NEC ProgrammableFlow:
Redefining Cloud Network Virtualization with OpenFlow
Table of Contents

The Place of Network in Cloud Computing.................................................................................................................................................3
The Challenges in Cloud Networks..............................................................................................................................................................3
Network Virtualization - Redefined..............................................................................................................................................................4
Introducing NEC ProgrammableFlow Network............................................................................................................................................5
ProgrammableFlow Network Architecture...................................................................................................................................................6
ProgrammableFlow Network: Core Advantages..........................................................................................................................................9
Conclusions................................................................................................................................................................................................10
The Place of Network in Cloud Computing

Cloud computing has ushered the IT enterprise and service providers into a new era that redefines how IT resources and services are delivered and consumed. With Cloud computing, disparate and distributed physical resources such as computing and storage are pooled together for on-demand instantiation and self-service delivery with elastic scaling of resources. The goal is to have a drastic reduction in the total cost of operation. To that end, resource virtualization is the key technology enabling the cloud computing model. Virtualization allows creation of a logical abstraction layer above the physical resource thereby enabling a programmatic approach to allocate resource while hiding the complexities of managing them. The result is efficient resource utilization, simple manageability, on-demand and programmatic resource instantiation, and resource isolation for better control and availability.

In a cloud environment, the network is a critical resource that connects various distributed and virtualized components - servers, storage elements, appliances and applications. For example, it is the network that allows aggregation of physical servers, efficient virtual machine migration, and remote connection to storage systems, effectively creating the perception of large infrastructure resource pool. Furthermore, it is also the same network that enables delivery of cloud based applications to end users. Yet, while every component in a cloud is getting virtualized, the physical network connecting these components is not.

Without virtualization, the network is one physical common network, shared by all cloud end-users and cloud components. Without virtualization, the network likely becomes a single complex system in the cloud as the cloud evolves to provide new services with diverse requirements while trying to sustain the scale.

The Challenges in Cloud Networks

Today’s network is made to behave like “one size fits all” architecture in meeting the diverse requirements of a cloud. The security policies, data forwarding protocols, and underlying network topology are all designed looking at the sum of all requirements, and does not allow optimal usage and management of the network.

Topology-dependent complexity

Every data center network topology is tuned to match the traffic requirement. For example, a network topology optimized for East-West traffic (among servers in a data center) is not the same as the topology optimized for North-South traffic (traffic to/from Internet). The topology design goals are often conflicting. The topology design also depends upon how the Layer 2 (L2) or Layer 3 (L3) layer is going to use the effective network capacity, thereby creating a cross dependency with the forwarding protocols in use. For example, adding a simple link and switch may not provide additional capacity using a spanning tree based L2 forwarding protocol. The use of more inter-switch LAGs versus multiple spanning trees for a L2 network versus ECMP for L3 network can all lead to different topology based designs, thereby adding to the complexities. Furthermore, evolving the topology based on traffic pattern changes also requires complex configurations of L2 and L3 forwarding rules.

Multi-layered network complexity

A typical data center network is composed of multiple layers, where each layer provides certain specific roles. An example three layer scenario includes TOR (top-of-the-rack) layer connecting the servers in a rack, Aggregation layer connecting multiple Racks, and a Routing layer providing connectivity to/from the Internet edge. Layers can operate using L2 or L3 forwarding protocols. Layering imposes significant complexities in defining boundaries of L2 domains, in defining the L3 forwarding network and forwarding policies, and in dealing with layer-specific multi-vendor equipment.
Location-dependent constraints

Servers and other network appliances are typically tied to a statically configured physical network, which in turn implicitly creates a sense of location constraint. For example, a server IP address is determined by the VLAN or subnet it belongs to. VLAN and subnets are based on physical switch port configurations. Consequently, a virtual machine cannot be easily moved across the network because it will require changes in the VLAN and/or subnet configurations. Such static location binding puts constraints on VM migration and reduces the level of resource utilization. Additionally, physical mapping of VLAN or subnet space to the physical ports of a switch/router often leads to fragmentation of the resource pool, for example the IP address space.

Policy enforcement complexities

A network also has to enable multiple forwarding policies, such as access control to the end-users or traffic isolation. Such policies need to be flexible to build a private cloud. Policies directly impact the configurations of each router and switch. Changing requirements make it difficult to configure and manage such policies.

In addition to the above complexities, different protocols such as OSPF, LAG, VRRP, various flavors of spanning tree protocols at L2, along with other vendor-specific protocols makes it a challenging task to build, operate and inter-operate a network of scale.

In summary, the resultant burden of design, configuration, and operation management complexity often limits the scale of network growth, increases the time to deliver services, and increases the total operation cost.

Due to these complexities, the cloud environment is able to provide servers, storage and even network appliances (load balancers) as Infrastructure-as-a-Service (IaaS), but not Network-as-a-Service. Yet, Network-as-a-service is going to become a critical necessity for enterprises trying to have their own customized private virtual cloud as well as for operators to build and manage custom networks while meeting changing requirements.

Network Virtualization - Redefined

The current legacy network does provide a certain level of virtualization. They can be categorized into two types: Node level virtualization and End-to-end path level virtualization. An example of node level virtualization is vSwitch from VMWare where an L2 virtual network can be defined at a physical host. Yet another example of node level virtualization is the Virtual Router Framework (VRF) which enables creation of multiple virtual router instances on a physical switch. VLAN can also be considered as a node level virtualization where single switch/ports can become a member of multiple L2 networks. Path level virtualization is seen in creation of GRE or L2TP tunnels which provides a virtual pipe between two points.

Limitation of node and path level virtualization

While node level virtualization does have its advantages, it still requires handling the configuration and management complexity at node/port level. Path level virtualization helps in cases where the end-to-end traffic pattern is known a priori and requires point-to-point virtual pipes. However, for any-to-any connectivity as observed in today’s cloud environment, managing and creating large number of virtual pipes limits the scale. Furthermore, such virtual pipes cannot always ensure full usage of the network backplane.

Introducing network level virtualization

NEC ProgrammableFlow takes the first step in offering a complete network level virtualization solution where the entire physical network is virtualized end-to-end. A network level virtualization completely hides the underlying physical network, while exposing a simple virtual network that is, easy to manage, configure and operate.
Network level virtualization is inspired by the same principles that are used to virtualize other infrastructure resources such as physical machines and storage systems. This paradigm defines a virtualization layer that pools disparate network elements (switches, routers) into creating one or more virtual network instances. By principle, the virtual network must at least provide the same features as available from a physical network. Following the same analogy as server virtualization, the ProgrammableFlow network virtualization approach enables the following key features:

- Virtual network topology decoupled from the actual physical network topology
- Ability to create flexible virtual networks to meet diverse requirements
- Ability to have multiple virtual networks co-existing on top of shared physical networks.
- Support isolation ensuring virtual networks can operate independently
- Programmatic approach to dynamically create/deploy virtual networks
- Enabling better physical network resource utilization through network resource pooling
- Reduce network design and (re) configuration time and complexities

**Introducing NEC ProgrammableFlow Network**

NEC’s ProgrammableFlow (P-Flow) Network is a network virtualization solution where multiple interconnected switches can be virtualized into a large common network resource pool with unified and dynamic control on the switch forwarding behavior. The aggregated and virtualized resource pool acts like a big switch providing layer-2 to layer-4 functionalities. The P-Flow network solution is based on two core principles: a) Decoupling the control plane from the data forwarding plane and b) Separation of the logical network plane design from the physical network plane operation.

**P-Flow Network control and forwarding plane**

In a legacy network, switches and routers participate in implementing the control plane logic. For example, a simple L2 switch implements multiple control plane protocols and functionalities such as Address Resolution Protocol (ARP) and Spanning Tree Protocol (STP). With some of these protocols implementation is vendor specific, which limits multi-vendor interoperability.

In a P-Flow network, the control plane functionalities are implemented in a server/appliance based (remote) controller box, limiting the network hardware/switches to only basic data forwarding functionalities. The resultant physical hardware infrastructure becomes uniform in terms of interface and functionalities enabling simple operation management. The software defined controller logic hosted in the controller box leverages OpenFlow switch control protocol (www.OpenFlow.org) to define the packet forwarding behavior and control.

OpenFlow provides an open protocol to dynamically program flow entries, into a switch. OpenFlow also defines the packet forwarding functionalities in an OpenFlow capable switch. Specifically, there exist three basic functionality definitions: a) Rules to match incoming packets based on the subset of 12-tuple packet header fields; b) forwarding actions such as egress port to be forwarded and c) advanced flow-level statistics collected from switches. More information about OpenFlow protocols can be found at www.OpenFlow.org.

By supporting OpenFlow control protocol, NEC P-Flow network control plane logic can be applied to a network comprised of OpenFlow supported switches from any vendor. Software-defined control plane logic framework in a P-Flow network allows support for existing legacy features as well as enabling richer functionalities to accommodate future feature requirements. The centrally defined control plane logic is simpler to manage than to individually manage a complex interconnection of switches/ports and their individual packet forwarding behavior.
Separation of logical plane from physical plane

Logical abstraction plane captures the high level network functionality requirements. An example of such a requirement is having a set of servers belonging to a given subnet sharing a L2 bridged domain. Implementing the functionality in the current legacy system requires an understanding of the physical topology of the switches connecting the servers. Necessary configurations, such as spanning tree and VLAN configurations need to be applied to each participating switch to ensure such L2 connectivity.

The above functionality requirements can be translate to a simple logical plane design - connecting all servers to a L2 bridge. However, in an existing legacy network, the logical plane design is deeply integrated with the underlying physical network with constraints to inter-switch/router protocols and specific switch/router configurations.

The uniqueness of the P-Flow network design is in exposing a logical abstraction plane called the Virtual Tenant Network (VTN) framework, which enables deployment of a logical plane on top of any underlying physical network topology. Specifically, P-Flow VTN framework can automatically map a logical plane design into underlying physical network functionalities. These physical network functionalities are dynamically enabled by creating or modifying flow rules at individual switches leveraging the OpenFlow control protocol. The logical plane definition not only hides the complexity of the underlying physical network but also provides better management of network resources, achieving significant reduction in time to reconfigure network services and minimizing network configuration errors.

**ProgrammableFlow Network Architecture**

The ProgrammableFlow network architecture consists of three basic parts: A) P-Flow Controller and B) OpenFlow control protocol and C) OpenFlow enabled switches. The architecture diagram is shown below.
As shown in the diagram, the NEC OpenFlow enabled P-Flow switches or any 3rd Party OpenFlow enabled switches can be connected in any topology to create the data forwarding plane. There exists an out-of-band secure communication channel connecting each switch with the P-Flow controller. The P-Flow controller is deployed with full redundancy using standard clustering techniques. The out-of-band network and redundant controller eliminates a single point of failure in the system. Additionally, for large scale operations, multiple distributed controllers can be deployed while providing seamless control and management of all switches across the data center.

The controller communicates with each switch to set-up a forwarding table as and when required using the OpenFlow control protocol. In typical scenarios, the forwarding table at a switch is updated either in a proactive or reactive manner. In the case of proactive updates, the controller pre-populates the flow tables in the switch from high level functionality requirements. In the case of a reactive update, the controller updates the flow tables only when packets for a new flow are received by the controller. Therefore, the logic at the controller determines the right forwarding table entries at each individual switch, to ensure a consistent end-to-end connectivity as defined by the requirements. The above design essentially alleviates the need for defining complex logic at the switch, and avoids the need to repeat the same logic at every switch, drastically reducing the configurations required at each switch.

**ProgrammableFlow virtual networks**

**Core approach**

P-Flow virtual networks allow network operators to construct custom networks to meet their functionality requirements while being agnostic about the underlying physical network. The virtual network framework eliminates the need for (re)configuration of physical network devices. The virtual network framework exposes a rich set of constructs to the operator to enable network-wide programming, as opposed to device level configurations. The framework allows the creation of on-demand networking where a completely new virtual network can be created and deployed on-the-fly. The virtual network plane also allows the use of the OpenFlow based control by automatically mapping the end-to-end network requirements to individual switch level flow entries.

**Network resource pooling**

One of the common concepts across any virtualization paradigm is resource pooling where diverse distributed resources (eg. Servers or storage) can be unified or aggregated into a logically defined large pool for efficient allocation and management of resources. It is difficult to apply the same concept to networks as network resources (switch, port, links) are tightly coupled through complex protocols along with different roles (L3 routing, L2 switching). In contrast, OpenFlow defines identical behavior for all switches and a common interface. Leveraging the above characteristics, P-Flow virtual network framework allows network resource pooling by creating one big switch perception, effectively simplifying management and improving resource allocation efficiency.

**Virtual network constructs**

The VTN framework provides a rich set of constructs to create virtual networks that can meet simple to complex network solution requirements. In order to create a basic L2 and L3 network, P-Flow VTN framework provides vBridge, vRouter and vLink constructs. In order to connect the network to end points, a VTN framework defines vExternals, which can refer to a legacy router/switch or physical server/virtual machines. A simple example of L2/L3 network is shown in the diagram below.
A P-Flow VTN framework also provides advanced constructs that can help build intelligent flow based networks. Examples of such constructs are vFilter and vRedirect - vFilter defines matching sequences on packet header fields and associated actions such as pass or deny. vFilter can effectively implement any L2-L4 based ACLs. vRedirect allows creation of an explicit forwarding path from a virtual interface of vBridge or vRouter to vExternals. vRedirect works in conjunction with vFilter where a flow is first matched based on vFilter and all packets are directly forwarded to the switch port which connects to a given vExternal.

**Virtual Tenant Networks (VTN)**

The ProgrammableFlow VTN framework allows multiple virtual networks to co-exist on top of one physical network. Each virtual network can have its own network topology and policies. From an operator standpoint, each virtual network operates independently with complete traffic isolation. In other words, packets can be forwarded from one virtual network to another. Co-existence of multiple virtual networks significantly helps in network operation. For example, one can design, deploy or reconfigure a given VTN independently. Such a framework also allows staging and testing of a network without interfering with an operational network. VTN also allows a multi-tenancy model to be extended to a network, where a tenant can define his own network – an important requirement in building a private cloud. The diagram below illustrates the VTN deployment on top of physical network.

**ProgrammableFlow controller architecture**

The ProgrammableFlow controller architecture enabling network virtualization feature is shown in the diagram below. The controller system consists of PFC-Core, which provides key services such as Topology manager, Path mapper, Flow mapper and communication control with OpenFlow enabled switches. A topology manager maintains the current state of the network by continuously monitoring the switches and links. The Path mapper computes the end-to-end paths based on the VTNs and maintains path level states. Flow mapper maps the path information to switch level flow entries, which then gets communicated to the OpenFlow enabled switches.
On top of the network core is the VTN plane, which provides network operators or designers with the model and interface to create, edit and deploy virtual networks using the in-built constructs. The VTN plane provides a framework for VTN creation and management using a CLI and script based programming, APIs for creating virtual networks. VTN plane translates the VTN definitions into corresponding network states and deploys the virtual network on a physical network through the network core.

The ProgrammableFlow controller also provides a Network Management System with a rich user interface for virtual network configuration, management, and visualization. Network Management user interface provides automatic visualization of the virtual and physical network topology, tracking of flow paths and traffic information.

The controller exposes APIs for third-parties to develop new solutions leveraging the controller features. APIs can also be used for exporting network state information to third party applications.

### ProgrammableFlow Network: Core Advantages

ProgrammableFlow network solution and architecture make a radical jump in terms of how networks can be deployed, operated and utilized for performance. Some of the key advantages are as follows:

**Simplicity of management:** By decoupling the control layer, P-Flow provides one uniform control to an entire network of switches. A single point of management reduces the complexity of managing individual switches. The VTN framework further simplifies the network deployment by exposing a simple logical network model. VTN model based deployments and operations also ensure network wide consistency while preventing individual switch level misconfigurations.

**Compaction of network layers:** As opposed to legacy networks, ProgrammableFlow network can provide L2, L3 and selected L4 features along with intelligent ACLs in the same switch. Furthermore, the P-Flow network can behave like a single big switch instance in integrating L4-L7 appliances. This feature makes the network more homogenous as opposed to multi-layered resulting in significant cost and complexity reduction and performance improvement.

**Performance:** ProgrammableFlow virtualization allows automatic computation and deployment of multiple paths for flows through the network. The flow traffic is dynamically distributed among the multiple paths based on policies. Use of multiple paths significantly adds to the end-to-end bandwidth performance without adding a protocol level complexities or constraints. For example, the use of P-Flow multiple paths policies can be seamlessly applied to a layer-2 as well as layer-3 network circumventing any constraints imposed by spanning tree protocol in legacy networks or the need to define yet more protocols such as TRILL or Shortest path bridging.

**Reliability:** ProgrammableFlow provides automatic end-to-end reliability where traffic is routed through alternate paths in the event of link or node failure. By having continuous detection of network topology and a decoupled control plane, the switching to an alternate path happens fast, requiring nearly zero convergence time.

**Adaptive scalability:** ProgrammableFlow provides a logical or virtual plane abstraction which is independent of the physical topology. This feature allows network architects and operators to independently scale up and down the physical network interconnect without having to worry about breaking the existing functionalities. For example, one can add physical switches and links without changing the virtual network. ProgrammableFlow automatically detects the updated physical network and maps the traffic on the newly modified network while maximizing the network utilization.

**Service deployment time:** ProgrammableFlow significantly reduces service deployment time by automatically translating simple virtual network designs to complex physical network configurations. The deployment and reconfiguration of the network is almost instantaneous. The virtual network model provides several implicit network logic validations, reducing the time to test and debug.
ProgrammableFlow: Redefining Cloud Network Virtualization with OpenFlow

**Service interruption time:** The isolation of the virtual tenant networks in a ProgrammableFlow network allows staging of network for testing and verification without affecting existing operational network. Furthermore, a ProgrammableFlow network allows maintenance scheduling where a given switch can go through upgrade maintenance without affecting end-to-end service. The network automatically reroutes flows to support such maintenance activities. The above features enable operators to provide high availability.

**Programmability:** The virtual network framework provides a programmable approach to define networks, using the rich virtual network constructs. The ProgrammableFlow controller provides a simple scripting language and CLI interface to define and construct custom virtual networks. The vFilters and vRedirects in combination provide flexible programmatic ways to create intelligent solutions.

**Monitoring and visualization:** One of the key benefits of an OpenFlow based P-Flow network is that it enables a single point view of the entire distributed network. By continuously monitoring the switches as well as collecting statistic logs from each switch, the P-Flow controller intelligently maps this real-time information into useful visualization information. For example, network operators can view the entire network topology, port-wise interconnections, real-time flow path on the actual physical network and much more. Furthermore, the PFlow network can provide dynamic flow-level monitoring compared to existing legacy based port or VLAN level monitoring. Operators have the flexibility to define flows and acquire flow specific information. For example, operators can track, account and monitor application (storage, compute) specific flows. For a network operator, it eases the overall management of a complex physical network.

**Conclusions**

NEC ProgrammableFlow takes a first step toward end-to-end network-level virtualization with the potential to make a radical impact on how cloud networks are operated, managed and scaled. ProgrammableFlow leverages OpenFlow based open control protocols to support multi-vendor switches, while creating topology independent virtual networks. Having a homogenous single layer network allows efficient network resource pooling yielding better capacity utilization and higher availability.

The unique virtual tenant network plane in the ProgrammableFlow controller can translate simple solution requirements to complex physical network configuration in an automated and dynamic way - reducing complexity and deployment time. Finally, network virtualization based on direct control on forwarding plane of switches enables new intelligent features related to policies and routing beyond what is available in legacy networks.