Network Working Group
Internet Draft
Intended status: Information
Expires: April 2012

D.Stiliadis
F.Balus
W. Henderickx
Alcatel-Lucent

N. Bitar
Verizon

Mircea Pisica
BT

October 31, 2011

Software Driven Networks: Use Cases and Framework

draft-stiliadis-sdnp-framework-use-cases-01.txt

Abstract

This document presents an application framework and associated use cases to help define the scope of the SDNP working group. The use cases start with an abstract representation of a multi-tier application and illustrate the composition of a network service that can meet the requirements of the particular application. The service composition process is split into several steps, including application requirement specification, network mapping, service binding, and policy control. We also provide examples of interactions between an SDNP controller and L2 and L3 control layer protocols in order to deliver the end-to-end service. Finally, as a derivate of that, an architecture framework for SDNP that meets these requirements is proposed.

Table of Contents

1. Introduction

In order to define the scope of the proposed SDNP working group, we present a generic application use case and a sequence of steps needed for mapping the application requirements to a deployable network service. We also present a generic architecture framework that can meet such requirements.

The goal of SDNP is to define a method where applications can request services from the network and these services can be automatically deployed. We view technologies such as FoRCES and OpenFlow as complementary and orthogonal to SDNP. Unlike Openflow, where the goal is to introduce a disaggregated control layer, the goal of SDNP is to enable existing control planes to become more adaptable to application requirements and to allow rapid and reliable configuration changes by introducing a framework for communication between applications and network control planes. More importantly, the framework should be applicable to communicating application requirements even when a variety of network technologies is used to provide the required services. In this context, SDNP is also useful to OpenFlow type networks, since it provides the interface between applications and control planes implemented in an Openflow controller.

In order to achieve these goals, applications should be able to specify their requirements in a generic way and these requirements must be translated in an actual network service. In general, application developers do not know in advance the type of networks that will be used and they would require an abstract and flexible framework for defining their requirements.

Often times a service spans multiple administrative domains where given application requirements are met by different networking technologies. For example, providing network isolation for the application servers can be achieved by VLANs, MPLS or other underlying L2 or L3 technologies. It is possible that an application that is distributed between multiple data centers and networks be deployed through VLAN isolation in one domain and MPLS isolation in another domain, and an IP VPN in between. The application does not care as to which underlying technology is used to implement the isolation, as long as the properties of the isolation are guaranteed.

In most instances the types of network technologies that should be used to deliver a given application will depend on policies either set by the network service provider, cloud service provider, another

administrative authority, and/or the application itself. In multi-tenant environments there can be a hierarchy of policy objectives set by different organizations and the implementation of the network service must take into account all policy restrictions. For example, an enterprise IT organization can define that any application deployed by their employees must adhere to specific security requirements, such as encrypted tunnels between application servers. At the same time, because of an SLA contract, the service provider requires that the service is always deployed over redundant paths. In these cases, the requirements imposed by the application will be mapped in such a way that they comply with both policies. If for example a user within that organization tries to deploy an application with guaranteed bandwidth between servers, because of the policies defined above, the service instantiation will be over an encrypted and redundant path that satisfies the bandwidth constraint. In the next sections we provide an overview of this application driven framework and illustrate with specific examples how an SDNP controller can interact with control layer protocols.

2. General Terminology

Network Spec: The definition of the network service requirements by an application.

Network Mapping: The transformation of application requirements to a network service on specific network technologies.

Binding: The binding of a network service to specific network elements.

SDNP Controller: The software controller that allows the specification of an application Network Spec, and implements the network mapping and binding functions. In binding the network mapping to a network element, the SDN control may talk to the network element directly or via another controller such as an Element Management System (EMS) or an open flow controller.

3. Framework

First we illustrate a generic framework for mapping application requirements to network services and then we illustrate in more detail the functionality of each component. Note that this framework is just for illustration purposes and specific implementations can combine multiple functional blocks in a variety of ways.

The basic blocks are illustrated in the next Figure:

```
         +------+  +---------------+
         |      |  |Application    |
         |  P   |  |Definition     |
         |  O   |  +------|--------+
         |  L   |         |
         |  I   |  +------|--------+
         |  C   |  |Network        |
         |  I   |  |Mapping        |
         |  E   |  +------|--------+
         |  S   |         |
         |      |  +------|--------+
         |      |  |Network        |
         |      |  |Binding        |
         +------+  +---------------+
```

Figure 1 Generic Framework

o Application Definition: This is a generic representation of the requirements of an application without specifying the underlying technology.

o Network Mapping: This function translates the requirements of the application to an actual network service that can be implemented by a specific network technology.

o Binding: This function maps the abstract network service on control planes and specific network elements.

o Policies: Provides the repository of policies that will drive the translation between generic requirements and network technologies as well as the binding to specific elements.

3.1. Application Definition

An example of multi-tier application definition is shown in the next Figure:

```
                  +------+                   +------+
               ---|Server|--|             ---|Server|--|
               |  +------+  |             |  +------+  |
   +--------+  |            |  +-------+  |            |  +-------+
   |Network |  |            |  |Network|  |            |  |Network|
   |Spec    |--|            |--|Spec   |--|            |--|Spec   |
   |Load    |  |            |  |(LAN)  |  |            |  |(WAN)  |
   |Balancer|  |            |  |       |  |            |  |       |
   +-------+   |  +------+  |  +-------+  |  +------+  |  +-------+_
               |--|Server|--|             ---|Server|--|
                  +------+                   +------+
                   Tier 1                     Tier 2
```

Figure 2 Multi-tier application description

From the application perspective, the definition consists of a set of compute and storage tiers interconnected by a network segment that meets specific requirements. The application does not care about the type of underlying technology, but rather about the properties of the network segment. Generic application requirements can include isolation, bandwidth, communication based on criteria other than shortest path, network redundancy, access control, etc.

In the Example of Figure 2, the application might require that a load balancer distribute load among all the Tier 1 servers. The application does not care about how the load balancer is instantiated or how it is configured, or whether it is a physical or virtual machine. If the first Tier represents a set of Web servers and the second Tier a set of Business Logic servers, the network spec between the Tiers only defines that web-servers communicate with business logic servers over a specific protocol and port and require isolation. It does not define how this isolation is achieved. The network mapping function, depending on the technology chosen will implicitly identify the need for access lists or firewall rules between the Tiers that will prevent any unauthorized access.

A different network spec will define the back-end connectivity requirements between the business logic tier and other enterprise services. For example, it might require that the Business Logic Tier must communicate with the enterprise Data Center over a secure connection, since critical data must be physically protected and stored within the enterprise premises. The application developer does not necessarily need to know whether the application is deployed across one or multiple DCs or what is the level of security required. However, the IT policies that have been supplied to the cloud provider should describe the necessary security mechanisms that must be deployed in order to comply with legal compliance rules or other policies. The policy will define for example that all access to data base Tiers must be encrypted, and the encryption strength that must be used.

Another type of network spec might define that all servers in a given Tier belong to the same LAN, since the service will rely on virtual machine motion for balancing the load or achieving reliability. In another example, machines within a Tier need to form a cluster that must support fencing that is achieved by a majority protocol between the servers. In this case, healthy servers can shutdown network connectivity to failing servers and the network must provide the necessary APIs to achieve that. At the same time, the network APIs must prevent an application belonging to one

customer to affect network connectivity of another customer or another application.

In all the above examples, the application spec defines the properties and attributes of communication between components without necessarily defining the mechanism for achieving this communication.

3.2. Network Mapping

The first transformation needed is to map the application requirements to a set of network technologies that are supported by the underlying physical network. Different service providers and networks can choose different technologies for this implementation.

For example, if the underlying network utilizes VLANs and VPLS, then it could map each of the individual networks in a different VLAN, and it could utilize VPLS for interconnecting data center VLANs with enterprise VLANs. Alternatively, if the service provider utilizes some L2 over L3 technology such as proposed in VXLAN [DRAFT-VXLAN] or PBB tunneling over IP, and IPVPN, it could use a combination of these technologies to map the application requirements to a set of network primitives.

This transformation function takes as input the application requirements, the available network services, and the policy definitions, and creates a design of a composite network service that meets the application requirements. For the example instantiation of the 2-tier application using VLANs, the resulting network service is shown in Figure 3.

```
                     VLAN B
            -------------------------------------------
                  |     |          |         |     |
                  |     | Web Servers        |     |
                 +--+  +--+       +--+      +--+  +--+
                 |  |  |  |       |  |      |  |  |  |
                 +--+  +--+       +--+      +--+  +--+
                  |      |          |        |     |   Business
                                                      Logic Servers
     +-------+    |      |          |        |     |         +-----+
WAN--|Load   |------------------------      -----------------| PE  |
     |Balance|      VLAN A                      VLAN C       +-----+
     +-------+
```

Figure 3 Implementation of 2-tier service with VLANs

In this network mapping, the DC/cloud service provider has chosen to implement the load balancer as a virtual machine, and has chosen to interconnect application Tiers through VLANs. It has chosen an IPVPN connectivity to the enterprise back-end. At this stage, although the network service has been defined, the exact VLAN numbers or nodes implementing the services have not been identified yet.

Note, that when a service spans multiple administrative domains, it is possible that different technologies are used in each domain to meet the same application requirements. For example, an application that is split between a service provider data center and an enterprise data center might rely on IEEE SPB for layer-2 isolation within the service provider DC and VLAN based isolation within the enterprise data center. In another example, a WAN request for a guaranteed bandwidth path between data centers in different domains can be implemented as a wavelength service in one domain and as an MPLS service in another domain. Therefore, the service mapping must not only define how the network service is provided in each of the domains, but it must also identify the necessary mechanisms needed for stitching services between domains. Obviously, this might introduce a very large number of permutations, and therefore the SDNP work must concentrate on frameworks and specific use cases to limit the scope.

Note also that in multi-domain implementations, there is no single master SDN controller. Each administrative domain will have its own SDN controller. In these cases, communication between SDN controllers must be done at the abstract level of service requirements rather than by defining explicit network technologies. In this way, different administrative domains can seamlessly interface to serve such applications, even when they rely on different network technologies.

The next step in the process is the binding of the network service to specific network elements.

3.3. Network Service Binding

During this step of the operation, the composite network service must be mapped to particular network elements and topologies. At this level, the necessary resources (servers, virtual load balancers, virtual firewalls, etc.) are mapped to specific physical resources. The network service interconnecting these resources must be instantiated through a sequence of programming steps between the SDN controller and the individual physical or virtual network elements or network management systems.

There are several different methods that can be used to implement the binding process and a set of different protocols that can be used for communicating with the individual network elements or network management systems. Existing protocols include SNMP, Netconf, and even Command Line Interface (CLI). Alternative solutions might rely on network management tools and specific APIs that they expose. As Openflow matures, an SDNP controller could also program services in networks that utilize Openflow by communicating with the Openflow controller.

The binding process will usually require a series of steps that can include:

- o L2 and L3 topology discovery,
- o Path selection for each service
- o Traffic engineering for providing some form of SLAs.
- o Configuration of network elements for L2 and L3 services

Depending on the underlying technologies, some of the above steps can be omitted and can be performed with distributed control planes. For example establishing an MPLS path for communicating between data centers can utilize LDP or RSVP-TE and does not need explicit knowledge of the network topology or configuring every element in the path. On the other configuring VLANs on an Ethernet network might require explicit configuration of network elements.

Nevertheless, the important characteristic of the binding process is that it requires a method for interacting with control layer protocols or network and element management systems. Even though a set of such mechanisms could be available in networks today, it is unclear that the interfaces are powerful or generic enough to allow such dynamic programming and this is the main focus of the SDNP work. Bridging the gap between the controlled environment of today's networks and an application driven network configuration will require a set of access control mechanisms that will protect the underlying infrastructure.

3.4. Policy Definition

Every transformation step in the above description must be driven by policies that are administered either by the service provider, the IT organization requesting the service or the applications themselves.

It is such policies that will determine how network requirements are mapped to network design and how services are bound to specific network elements. The policy complexity will depend on the service

offered by a provider and can include security, billing, compliance, performance related policies, etc.

4. Examples of Service Binding

As it is evident by the above description a complete framework for application-driven network programming offers several layers of abstraction and the work around SDNP must carefully choose the layer of scope in order to solve specific problems. Addressing the whole architecture might prove a huge and challenging task that will limit the usefulness or impact timely completion of the work.

Clearly SDNP work must address the binding step that will enable configuration of network elements and control layer protocols in order to deliver the service. SDNP will also need to develop the interfaces between different administrative domains for services that span multiple domains. In the next Section we present such a control protocol configuration that will illustrate the scope of control plane programmability of SDNP.

4.1. Example: An end-to-end data center service

We consider the example presented in the previous Sections. We assume that the implementation is done over an IEEE 802.1aq SPBM network within the data center and an IPVPN service will connect the DC servers with other enterprise resources. The SDNP controller will utilize two different mechanisms to program the DC LAN and IPVPN services. The logical service mapping is illustrated in the following Figure 4, and it is similar to that of Figure 3, where each VLAN is now replaced by a different service ID (ISID):

```
                   ISID A
          -------------------------------------------
                |     |          |         |     |  Business Logic
                |     | Web Servers        |     |  Servers
               +--+  +--+       +--+      +--+  +--+
               |  |  |  |       |  |      |  |  |  |
               +--+  +--+       +--+      +--+  +--+
                |      |          |        |     |
   +--------+   |      |          |        |     |         +---------+
   |Load    |-----------------------      -----------------| PE      |
   |Balancer|      ISID B                      ISID C      | +---+   |
   +--------+                                              | |VRF|   |
                                                           | +---+   |
                                                           +---------+
```

Figure 4 Service instantiation based on SPBM

4.1.1. LAN Configuration

```
                   +-------+     +-------+
                   |       |     |       |
                   |B-BEB  `------B-BEB  |
                   |       |     |       |
                   .-'--|--+     +--|--`-.
                .-'     |           \     `.
              .'       /             \      `-.
           .-'         |              |        `.
     +--.-'--+     +---/---+     +----\--+    +--`-.--+
     |       |     |       |     |       |    |       |
     |I-BEB 1|     |I-BEB 2|     |I-BEB 3|    |I-BEB 4|
     |       |     |       |     |       |    |       |
     +---|---+     +-------+     +-------+    +-------+
         |             |             |             |
     +---|---+     +---|---+     +---|---+     +---|---+
     |       |     |       |     |       |     |       |
     |       |     |       |     |       |     |       |
     |Servers|     |Servers|     |Servers|     |Servers|
     |       |     |       |     |       |     |       |
     |  T1   |     |  T1   |     |  T2   |     |  T2   |
     +-------+     +-------+     +-------+     +-------+
```

Figure 5 SPBM based DC networks

We consider a DC network based on SPBM [IEEE802.1aq] (see Figure 5). Within the DC LAN, network isolation is provided by assigning virtual machines or servers of a given data center tenant to different Service IDs (ISIDs). A set of edge bridges (I-BEBs) encapsulate Ethernet frames using PBB encapsulation. The I-BEB function can be implemented either at the Top-of-Rack switch or at the hypervisor virtual switch. A set of B-Component bridges (B-BEB) interconnect the edge bridges. The I-BEBs will map traffic from each tenant to a particular service instance (I-SID/B-VID) depending on the service requirements. In the example of Figure 5, application tier T1 is associated with I-BEBs 1 and 2 and application tier T2 is associated with I-BEBs 3 and 4. The SPBM protocol provides the mechanisms for shortest path forwarding and learning in the backbone space without the need of deploying spanning trees or MAC learning. SPBM also provides the mechanisms for service auto-discovery and flood containment when a service covers a subset of service nodes. For example, if servers of a given Tier are associated with a subset of the I-BEBs, SPBM will create the multicast tree for flooding that will be associated with only this subset of I-BEBs. In order to instantiate a VM in a new node, the corresponding I-BEB needs to be provisioned, and once this provisioning is complete SPBM will expand

the multicast tree and include this I-BEB in the multicast distribution automatically. In the example of Figure 5, if a new VM from tier T1 is associated with I-BEB 4, the multicast tree associated with this I-SID/B-VID will be expanded to cover I-BEB4.

4.1.2. IPVPN Configuration

As shown in Figure 4, the enterprise service in the DC is connected to other enterprise data centers and services through an IPVPN [RFC 4364]. The first time that a business logic tier server is instantiated and ISID C is created, the SDNP controller must also provision the PE with the corresponding VRF. The SDNP controller provides the necessary information to the PE (incoming interface, ISID/B-VID pair, route targets and route distinguishers), and instructs the PE to instantiate the VRF. Once the VRF is instantiated, the existing routing protocol mechanisms (MP-BGP) will be used to propagate the routes to the other VRFs of the same IPVPN. Note, that in this case the SDNP controller does not define forwarding behavior, but it rather instantiates part of the control plane.

The scope of the SDNP configuration is limited to the edge PE and does not include any other provisioning or configuration in other routers in the network. This provides a simple interaction between the DC management system and network protocols, since it does not impose the requirement for the DC management system to understand the full network topology.

In more complex environments with separate DC and WAN PEs, this configuration might apply only to the DC PE, and the inter-connection between DC and WAN PEs can be done using any of the options of [RFC 4364].

4.2. Configuration Alternatives

The SDNP Controller can choose one of two methods for provisioning the necessary network functions.

4.2.1. Network Element based Configuration

In the above examples, the function of the SDNP controller is limited to provisioning the "edge" elements (I-BEB, PE) and a layer-2 or layer-3 protocol propagates the necessary information through the network (SPBM or MP-B respectively). In both cases, the SDNP controller has the ability to directly program functions in the corresponding network elements and does not need access to all network elements.

In addition to the programming interface, the SDNP controller must have an exact representation of the physical topology so that it can identify which I-BEB and PE must be provisioned (i.e. it is aware of the physical connectivity between I-BEBs and servers, L2 topology and PEs). The controller can discover this information by polling the I-BEBs and/or using LLDP between I-BEBs and servers. The controller can also listen to the SPBM protocol to capture topology information. Alternatively I-BEBs can proactively notify the SDNP controller about topology changes and the activation of new physical or virtual servers.

4.2.2. Network Management based Configuration

In an alternative implementation, the SDNP controller does not directly provision network elements, but it utilizes a network management tool that is already deployed in the network. Note, that in several cases, cloud and network operators have deployed network management and OSS systems and would want all operations to be driven by the these systems.

In this instance, the SDNP controller does not need to maintain detailed topology information, and it just talks to the corresponding network management system to issue the requests for I-BEB and PE provisioning.

5. SDN Model and Reference Architecture

Based on the discussion above, and using the simple example of Section 4 as guidance, we can develop the SDN reference architecture that can capture these requirements. Figure 6 outlines this architecture.

In this model, SDN Controllers expose an abstract API to applications, where they can request specific network properties such as bandwidth assignments, network isolation, routing properties, redundancy properties, security requirements, etc. Communication between different SDN Controllers enables these entities to provide services across network administrative domains without being constrained by underlying technologies. The interface is limited to communicating the requirements of the application. Each SDN Controller can translate application requirements to different technologies based on the underlying network implementations. This approach provides the maximum portability for applications and does not burden application developers with network technology specifics.

```
                  | Application API             |

                  |           Spec              |
         +------------------+ API     ------------------+    +------+
         |SDN Controller 1  |--------|SDN Controller N  |    |  P   |
         |                  |        |                  |    |  O   |
         +------------------+        +------------------+    |  L   |
                 | Service API                 |             |  I   |
                 |                             |             |  C   |
         +------------------+ Intf   +------------------+    |  Y   |
         |Network Specific  | API    |Network Specific  |    |      |
         |Module            |--------|Module            |    |      |
         +------------------+        +------------------+    +------+
                 | Network API                 |
                 |                             |
             +----------+                  +---------+
             |Plug-in   |                  |Plug-in  |
             |Rest API  |                  |Rest API |
             +----------+                  +---------+
                 |                            |
                 |                            |
         +-------+----------+                 |
         |Network Management|                 |
         |                  |                 |
         +-------+----------+                 |
                 |                            |
                 |                            |
         +-------+----------+        +--------+---------+
         |Network           |        |Network           |
         |Control Plane     |        |Control Plane     |
         +------------------+        +------------------+
```

Figure 6 SNDP reference architecture

As it relates to the framework of Figure 1, the SDN controller implements the Network Mapping function and it includes the policy specification functions. The SDN controllers utilize a service API to communicate these requests with a network technology specific module that will be responsible for the translation of the requirements to specific technology primitives. An Interface API will enable network specific modules to stitch services together. This is a technology dependent API and it will enable functions such as mapping a VLAN to VPLS domain, or mapping an LSP to a wavelength and so on.

The network specific module is essentially implementing the network binding function of Figure 1. For every network technology, there is

a specific API to the control plane of network elements that will allow the binding and instantiation of the network service. Note, that in several environments it is possible that the network mapping is communicated with a pre-existing network management system and not directly with the control plane of network elements. This not only allows the evolution from current operational models, but it also enables the concurrent support of existing and future operational models. If for example an OSS is currently driving the deployment of network services through a network management system, such an approach would allow new applications to utilize the same infrastructure and provide an evolutionary path to a software defined network.

Note, that the network specific module might also include functionality related to topology discovery. The type of topology information required will depend on the underlying technology and it can range from a full network map in an Openflow environment to a limited resource view when the final binding operation is performed by a network management system.

5.1. Scope and Approach

Given the number of different networking technologies and implementations, defining all possible APIs within the scope of a single working group will be hard to tackle. The SDNP work should therefore concentrate on designing the framework, application network requirements schema, and API control and communication architecture, as well as defining the reference schema and API around a single networking technology as an example use case. Once the framework is in place, the technology dependent functions can utilize it to develop their own specific APIs and this can be done across multiple working groups.

6. Security Considerations

The SDNP controller security aspects must be addressed, including functions such as role based authentication and security of intra-SDNP controller communications.

7. IANA Considerations

IANA does not need to take any action for this draft.

8. Conclusions

This document has introduced a generic framework for mapping application requirements to specific network services within the context of Software Driven Networks. The document also outlined a reference framework as input to the definition of the scope of the SDNP work.

9. References

9.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

9.2. Informative References

[IEEE802.1aq] IEEE 802.1aq Shortest Path Bridging

[RFC 4364] E. Rosen et.al, BGP/MPLS IP Virtual Private Networks, RFC 4364, February 2006.

[DRAFT-VXLAN] Mhalingham et.al., "VXLAN: A framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks,", draft-mahalingham-dutt-dcops-vxlan-00.txt, September 2011.

10. Acknowledgments

This document was prepared using 2-Word-v2.0.template.dot. We also want to thank T. Sridhar for early comments on the draft.

Authors' Addresses

Dimitri Stiliadis
Alcatel-Lucent
777 E. Middlefield Rd
Mountain View, CA 94043

Email: dimitri.stiliadis@alcatel-lucent.com

Florin Balus
Alcatel-Lucent
777 E. Middlefield Rd
Mountain View, CA 94043

Email: florin.balus@alcatel-lucent.com

Wim Henderickx
Copernicuslaan 50
2018 Antwerp, Belgium

Email: wim.henderickx@alcatel-lucent.be

Nabil Bitar
Verizon
40 Sylvan Road
Waltham, MA 02145

E-mail: nabil.bitar@verizon.com

Mircea Pisica
BT
Telecomlaan 9
Brussels 1831, Belgium

Email: mircea.pisica@bt.com