E3Fabric control plane model

Items	description
Docs creating date	2017.8.1
Docs finishing date	
Initial author	jzheng.bjtu@hotmail.com
Description	This article mainly introduces the architectural model of the e3fabric and resource abstraction.

1. Underlay Infrastructure Network

Underlay infrastructure network, also as physical network, is called Ethernet cloud in TRILL terminology, this means no matter how the underlying network is structured and built, it must provide L2 connectivity[1]. Optionally, it can be:

- VLAN confined network which can support access VLAN port and trunk VLAN port, it depends.
- Clustered switch which can employ its own HA protocol as it wishes, this is to make the underlying network to be highly available.

We maintain the physical LAN segment object (administrator is responsible for this):

```
E3PhyLanSeg:
   id: <uuid>
   name: 'phy-lan-0' #unique
   lan: '130.140.25.0/24'
   vlan: 0 #non-zero mean this is a trunked vlan
segments, all the packet should be tagged with that
vlan
```

Also host is to record the host which joins the underlying network as vswitch:

```
E3FabricHost:
   id: <uuid>
   name: 'server-host'#unique
   ip: '10.0.12.1' # management address
   bonding_type: ['native', 'bonding']
   bonding_id: <bound-e3fabrichost-id> #foreaign
key,and nullable.
   active: [True,False]
```

When aggregating two or even more vswitches as one, here we use bonding_id if not NULL to index the real host which is bonding_type of 'bonding'.

Also another list must record the allocated IP addresses which is bound to a certain interface can be released when the virtual switch is destroyed.

```
E3interface:
   id: <uuid>
   lan-seg-id: <lan-seg-uuid>
   fabric-host-id: <fabric-host-uuid>
   address: '130.140.25.100/24' #unique
   enabled: [True,False]
   type: ['exclusive', 'shared', 'unknown']
   hw-id: '<host-name>-0000:41:00.1' #unique
   active: [True,False]
   reference_count: 0 #
```

This is not operational information, so it should be stored in SQL database, but database must be manipulated in a central node.
Then we can define the following operations (included but not limited to):

[1] We assume an 802.1d compatible L2 switch will never compromise a MPLS labeled packet.

```
<list> fetch_physical_lan_segs();
<list> fetch_fabric_host();
<list> fetch_fabric_host_as_topology();
<list> fetch_ifaces_into_seg(seg_id);
<list> fetch_ifaces_of_host(host_id);
<list> fetch_ifaces_of_host_into_seg(host_id,seg_id);
<list> fetch_neighbors_of_host(host_id);
```

2. Concerning Tenant Network

The underlying physical links[2] usually can be either exclusive or shared. The exclusive link has guaranteed bandwidth and throughput by hardware, thus being extremely expensive, but we can still find its room for use. The shared link shared by tenants' network instances, each time when a network is instanced, the weight of it is changed, but how to quantize it?

For simplicity, we use a reference count to measure how heavy the interface's load is, for 'shared' interface, the reference count can be larger than 1, however, the 'exclusive' interface's reference count can only be 1, and **both types of interfaces can transit to other type as long as their reference count is zero**. Then the basic path allocation is to find those legal hosts and interfaces to calculate least loaded path within the fabric.

When we are allocating virtual network resource, we should first to decide the leaf virtual switch nodes to join the the virtual network, in another word, we must find the compute resource somehow (this is out of the scope of this article), and then we can define the topology initially, but some compute instance may join and leave and even migrate the virtual network after the initial allocation of virtual network, all the cases should be taken into consideration.

2.1 Basic path allocation

the path allocation applies both shared and exclusive paths, the only difference is how we get the subset of the available overall topology: if the target path is shared, then all the candidates interfaces in the subset of the overall topology are tagged with 'shared', vice versa.
Here we give the definition of virtual network path.

```
E3Path:
   id: <UUID>
   path_type: ['E-LAN', 'E-LINE']
   physical_link_type: ['exclusive', 'shared']
```

Along with customer zone which indicate those leaf virtual switches that join the path. before going forward, it's should be declared how path could be stored, we store the path as link pairs of interfaces.

[2] Including virtual functions.

```
linkpair:
   id: <auto-increment>
   iface1: < E3interface-uuid-as-fkey>
   iface2: < E3interface-uuid-as-fkey>
   path_id: <path-id-as-fkey>
```

Note that these linkpairs are not order, we can still get related interfaces and hosts and analyze and validate whether the pairs set is a legal path.

2.1.1 E-LINE virtual network instance

If the target virtual network service is E-LINE, we could search the optimal path by employing Dijkstra's algorithm[3]. Given two leaf virtual switch nodes, we construct the classical search table to iterate step by step until we find the optimal path.

One thing to emphasize is the input data set is not regular to the algorithm, since the edges are indirect, and determined by the interfaces attached to the same physical LAN segments.

Imagine that we have *iface1* on host-1 and *iface2* & *iface3* on host2, all these three interfaces are attached into the same one segment, to calculate the optimal path, we should make the assumption we are two virtual links between the two host across the LAN segment with respective link weight $W_{iface1}+W_{iface2}$ and $W_{iface1}+W_{iface3}$.

2.1.2 E-LAN virtual network instance

Given the legitimate data set (hosts and interfaces) as input, we can get the minimum spanning tree using Prim's algorithm[4]. However, the out put is far more than what we want, all we need is a subtree where all the leaf virtual switches are included, the spanning tree must be pruned, the question is whether the subtree is still the spanning tree of the corresponding subgraph? Here we are going to prove it.

Denote the Graph is G, through the Prim's algorithm's iterations, we have a minimum spanning tree T, remove a point of v whose degree is 1 with edge e, we have a subtree T', G without vertex v as subgraph G'. we need to prove T' is the minimum spanning tree of G'. if not, we have another spanning tree T'' whose weight sum is even lower than T', that's $S_{T''} < S_{T'}$, then consider vertex v and its adjacent edges, we want to find a least weighted edge and make v part of the newly built spanning tree again, of course, the edge is still e (or edge which has the same weight with e), the newly built tree 's weight sum is $S_{T''}+W_e$, this is lower than S_T ($S_{T'}+W_e$), so we know the spanning tree T is not the minimum spanning tree of G at all, but this violates the condition that T is the minimum spanning tree of G, so conclusion is true.

[3] https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm
[4] https://en.wikipedia.org/wiki/Prim%27s_algorithm

Once we have a super set of genuine minimum spanning tree, through deep first search, we can find all the point to point like paths which may share same partial link pairs, thus determining the link pairs that belong to the E-LAN path. then store them into database.

2.2 Incremental path allocation

Initially we may know how many compute zones will join the virtual network, however, tenants' compute entity may come and go at any time, the virtual topology may change as well, for E-LINE virtual network path, once the allocated, it will last until it's reclaimed, if the connected compute entity must migrate, the E-LINE path is reallocated for simplicity. Path migration is not within the scope of this section.

Incremental path allocation only cares about E-LAN virtual network service, when newly come compute entity joins the virtual network, the virtual network path may change as long as the compute entity does not affiliate to any leaf virtual switches, this means that a new leaf virtual switch will join the virtual network path.

As Prim's algorithm says, we already have the snapshot that the existing path constitutes a minimum tree, as the algorithm steps forward, more and more node will join the tree until the target leaf virtual switch node comes being part of the tree. Later as normal procedure does, prune it, store it into database.

2.3 Partial path reclaim

For E-LAN virtual network instance, when compute entity leaves, the affiliating leaf virtual switch node may become homeless, which means no compute entities attach themselves to the leaf node. the leaf virtual switch node must leave the E-LAN spanning tree.

Remove the leaf virtual switch node along with its edge, if the adjacent node is still 'leaf' node (with only one edge), remove it, as the procedure goes on, it stops until all the 'leaf' nodes are genuine leaf virtual switch node.

Note if and only there are two leaf virtual switch nodes in the virtual network, if one of them leaves, there is no link pairs in the tree, but the I-component of leaf virtual switch still exist.

2.4 Path migration

When E-LINE virtual network instance migration happens, he virtual link must be reclaimed and allocated according the new positions of lead nodes.

For E-LAN instance, only part of the virtual links are reclaimed and allocated later, so the procedure is the combination of section 2.3 partial path reclaim and 2.2 incremental path allocation .

3.Label distribution

After we get the link pairs set, we know the topology, here we uniformly call it tree, since line is kind of tree in our topology. denote the set of leaf nodes as S_L, and the set of immediate nodes as S_M, then $S_M+S_L=S$ where S is the whole set of nodes. We have the number of given set S as $|S|$, also as cardinality.

Given a Tree S which has leaf nodes set S_L and non-leaf nodes S_M, here we shall analyze how many labels are installed to these nodes.

3.1 disaggregated label distribution model

In disaggregated label scenario, every end to end link traffic will be regarded as the same FEC, the traffic toward the same leaf node will not be aggregated. for any node n in S_M, it will generate label bindings for all the leaf nodes, thus the overall label binding generated by S_M is $|S_L|x|S_M|$, additionally, leaf nodes will generate label binding to their directly connected immediate node, the total unicast label binding will be $|S_L|x(|S_L|-1)$. The multicast is edge related, the total edge is $|S|-1$, and for each edge, there will be 4 bidirectional label binding, so the the total multicast label bindings is $(|S|-1)x4$.

In summary, we have label bindings count:

$$\begin{aligned} &|S_L|x|S_m|+|S_L|x(|S_L|-1)+(|S|-1)x4 \\ =&|S_L|x(|S_M|+|S_L|-1)+(|S|-1)x4 \\ =&|S_L|x(|S|-1)+(|S|-1)x4 \\ =&(|S|-1)x(|S_L|+4) \end{aligned}$$

so the total number of label binding has the spatial complexity $O(|S||S_L|)$, i.e. $O(|S|^2)$

3.2 aggregated label distribution model

At a certain immediate node, the traffic toward a particular destination node could be aggregated, however, new complexity is introduced, here this part is omitted, in fact, we are going to use disaggregated label distribution.

3.3 unicast label installation

imagine that we have the following spanning tree as Fig 1 shows, the leaf nodes set is {L1, L2, L3}, while immediate nodes set is {n1, n2, n3}, for simplicity, given the immediate node n2, here we demonstrate how <L1, L2> LSP labels are installed.

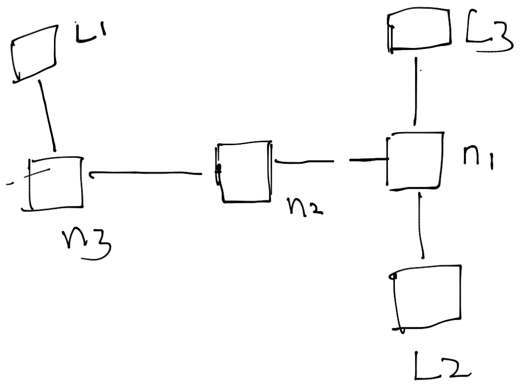

Figure 1 sample spanning tree for installing labels

Once the label distribution agent on n2 receive notification that topology is changed, it analyzes and knows there is a new LSP established between L1 and L2, n2 first establishes an ILM entry with left side interface in Fig 1, which is illustrated by label binding <from: L1, to: L2, label: 2> and notify this label binding to n3 to tell him, through label 2, the the traffic from L1 to L2 is viable. However, at presence, n2 does not know how to forward with packet labeled 2 since no NHLFE is correlated with ILM entry. Once n2 receive similar notification <from: L1, to: L2, label: 3> from right side interface from n1, it says: label the packet from L1 to L2 with 3, I can forward for you, then n2 knows it, correlate these two label bindings, when packet passes through n2, if the packet is labeled with 2 from left side interface, then swap 2 with 3, send out to the right side interface.

However, LSP should be bidirectional, this means we need to install the reverse similar label binding entries to lead traffic from L2 to L1.

Of curse, every node will remember the label on its own locally.

3.4 multicast label installation

multicast LSP is similar to unicast except that the NHLFE is a set or an interface list, in the rest of the thesis, we name it multicast list. the node still advertises to its neighbor through which label the multicast traffic is viable, the list initially is empty, upon receiving a multicast label binding notification, it updates the multicast list entry.

When forwarding a multicast label packet, walk through the multicast to find the interfaces except the one from which it's received, send a copy to these interfaces. [5]

[5] this how LAN service is emulated.