Volume 6, Issue 10, October 2016 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Dynamic Service Function Chaining of Network Functions Using SDN

Vineetha B
PG Student, Department of CS&E
CIT, Gubbi, Karnataka, India

Dr.Shantala C P*
Professor, Department of CS&E
CIT, Gubbi, Karnataka, India

Abstract: ***Since the network infrastructure of an organization is growing, it's very complex to manage and control such systems from a central-based control system like laptop through programs. For this purpose, here the new technology called SDN is used to manage and control the system programmatically. The network services are provided based on user requirements. SDN (Software Defined Networking) is an emerging technology in which network decouples into data and control plane. Data plane whose main responsible is just to forward the data and the control plane whose task is to make decision like how data should be forwarded. A group called Open planes require some protocol. An example for such a protocol is OpenFlow. The primary standard interface intended for SDN is OpenFlow. The main advantage of this is high performance, controlling data flow across different vendor's system devices. The project mainly concentrates on providing different network services to the client in a single system so that he/she can use specific services based on their requirements. The different services are Load balancer, Firewall, Video optimizer, Intrusion Detection and Prevention etc. This can be accomplished via "Service Function Chaining" by means of SDN.***

Keywords: Software Defined Network, Open Flow, Mininet, Service Function Chaining.

I. INTRODUCTION

Network is a collection of two or more computers linked collectively. The networks are classified into three categories as LAN, WAN, MAN. From past decades networking principles remained unchanged [1]. Switches and routers are used to assemble the networks. Devices are developed by number of vendors commonly using proprietary operating system and interfaces. An institution has to appoint a specialist on every router brand for building a heterogeneous network on devices from distant vendors. Because of this probability of configuration mistakes also increases while configuring different systems.

Along with this now a day's maintain and managing of the network systems are becoming more complex and also controlling a network by manually becoming more difficult because of increasable amount of growth in network infrastructure, and also recovering disasters is more complex. Because have to go to that network node and to solve that problem by vender specific solutions, and debug that error, so to overcome this network complexities. A new technology called software defined networking is used. SDN is innovative approach to designing, building and managing network, provides flexibility to control a network.

SDN consists of two planes namely control plane and data plane. Here entire network is controlled by program which is in control plane. It enhances elasticity; thus network provides fast disaster recovery. This technology uses some of the approach or technique for the interaction among control plane and data plane. The method is called as OpenFlow [2] protocols. The Controller used is OpenDayLight which is a written in java. For emulating network system via emulator called Mininet, for creating virtual network.

Software Defined networking is the latest technology which decouples data and control plane of a network. Switches are used for fundamental packet forwarding consists of flow tables populated with the localized flow rules [2]. Rule indicates how incoming packets are handled based on the matching fields. These are managed by the remote "controller" in the SDN. Control plane indicates how the packets should be moved. Communication between controller and the switches will be in a secure manner by means of a standard and open interface like OpenFlow protocol. It consists of inner flow table along with a consistent boundary to append or eliminate flow inputs [2]. The architecture allows for a variety of significantly more flexible and efficient network management solutions. A logically centralized controller provides application developers with a combined programmable interface on which to deploy software and higher level of application. It moves towards for providing flexible network programmability. It provides valid time organization, procedure and monitoring of a network.

SDN encompass of three layers and their interactions are shown in figure 1. If there are significant and wide-area region network, next we may use more than one SDN controller [3]. Through network policies control layer always balances the network states in either a distributed or centralized manner. Because of dynamic flow of activities the network policies should be efficient timely, for the unlimited right to use the universal network elements and resources. The SDN applications are present in the application layer of the SDN. The interactions among the management level as well as control level are supported by a group of APIs such as northbound open APIs. Forwarding packets in the data

© 2016, IJARCSSE All Rights Reserved

layer can be employed by software-based OpenFlow switches via OpenFlow Controller as well as the interaction among the switches and the Controller via southbound APIs.

The interaction between the layers, the SDN permits a whole and universal view of the network, as well as also includes a controlling the whole network in the control plane for the management of more traffic flows. Simplification of network management, reducing of operating costs, support modernization and development in recent and upcoming network are all will be done in the SDN.

The layered analysis of SDN architecture is demonstrated in the figure 1. As shown there are three different planes:

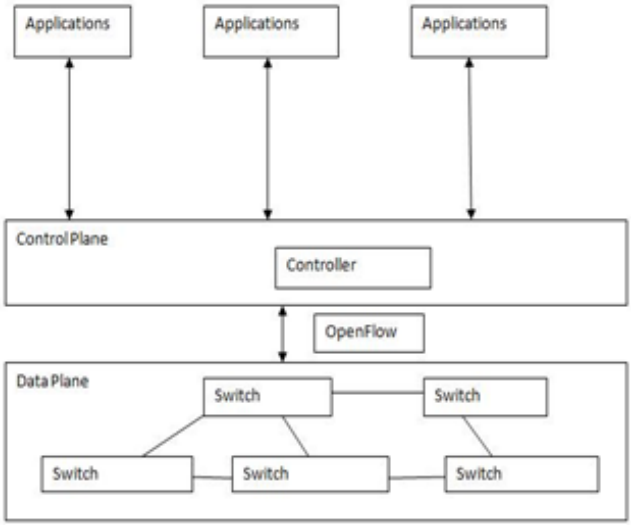

Figure 1: SDN Architecture

Forwarding Plane

It comprises the network devices or forwarding devices like switches, routers, hosts etc. In SDN network devices are acting because just forwarding hardware available during open interface at an abstraction level, the switches are used for this type in SDN. In the network, the OpenFlow switches are of two types. The Pure switches consist of no inheritance features and entire depend resting on controller for making forwarding decision. The second type of switches called Hybrid, which maintains OpenFlow along with traditional operation and protocols.

Control Plane

It includes a centrally based SDN Controller (NOS) that controls an overall analysis of the system that takes requests through APIs from application plane and performs dissimilar operation and monitoring the network devices using standard rules. Network Operating System (NOS) collects physical network state information distributed across all domains and as well it empowers the SDN to present an indication of the physical system state to a case of the control application. The network is programmable through software applications running on top of the NOS that communicate with the fundamental data plane devices.

Table 1: Summary of Controllers

Controller	Open Source	Language	Multithreaded	GUI	Origin
NOX[13]	yes	C++/Python	no	yes	Nicira Networks
NOX-MT[14]	yes	C++	yes	no	Nicira Networks and Big Switch Networks
POX [15]	yes	Python	-	yes	Nicira Networks
Maestro[16]	yes	Java	yes	no	Rice University
Beacon [17]	yes	Java	yes	yes	Stanford University
SCAN [18]	yes	C++/Python	no	yes	Nicira Networks
RISE [19]	yes	C and Ruby	Non-guaranteed	no	NEC
Floodlight [20]	yes	Java	-	yes	Big Switch Networks
McNettle [21]	yes	Nettle/Haskell	no	no	Yale University
MUL [22]	yes	C	yes	yes	KulCloud
RYU [23]	yes	Python	-	-	NTT OSRG and VA Linux
OpenDaylight[24]	yes	Java	yes	yes	Multiple contributors

© 2016, IJARCSSE All Rights Reserved

Application Plane

Include arrangements that importance on the development of system administrations. These arrangements are mainly programming applications that correspond with the controller.

Northbound and Southbound API's

The architecture of SDN consists of northbound APIs whose main task is to interact linking the Controller and managing applications running on the system. The northbound APIs are mainly help to provide new application with allow computerization of the network to organize with the requirements of dissimilar applications by means of SDN control plane programmability.

The communication between the Controller and the elements in the data layer called switches and routers are via southbound APIs. Southbound APIs provides well-organized management more over the network and enables the Controller in the control plane to dynamically make changes related to current requirements as well as demands.

SDN [4] is an emerging technique for management, configuration and operation of networks. The change in the architecture is revolutionizing the management of large scale enterprise networks, data center networks which requires dynamic changes many times a day. The SDN is adopted in the corporate world because of its impact the management of control system networks. It allows programmatic change control platform, which allows whole network to be managed and controlled by single assets, simplifies the understanding network and easily, continuous monitoring can be done. The control system flows are more consistent and continuous than ever-changing nature of corporate network flow. The architecture is able to optimize both machine-to-machine communication as well as people-to-machine communication. It is decoupling of the system that decides where the traffic is sent from the systems that performs the forwarding of the traffic in the network.

The traditional network deployment process starts with designing the topology, configuring the network devices and at last setting up the required network services. To achieve the optimal usage of network resources, all application data must flow in the direction of routes determining by switching protocols and routers. Network administrators need to reconfigure network to avoid loops, gain route convergence speed and prioritize a certain class of applications.

Benefits of SDN for enterprises are:

- ✓ It provides the services like speed and agility: By creating virtual machines, SDN can be simply set up in the network. Comparing to old physical network, SDN can be set up is a distant better achievement to virtual machine [6].
- ✓ Gives elasticity to network: Network experimentation with no impact" is enabled by SDNs. That means anybody can leap over the limits imposed by SNMP. By latest network configuration, experiments can be liberally done without constrained by their consequences.
- ✓ Provides enhanced security: SDN provides fine grained security for applications, endpoints. Those not known by the conservative hard-wired network.
- ✓ Provides effectiveness and low costs: SDNs correct cost saving is still in uncertainity. From the study about SDN, 50 percent of the administrators who make use of SDNs tell that they sold technology to their business executive as a money-saving methodology. Better opportunity in lower OPEX costs due to enhanced network management. SDN increase an organization's capability to innovate and remain agile, by automation and flexibility across the network.

II. BACKGROUND

More generally used term in Internet is "Cloud". Here, the users require not having infrastructure for dissimilar computing services. The users have right to use their information from any computers as well as from any part of world.Many services and models create Cloud Computing advantageous and easily available to the end user. This technique has two models. They are Deployment model and Service model. The deployment model consists of three types i.e., Public Cloud [5], Private Cloud [5] and Hybrid Cloud [5]. The private cloud is also called as internal cloud or corporate cloud which manages over their information compared with which is provided by the third party association. Public cloud provides services to everyone through Internet. In public cloud company rents ability along with they will pay for what they use. Hybrid Cloud is grouping of Private and Public Cloud.

The Service model contains three kinds of services i.e.,

Infrastructure-as-a-service(IaaS) [5], Platform-as-a-Service (PaaS) [5] and Software-as-a-Service (SaaS) [5]. In IaaS, only system is provided where as in PaaS system and operating system is provided as well as in SaaS operating system, necessary software and network is provided. Cloud service is well-known because it reduces the difficulty of network, customers do not want to buy software licenses and data in cloud is not simply missing.

- ➢ **Software-as-a-Service**

SaaS is a model in which a function is introduces as an administration with customers who obtain it through web. When the products are obtained off-site the customers require not overseeing it.

- ➢ **Platform-as-a-Service**

PaaS is a function delivery model. PaaS gives each resource, services thoroughly from Internet. That is not essential to download or install software.

- ➢ **Infrastructure-as-a-Service**

IaaS is an administration provided by distributed computing. SaaS and PaaS provide capacity to customers IaaS does not provide. This gives the components so the organization can embed whatever they want on it.

© 2016, IJARCSSE All Rights Reserved

OpenFlow

The interaction linking the control plane and the data plane in a SDN is using OpenFlow. It is the initial standard interface which is considered for SDN. It provides high performance grainy traffic accesses numerous vendors' network devices. The decoupling idea in SDN uses OpenFlow for the exchange of information linking the two planes.

OpenFlow architecture is shown in figure 2. Through OpenFlow protocol, the OpenFlow switch consists of single or additional table called flow table and a generalization level which interacts strongly via the Controller. Flow table consists of Flow entries. It indicates how packets are processed and forwarded. It contains

- **Match field:** Match field contains information which is found in the header of the packets, ingress port and metadata. These match field are used to equivalent incoming packets.
- **Counter:** For a particular flow counters are used to collect statistics. Those consist of number of received packets, flow duration and number of bytes.
- **Action:** These are set of instructions. Actions are useful upon a match. These set of instructions tells how to handle a matching packets.

Packet header fields would be extracted and coordinated in opposition to the flow table entries matching field's portion on arrival of packet at an OpenFlow switch. When an identical entry finds, switch will apply the suitable list of commands, otherwise activities which associates to the matched flow entry. If match is not found, switch action will be based on the commands which are defined with the table-miss flow entry. To take care the table misses, every flow should have a table miss entry. The mentioned entry mentions a set of commands which needs to be performed in case of no match set up for an incoming packet, for example drop a packet, maintain the identical process to subsequent flow table, and advance packet over the flow channel to the controller. Multiple tables and pipeline processing are supported by OpenFlow figure 2 Coming to the next option is hybrid switches which comprises of together. This helps in the process of transfer dissimilar bundles by subsequent IP forwarding schemes.

To process in and OpenFlow area, few machines which needs the interaction of OpenFlow Controller have to maintain the important OpenFlow protocol. During the boundary, this Controller moves along possible changes made to the switches or routers. This flow table enables administrators of network to divide transfer of packets, controlling moves and trying the latest configurations and applications.

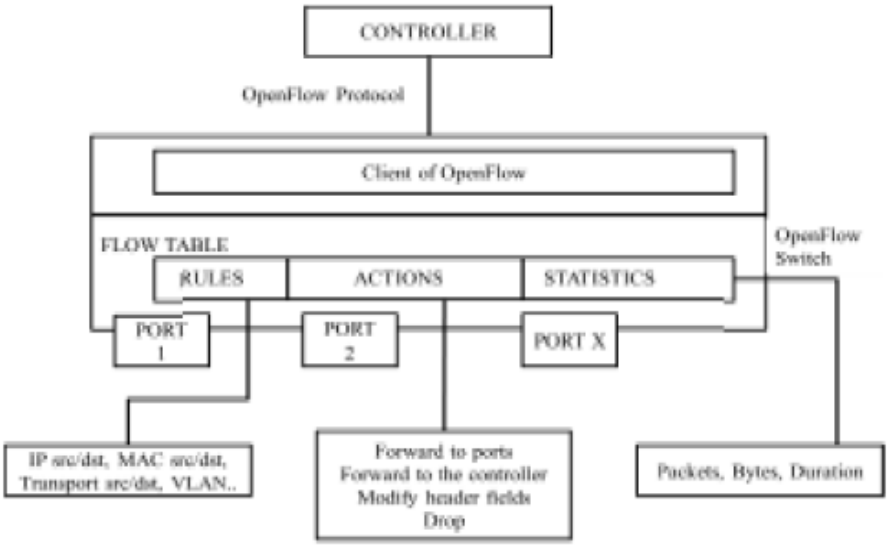

Figure 2: OpenFlow Architecture

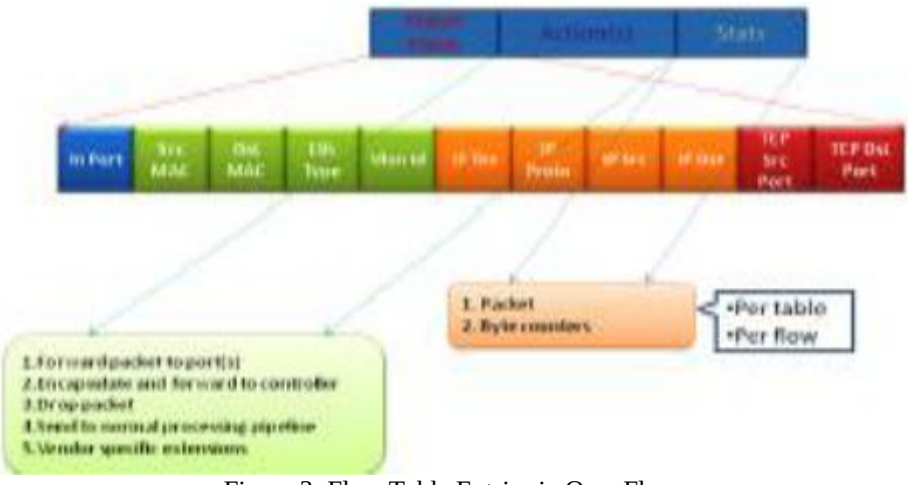

Figure 3: Flow Table Entries in OpenFlow

© 2016, IJARCSSE All Rights Reserved

The figure 3 shows OpenFlow table entries, it consists of list of entries to evaluate incoming packets from hosts next to flow entry, every flow entry includes match fields, statistics, activities to be performed, and similarities are identified from the first of the original flow table, in precedence way the entries are matched. It comprises the OpenFlow Work flow as packets reaches at each switch, the packet's header field is compared with flow table entries. If the equivalent found it will perform the equivalent instructions (forwarding to specified port), If match not found, it sends to the controller by means of the OpenFlow channel then it will drop the current flow table to start next flow entry table.

The fundamental data forwarding method along OpenFlow is described in figure 4. The switch gets a data; it then checks the header of the bundles that was corresponding next to the stream table. When the stream table is identified whose header part wildcard identifies the bundle's header, then that input is considered. In case, when there are numerous inputs of such kind are identified, bundles be identified depend on prioritization which means, mainly exact input or the wildcard with the maximum precedence is considered. Again the switch upgrades the counters of to kind of input. Ultimately, different activities indicated by the stream table input on the bundles are performed by the switch, for example: it moves the packet to next ports or else no entry is identified the packet header, then it considers usually informs to its controller regarding the packet, which is buffered while accomplished of buffering by the switch. At the end, it appends either unbuffered packets or on the beginning bytes of the buffered packet by means of PACKET-IN notice as well as moves it to the controller; this is general towards append the bundle header as well as number of bytes defaults to 128. This controller gets the PACKET-IN notice distinguishes accurate activities for the bundle and introduce one or many suitable inputs in the requesting switch. Buffered packets are sent based on the guidelines; it is activated with assigning the buffer ID in the flow inclusion communication or in express PACKET-OUT communication. Mainly the controller adjusts up the entire flow way for the bundle in the system by changing the stream tables of all switches on the way.

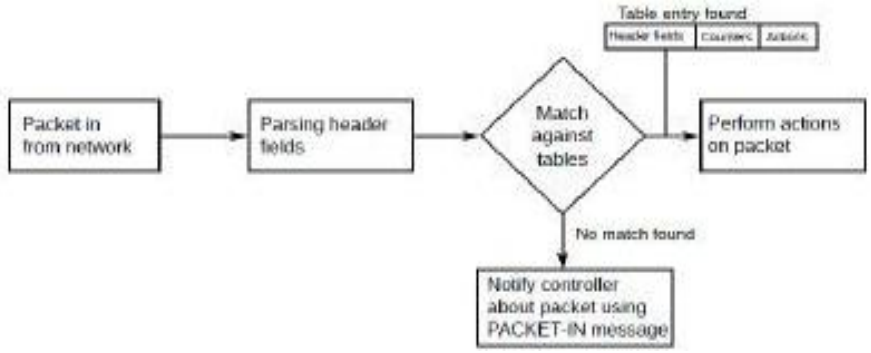

Figure 4: Data Forwarding along OpenFlow

OpenDayLight SDN Controller

ODL is a more transparent approach that Fasters the latest innovation and reduces threat, OpenDayLight is a software for forwarding elements, OpenDayLight controller uses a southbound plug in to interaction between physical devices (data planes), and by using northbound plug-in to expose interface to those writing a applications to the controller, network services and applications to controller, it support application that is run on top of the controller ,it support plug- in for inter-controller communication. The figure 5 represents the architecture of ODL controller.

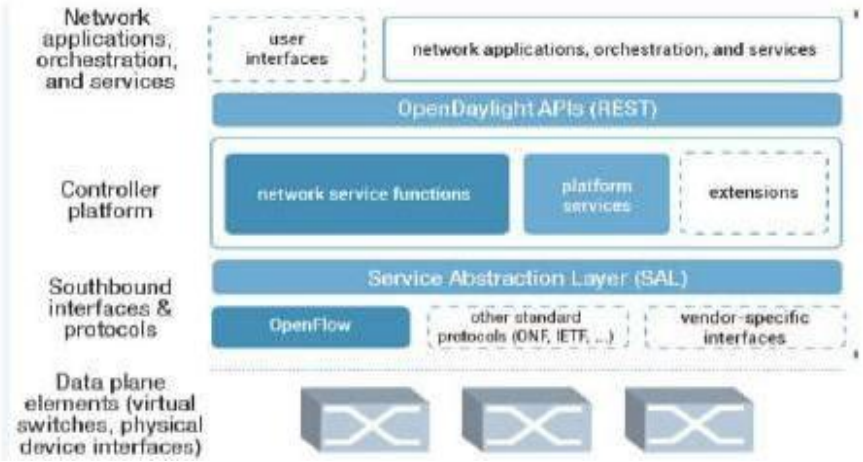

Figure 5: Architecture of OpenDayLight

III. SERVICE CHAINING CONCEPT

The interconnection of service instances, termed service chaining, works by mapping packets to service chains at the edges and forwarding them between service instances. The service chaining concept proposed here is based on OpenFlow and uses layer 2 addresses to identify service instances as discussed later.

© 2016, IJARCSSE All Rights Reserved

SFC: Services Overlay Model

- ✓ Decouples Service Function from Topology
- ✓ Overlays/Underlays for Transport

SFC: Orchestration

- ✓ Service chain definition
- ✓ Service chain instantiation

SFC: Policy

- ✓ Policy based Service Chaining
- ✓ Transferring Metadata for Context

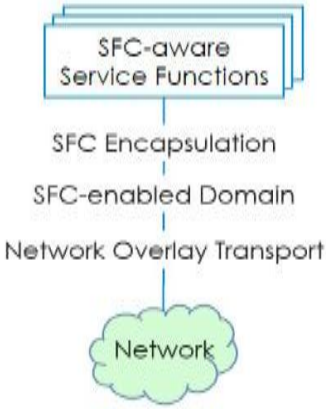

Figure 6: SFC Model

SFC Components

Service Function (SF):

- ✓ Function primary undertaking is implied in favor of precise handling of approaching bundles.

Classifier:

- ✓ Locally instantiated policy.
- ✓ Service profile matching of traffic flows for forwarding actions.

Service Function Forwarder (SFF):

- ✓ Forward to one or more associated Service Functions (SFs).

SFC Proxy:

- ✓ Eliminates as well as appends SFC encapsulation on.
- ✓ Can be used for SFC unconscious administration capacity.

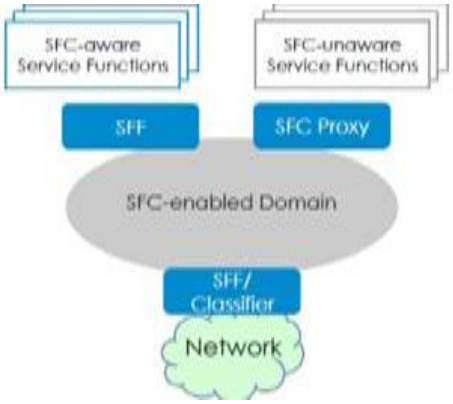

Figure 7: SFC Components

The applications of the dynamic Service Function Chaining are

1. Internet of Things

- ➢ Computing everywhere or Fog computing.
- ➢ Data aggregation and analytics at various levels.

© 2016, IJARCSSE All Rights Reserved

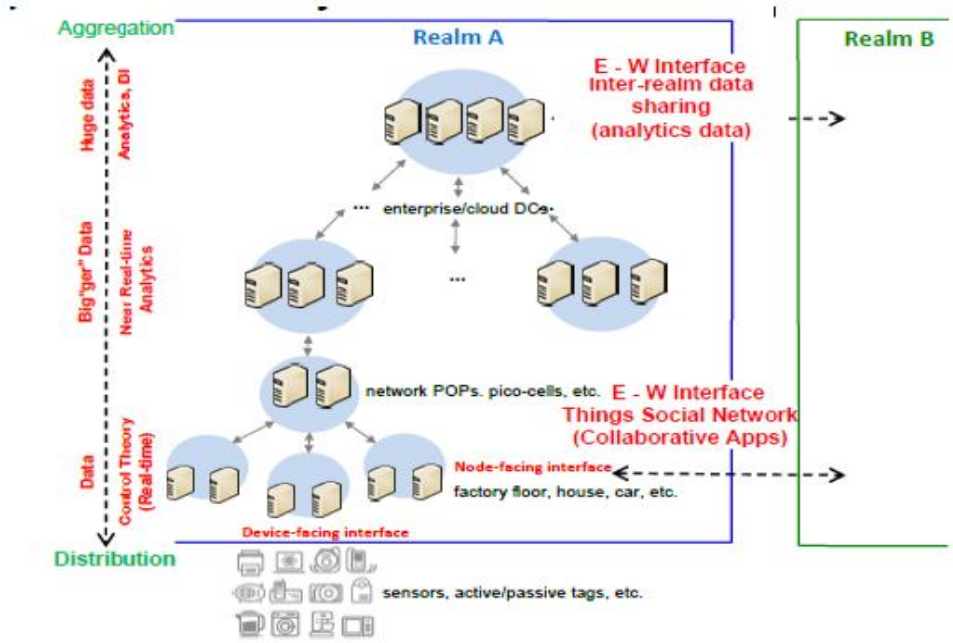

Figure 8: Internets of Things

2. Smart WANs

- Service Chaining.
- Message-level Middleboxes.
- Packet-level Middleboxes.
- Dynamically place services at PoPs based on application topology.
- Provide differentiated transport.
- Contextual mapping of messages to transport QoS.

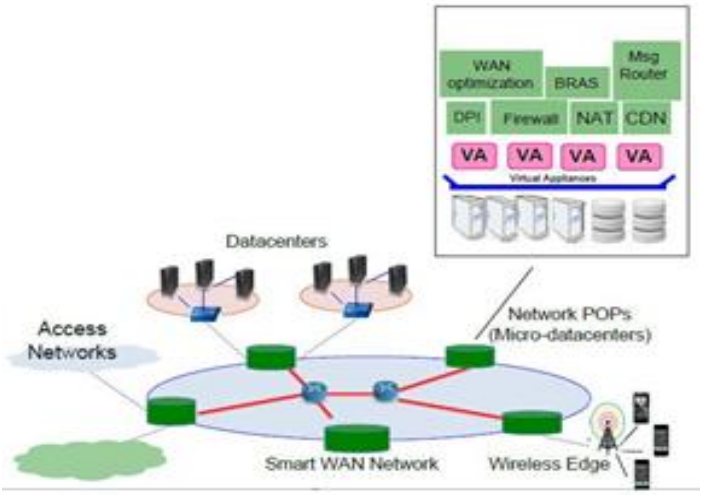

Figure 9: Smart WANS

3. Massively Distributed Apps

- Online games.

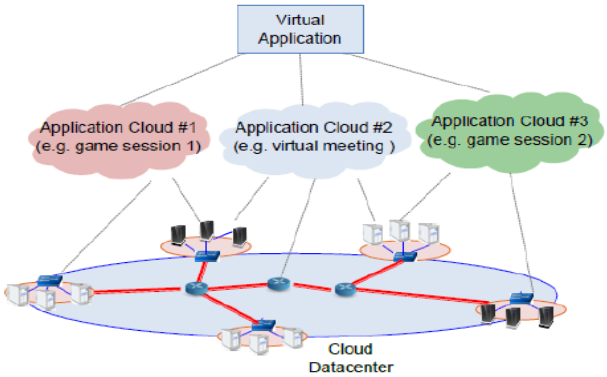

Figure 10: Massively Distributed App

© 2016, IJARCSSE All Rights Reserved

IV. EXISTING SYSTEM

Enablers are network services which are deployed as hardware appliances that are physically connected. These devices play a vital role in network operators achieving the security and performance they desire. Examples include the following devices:

- Firewalls
- Traffic Optimizers
- Network Address Translation (NAT) devices.
- Web Proxies
- Load Balancers

These devices are used to support a variety of applications. In the static service chain model, all traffic will have to flow through each of the enablers, even though only a subset of these services may be required. This means that the various appliances implementing these services will have to have enough capacity to manage the full traffic pipe, even if they will just have to let the flow pass-through without processing it. This also means that each service function will have to have its own internal capability for deciding whether a traffic flow has to be processed or dropped.

There are several limitations to this approach:

- ❖ All services have to be designed with a pessimistic approach, building them for the maximum possible capacity.
- ❖ As the traffic requirements increase, all services will have to increase in capacity, regardless of their actual usage; this could mean that sometimes even discontinued or very low-usage services will have to be scaled-up.
- ❖ There is little to no granularity in how the traffic is labelled and in how services are applied to specific flows.

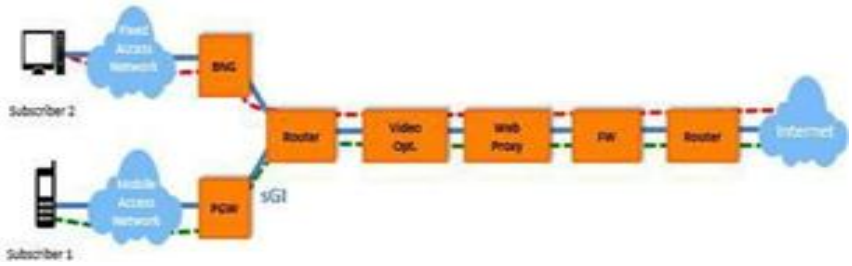

Figure 11: Static Service Function Chaining

In the example scenario illustrated in figure 11, Subscriber 1 wishes to access video content on their mobile device. The user would simply need the video optimization service, as well as basic firewalling. However, the user's traffic will have to traverse the entire chain. Adding to this, services must often be applied in a specific order, which implies the need for complex routing techniques and VLANs to ensure that this performed correctly. This example highlights the sub-optimal use of network and compute resources, as the entire service chain has to be traversed, regardless of whether this is required or not.

For a given service chain, which will be used to support a new application, the operator must first identify if existing appliances/topologies are able to cater for the application's needs. If not, a new topology must be considered; network devices must be purchased, physically connected and manually configured. For instance, in the case where the application's user base grows or there are reasonable peaks where the application is being used more often, operators have to constantly overprovision their network to cope with traffic based on estimates, when in reality the volume may never actually reach the expected level.

V. ARCHITECTURE

With SDN and NFV however, physical devices are replaced by virtual machines, which accelerate an operator's time to market for a new service. The reason for this is that, in SDN, the central-based controllers have a universal viewpoint of the network, which allows for new service chains to be created by an easy change in policy related with a flow.

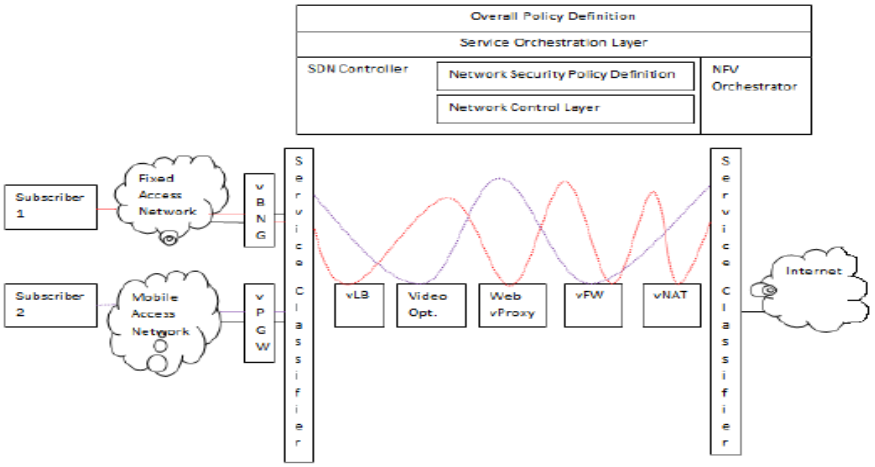

Figure 12: Dynamic Service Function Chaining

© 2016, IJARCSSE All Rights Reserved

As illustrated in figure 12, when traffic arrives at the network gateways, it is now labeled by a dedicated classification device with the use of deep packet inspection (DPI). The traffic is then intelligently forwarded to the required services, based on the service identifier. The identifier itself can be derived from a field in the traffic or it can be directly programmed in the switch flow-tables. This allows for network and compute resources to be used more efficiently, as traffic only flows through the required services. The provider is thereby relieved from constantly having to over-provision the network. As the proof of this benefit, in a case study with a Tier-1 service provider. An SDN based service chaining solution allowed the provider to reduce capital expenditures for a service by 80%.

Benefits of Proposed System

Agile

- ✓ Modified services to subscribers.
- ✓ Decrease TTM latest services.
- ✓ On-demand service delivery.
- ✓ Increase ARPU.

Automation

- ✓ Simplified end to end service Orchestration.
- ✓ Computerized configuration and provisioning.
- ✓ Reliable policy enforcement: SLA,Compliance.

Reduced TCO

- ✓ Operational simplicity efficient resource utilization.
- ✓ Active capability scale up/down.
- ✓ Pay-as-you-go usage model Agility Automation.

VI. IMPLEMENTATION AND TESTING

The overall working of the project is described in the below pseudo code.

Input: Application to be assessed on the Internet
Output: Particular appliances present in the Internet
Classifying the set of data packets and forwards the packets by service function forwarder
If (Host IP address is present in the Flow table)
Selects the particular Service Functions from the Service Function Chain
Providing application from the Internet
Else
Unable to access the application
End if

Starting OpenDayLight and Service Function Chaining

1. Start OpenDayLight by doing the following command

./sfc-karaf/target/assembly/bin/karaf

2. In the web browser

http://10.0.2.15:8181/apidoc/explorer/index.html
Authentication required screen before proceeding

i. Username= admin
ii. Password= admin

3. Open a new web browser Tab

http://10.0.2.15:8181/sfc/index.html

iii. Change the IP address to the one in step "i"
iv. Username=admin
v. Password= admin

Starting of SFC Agent

The SFC agent allows for almost configure-less deployment. It takes care of spawning SFFs and SFs as needed and configuring the data plane for those. OpenDayLight and the agent communicate through a Southbound REST interface that is extremely easy to use, learn and eventually implement in any product.

Before starting the agent, edit the file start_agent.sh and substitute the IP address by the IP address of the host running ODL. In this example we are running everything on the same host but in reality ODL could be running anywhere.

1. Open a new terminal new windows
2. SSH to your Ubuntu Machine
3. cd sfc/sfc-py
4. vi start_agent.sh
5. Modify the ip address with your local ip address

© 2016, IJARCSSE All Rights Reserved

6. Start the agent
a. Sudo ./start_agent.sh

Create a Service Function (Appliances)

Provide the necessary information like Service Function name, type, IP management address, Uri etc.

Create a Service Function Forwarder (Switch)

Provide the necessary information like Service Function Forwarder Name, service node, IP and port which helps to configure this SFF, URI of REST based management etc.

Create a Service Function Chain

1. Add Service Function Chain
2. Service Function Name Call it "Chain-1"
3. Result of the Chain-1 Creation
4.Drag the dpi Service Function to the Service Function Chain

Create a Service Function Path

1. Deploy this Chain
2. Enter Unique path name Chain-1-Path-1
3. Verification
a. Go to Service Function Path and check for Chain-1-Path-1

Create a Rendered Service Path

1. Enable Symmetric Path
2. Save
3. Verification
4. Create Rendered Service Path
5. Verification of the Rendered Service Paths

Send Packets

1. Open a new terminal
2. cd ~/sfc/sfc-py/sfc
3. run the following command

Python3.4 sff_client.py –remote-sff-ip 10.0.2.15 –remote-sff-port 4790 –sfp-id 1 –sfp-index 255

Table 2: System testing for different Test cases

Sl. No	Test case input description	Expected Result	Status
1	SDN Controller: Start	Host Reachable	Pass
	SDN Controller : Stop	Host Unreachable	Pass
2	Service Function Forwarder: Valid IP	Allows forwarding of packets	Pass
	Service Function Forwarder: Invalid IP	Not allows forwarding of packets	Pass
3	Service Function Chain: Specific Service Function is present	Provides the service function	Pass
	Service Function Chain: Specific Service Function is absent	Change the topology	Fail

VII. CONCLUSION AND FUTURE ENHANCEMENT

In this project successful implementation of service function chaining in Software defined network using OpenDayLight is done. Here different network services like Load Balancing, Firewall, Deep Packet Inspection, Video-Optimizer and Parental Control are used. By incorporating this in a single network client can use different services based on their specific operations or functions. The Dynamic Service Function Chaining provides dynamic use of the service thus reduce the time complexity.

As a future work, use of some algorithms like SF Selection algorithm will be implemented in the ODL SFC Project in order to evaluate and prove its benefits on real networking environment. Using a real system, we will be able to monitor the actual SF instances loads and links delays and apply the algorithm also on the runtime phase of the SFC lifecycle. In this way, it will be possible to instantiate multiple SFPs for each SFC in order to provide dynamic SFP on runtime to avoid congestion and SFs instances overloads.

© 2016, IJARCSSE All Rights Reserved

REFERENCES

[1] Martin Vizv´ary "Mitigation of DDoS Attacks in Software Defined Networks" Jan 2015.

[2] Nick McKeown: Stanford University, Tom Anderson: University of Washington, HariBalakrishnan: MIT, Guru Parulkar: Stanford University, Larry Peterson: Princeton University, Jennifer Rexford: Princeton University, Scott Shenker: University of California Berkeley, Jonathan Turner: Washington University in St. Louis "OpenFlow: Enabling Innovation in Campus Networks" Mar, 2008.

[3] ONF White Paper "Software-Defined Networking: The New Norm for Networks" Apr, 2012.

[4] Woosik Lee, Yoon-Ho Choi, Namgi Kim "Study on Virtual Service Chain for Secure Software-Defined Networking".

[5] Mohammad Sajid, Zahid Raza "Cloud Computing: Issues & Challenges", International Conference on Cloud, Big Data and Trust 2013, Nov 13-15, RGPV.

[6] Ben Kepes "SDN meets the real-world: implementation benefits and challenges".

© 2016, IJARCSSE All Rights Reserved