Chapter 14: Buyers guide - What to look for

SDN being a relatively new technology and with the lack of standardization, business executives and managers who need to make choices and selections for their business find it hard to differentiate one offering from the other. In this section, we attempt to breakdown the necessary components of any SDN solution so that readers can ask the right questions to make the right choices for their networking needs.

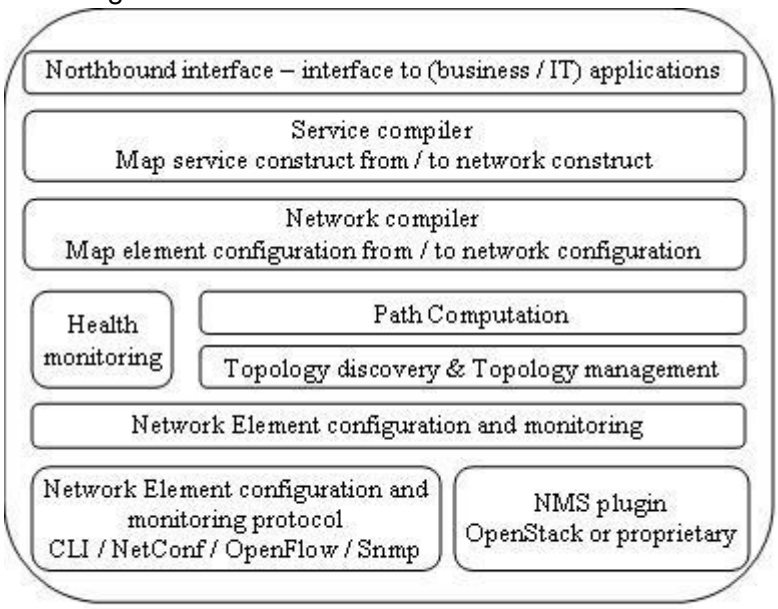

Figure 14: SDN control components

Northbound interface. This is the interface that the controller exposes to applications / business applications. Common methods for this interface are REST APIs or XML. This interface allows business or IT applications to directly make service requests to the controller.

Service compiler. The controller needs to provide a translation from service constructs (example – requirement for a redundant leased line) to a network construct (example – 1:1 bidirectional redundant EVC – Ethernet Virtual Circuit). This hides the network type from the service and allows an operator to make network upgrades seamlessly.

Network compiler. The controller needs to take in the existing topology information of the network in terms of nodes and connections, and must transform this into a network level construct. The controller must also provide the translation from a configuration required in the network to the configuration that must be applied in individual nodes to realize the network construct.

OAM (Operations, Administration and Maintenance). The controller must have a view of the status of every link and node in the network complete with fault information, performance information and connectivity information. Optionally, this can be fetched from the existing NMS if the NMS already this view. This is also sometimes variably referred to as Health Monitoring.

Path computation. This module computes the path from one end of a service to another, through the network. The most commonly used set of algorithms for path computation are the Dijkstra algorithm with its many variants. It is expected that with the path computation moving into servers, there will be a lot of innovation in this area. For this reason, the controller must allow a user to be able to plug-in proprietary algorithm implementations.

Topology discovery and topology management. The controller mist be able to learn the topology of an existing network when connected into it. The controller must keep itself updated of any changes in the topology. There are various possibilities for this, the most common being the use of LLDP for topology discovery and updates. Other protocols such as EOAM, LMP, etc may also be used since these have methods for nodes at 2 ends of a link to find each other.

Network element configuration and monitoring. The controller must convert the result of path computations into configurations to be applied at each network element. This configuration can either be dispatched to the NMS from where the configuration is actually applied to the network elements, or the controller can directly apply the configuration to the network elements. Any notifications and state changes from the network elements must be updated to the controller (either directly or via the NMS) so that the controller is able to re-compute and re-apply service constructs where necessary.

Network element configuration and monitoring protocol – CLI / NetConf / OpenFlow / SNMP. The controller must dispatch the configuration for each network element to the network element through the management protocol that the network element supports. The traditional options of CLI or SNMP can still be used where devices do not support OpenFlow or NetConf. NetConf has been recommended by the ONF as the preferred way for telecom networks.

NMS (or other management system) plug-in. Especially in telecom networks, since the NMS interoperability with the network elements has been established and field proven, operators may want to continue managing the network elements through the installed NMS. The controller will then have to work with the NMS or be integrated into the NMS as an additional module with a plug-in.

There are some other critical attributes required from a controller which operators and OEMs with a networking background will be familiar with.

Distributed and hierarchical functioning. For small or simple networks, a single controller may be sufficient for the computes required. For larger networks, comprising 10s of 1000s of endpoints, and maybe 100s or 1000s of switches, a single physical node will be unable to satisfy the requirements for computation and performance. So, controllers must be designed for distributed processing with the load being shared across multiple servers / nodes. The network could be segmented into some logical partition with each partition being controlled by a controller. These controllers of parts of the network could then be controlled by a next higher level controller giving rise to the concept of hierarchical controllers. Standards for this are not

yet even being debated leave alone being defined though some products claim to support this in proprietary ways.

High availability. Since the availability and integrity of the network is completely dependent on the controller, the controller is a single point of failure, redundancy and availability needs are stringent even for the smallest networks. Quite often, load sharing and high availability could be built into controllers in a combined way so that 2 controllers can load share and provide redundancy for each other in small networks.

Security. The controller is a single point of vulnerability for the entire network. If the controller is compromised, the entire network is compromised. Malicious attacks could be concentrated on the controller instead of trying to find weak points in a network of devices. Thus, it is both easier to attack a SDN network and easier to safeguard it since attacks and protection schemes could both be focused on the controller alone.