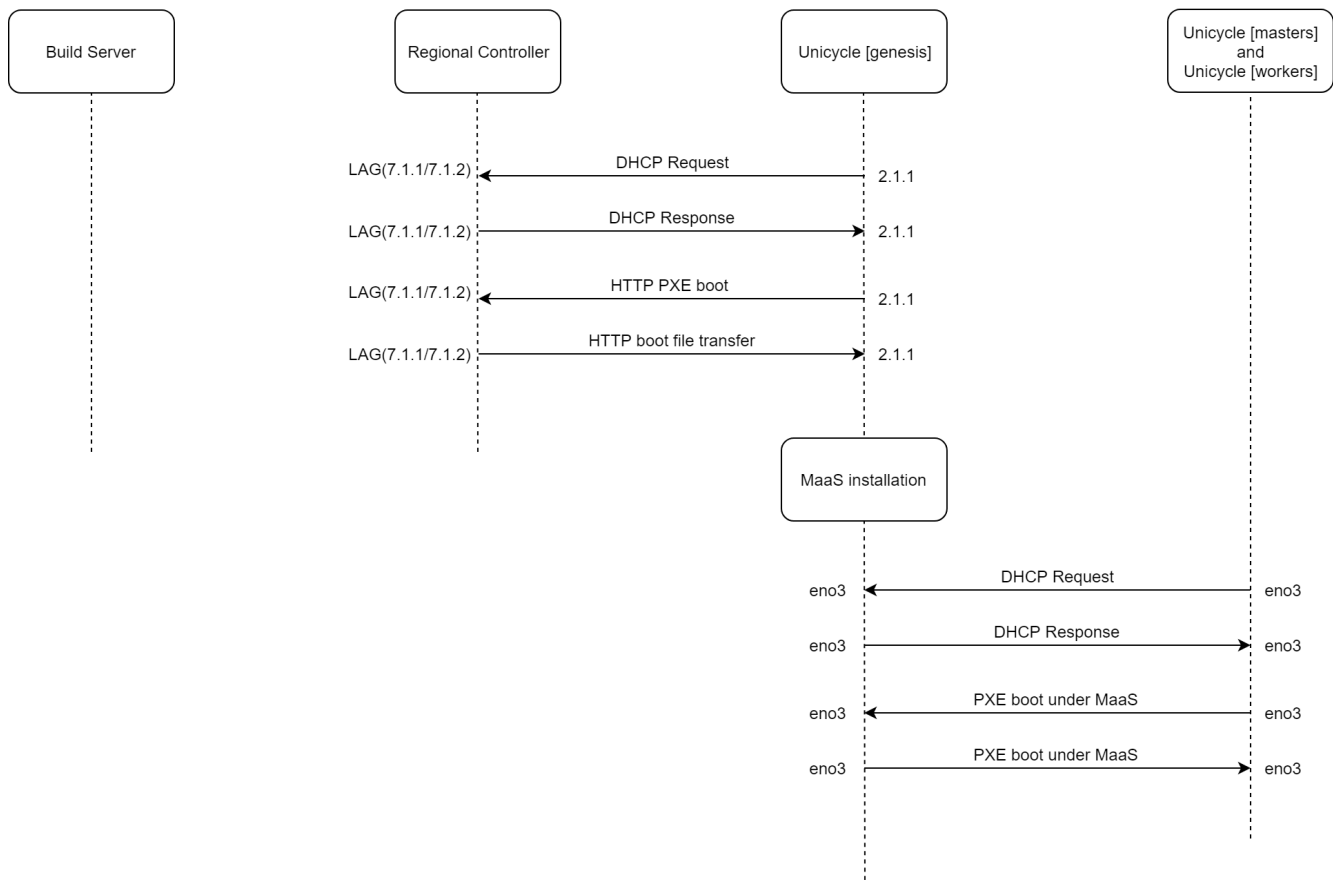
The '*host*' network must provide connectivity across the WAN between the Unicycle pods's edge site and the RC in a more centralized location.

In R1 the '*host*' network has been verified as a single L2 broadcast domain.

Unicycle masters and workers (all Unicycle nodes other than Genesis) are configured by the Redfish API calls issued by the RC to make DHCP Requests on the edge site's local '*pxe*' network's L2 broadcast domain. The Genesis node's MaaS acts as the DHCP server.

Unicycle masters and workers (all Unicycle nodes other than Genesis) are then provisioned by the MaaS server running on the Genesis node over the remote site's '*pxe*' network.

The Unicycle DHCP/HTTP PXE/PXE process flow is summarized below (omitting the Redfish API calls):



It is possible to split the 'host' network spanning the WAN into multiple routed L2 domains using functionality such as DHCP helper/relay on the Unicycle Genesis TOR but this is unverified in R1. Note: this is not applicable to the other nodes in the Unicycle pod as the DHCP and PXE boot process is supported over the local 'pxe' network.

## IP Address Plan

In the R1 release the following subnets are required as a minimum

Network name	Subnet size	Needs to be externally routable**?	Function
'oob'	Any*	No	Provides the connectivity between the Build server and RC and then RC and Rover/Unicycle nodes for Redfish API calls to configure/interrogate the servers' BIOS.
'host'	Any*	Yes	Provides the edge site's isolated network for MaaS provisioning (MaaS runs on the genesis node once provisioned). It does not need to be routable to any other network.
'pxe'	/24 fixed in R1	No	Provides a remote site's L2 local network for MaaS provisioning.
'ksn'	/24 fixed in R1	No	Provides the connectivity and BGP route exchange for the calico network.
'storage'	/24 fixed in R1	No	Provides the openstack storage network
'neutron'	/24 fixed in R1	No	Provides the openstack neutron network
'datapath'	Defined in openstack	No ***	Provides the openstack datapath network for the SR-IOV variant VMs
'vxlan'	/24 fixed in R1	No****	Provides the openstack storage network for the OVS-DPDK variant VMs

Notes:

\* The 'oob' network may be split in to multiple subnets if required. Each server except the Build Server (and the RC if built on a VM) requires an iDRAC/iLO address which must be manually pre-provisioned into the server.

\*\* Whilst not required designs where these subnet are externally routable are valid.

\*\*\* Whilst it is not required to have external or inter subnet connectivity on the datapath for VMs to have interaction with any entity outside the deployed edge pod this would be required.

\*\*\*\* In R1 for the Unicycle OVS-DPDK blueprint all VM traffic is vxlan encapsulated

## BGP Networking

In case of Unicycle calico is used as the kubernetes CNI thus pods routing information is exchanged via BGP between all controller and worker nodes.

Two architectural options exist to implement the route exchange for calico pods

- (1) Using an external fabric BGP router and peering the calico pods to this router using eBGP, or
- (2) Using the more traditional full iBGP mesh between all calico pods

Validation has been primarily performed using the external fabric router approach. However limited validation has also been performed using the internal iBGP calico mesh.

In the first case the external fabric router needs to eBGP peer with all calico pods. To do this dynamic BGP peering is defined in the fabric router to accept BGP connection requests from any BGP speaker with an IP address from the calico 'ksn' network.

The site specific input yaml files documented in the [Unicycle Deployment](#) section of the release 1 documentation contain examples from the validations lab deployments of the the BGP related information required to force the calico pods to peer using eBGP such as own ASN and the external ASN of the fabric BGP router.

The Akraino/Airship deployment process configures the Unicycle pod's BGP configuration of the calico nodes. Modification of the Unicycle site specific input file fields allows the user to implement either the first or second BGP architectural option and in case of the first option the values of the external eBGP peer and its ASN.