Implementation and Analysis of Distributed Network Functions Virtualization

Abhirami Shankar, Aditya Saroja Hariharakrishnan, Omkar Dattatraya Pawar, Hariram Natarajan, Shriya Rodi, Antara Kolar
Department of Electrical, Computer, and Energy Engineering
University of Colorado Boulder

***Abstract*—A majority of communication service providers are using Network Function Virtualization (NFV) currently as a method to accelerate service delivery and reduce associated overhead and costs at the same time. Ease of deployment, vendor independence and cost reduction are some of the factors that NFV addresses. While distributed systems have for long been used to provide decentralized operation, through this paper we propose a system to integrate NFV and distributed systems, so that the advantages of either system can be enhanced upon with the other. We focus mainly on the customer edge and the distributed cloud. Evaluations for the system are primarily on the latency and round trip time measured by comparing with a centralized system where all the functions are present in the cloud verses functions being distributed over the entire architecture.**

***Index Terms*—NFV, edge computing, distributed cloud, latency.**

I. Introduction

Internet has been around for decades now. However, the way Internet is expected to work and provide services has changed a lot during this period. Today, organizations, employees and regular customers are using the Internet for myriad of applications and services. Each consumer connecting to the Internet expects customized services and these expectations vary as per the business needs. However, a network which can grow or shrink as per requirements, which can be deployed faster, and a network that can be configured and managed from distributed locations are the basic requirements for todays networks. These new demands on the Internet have put many challenges on the Internet Service Providers (ISP). Traditionally, ISPs build networks to provide higher bandwidth, less latency, and a secured infrastructure to transport data packets. But now the need has arisen to build networks which are more scalable, faster to deploy and wherein network elements can modify their functions as per business needs. This will provide optimal performance for different types of services rather than being optimized for a single type of service or application. Todays networks are hardware-specific, and thus vendor dependent, expensive, complex to build, manage and as a result, unsuitable for scalability, flexibility in operation and innovation. These are the challenges pertinent to building todays networks and our project aims to address some of these challenges. The underlying technology implemented is Network Function Virtualization (NFV), which decouples network functions such as switching or routing from specialized hardware devices and implements these functions as a software on a commercially available white box that runs x86 servers, giving users the flexibility to run any OS for example, Linux or Windows on the standard server. Traditional approach towards NFV concentrates implementing network functions in data centers or at some centralized location in network. However, this approach may not be optimal for all the use cases. Thus, our project aims at implementing the Network functions distributed over the entire network topology including customer premises in addition to distribution on the cloud.

II. Background

Network functions virtualization is an idea which eliminates the need for specific physical hardware such as a router or switch. It involves implementing network functions in software that can run on an industry standard server hardware, thereby eliminating the need to deploy function specific network hardware. We can thus have a virtualization layer like hypervisor and spun multiple Virtual machines (VMs) running network functions. In case the load on one particular standard server tends to increase, it can be moved and instantiated in various locations in network as required, without the need to install new equipment. This in turn improves flexibility, increases efficiency and openness i.e. makes it vendor agnostic.

III. Distributed NFV

There are multiple approaches that exist to deploy white boxes or standard servers in the infrastructure to run virtualized network functions. The centralized approach advocates hosting virtualized functions in single location such as data center or infrastructure owned by service provider. However, a better distributed approach is possible which encourages placing the virtualized network functions throughout the network which includes the customers premise too. The following figure describes the basic idea of distributed network function virtualization. This includes implementing NFVs wherever they will be more effective and will be less expensive to implement. There are numerous advantages to implement certain NFVs at the edge which is within customer premises. The choice of what kind of functions need to be deployed at customer premises depends on how essential that functionality is to the customer, and how much control of the function should be given to the customer. Certain functions such as Quality of Service (QoS) management will provide better performance if implemented locally within customers premises, since it depends on the customer as to how they want to prioritize the services. This will also give more control to

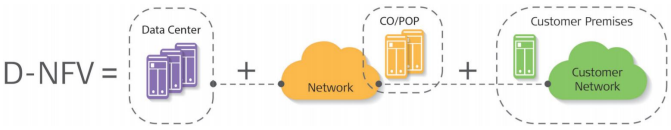

Fig. 1. Distributed NFV

the customers as they can influence the implementations as per their policies and available resources. Thus, implementing few NFVs at customer premises is more feasible, cost-effective and can be more flexible. The distributed approach of deploying and implementing NFVs at various locations throughout the infrastructure as per requirements is thus desirable to address the current and future challenges in network deployments.

IV. Motivation

The challenges mentioned in above sections are crucial to existing as well as future networks, and solutions for these challenges will certainly provide better network infrastructure. By implementing NFV, Internet Service Providers can replace the vendor specific network equipment with well known and standard servers like those based on x86. This will significantly reduce the capital and operational cost and vendor lock-in for the equipment can be completely eliminated. Incorporating new infrastructure will be as simple as connecting a standard hardware and downloading required software to make it work as a network equipment. Such scalability will result in faster deployment of network devices and thus services and applications running on it, as compared to months required for deployment in the hardware centric approach.The agility gap is thus eliminated and service providers compete for the same network service at much lower rates. NFV approach will also enable customizing the role of equipment as per the service requirements. Another motivation to address these challenges is that NFV will accelerate the innovations and make it easier to validate. We can develop new technologies and apply them using NFV for validation and testing before bringing it to the production environment. Compared to the time and cost associated with implementation and testing on specific hardware, NFV can generate results early and economically.

V. Related Work

In the last few years, NFV has been developing among service providers who are currently working on providing a standardized framework for the new architecture and technologies. In October 2013, the NFV ISG published its first five documents, proposing a framework to support interoperable NFV solutions. Among the topics addressed in these first specifications are NFV terminology, requirements, architectural framework, and use cases.NFV has to be properly managed and monitored from its earlier stages and this has been accomplished by the NFV MANO(NFV Management and Orchestration). The NFV MANO is a working group of the ETSI(European Telecommunication Standard Institute).Another important step of the NFV MANO is to include a SDN Contoller in its future infrastructure. Thus, the management and orchestration of all the cloud resources is accomplished by the NFV framework of ETSI.This has been further discussed in section [VI] of the paper. In March 2015, Telco systems launched Open Edge Alliance which builds an on-line store of applications certified to run seamlessly on Telco Systems carrier-grade, a Distributed NFV-hosting Cloud Metro platform. The Virtual Distributed Ethernet (VDE) switch or the bridging code is used by VMware distributed switch and Linux-based virtualization environments. For the virtualization layer, networking vendors are also starting to create virtual switches and other virtual network devices. Moreover, a scalable virtualized CPE solution is Juniper Networks Cloud CPE. It replaces dedicated CPE hardware with multiple routing and security VNFs, such as Junipers vMX virtual router and vSRX virtual firewall, into a highly scalable end-to-end solution. The Juniper Cloud CPE solution is built on Junipers NFV architecture that automates service delivery regularly and rationally across distributed, centralized, and overlay deployment models.

Ciena's Distributed Network Functions Virtualization (D-NFV) solution addresses the most prevalent use case in NFV today that is enterprise virtual customer premises equipment. Additionally, it has introduced its Service-delivery switch and Service-Aware Operating System (SAOS) into concert with Ciena's Blue Planet orchestration software. This has resulted in a complete D-NFV offering. CenturyLink's managed services strategy is supported by the DNFV concept as it provides a platform for developing unique NFV-enhanced network services by bringing desired functionality directly to the customer premises. An opportunity to prove vendor interoperability in the industry is provided by this ETSI multi-vendor concept. Furthermore, which functions work best at the edge or customer premises versus the network core are also determined by taking into account multiple factors which include customer control, resources available at the customer premises, decision making for services like QoS etc. We have proposed an approach in which the virtualized network functions are distributed over the entire network. Additionally, implementation of NFVs at the client premises would be easier, economical and flexible as per the clients policies and resources. Thus, it will be highly beneficial if NFVs are implemented at multiple locations all over the network along with the centralized implementation, thus guaranteeing redundancy, fault tolerance as well as efficiency at the same time.

VI. Architecture

The NFV framework as proposed by ETSI is the basic architecture on which our implementation has been built. This mainly contains three major functional components, as shown in Fig. 2

1) NFV Management and Orchestrator
2) NFV Infrastructure
3) VNFs

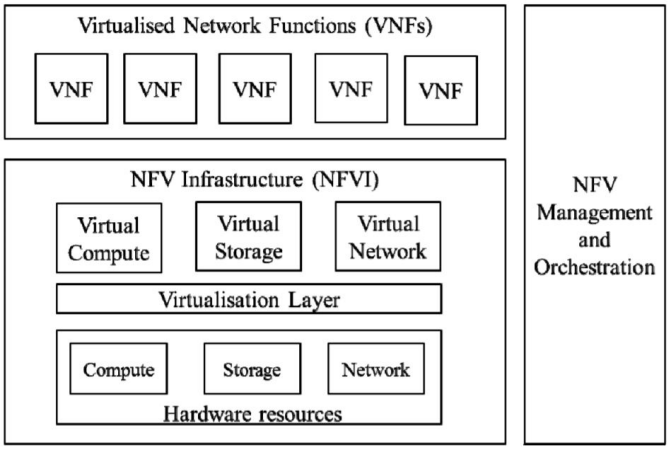

Fig. 2. Distributed NFV

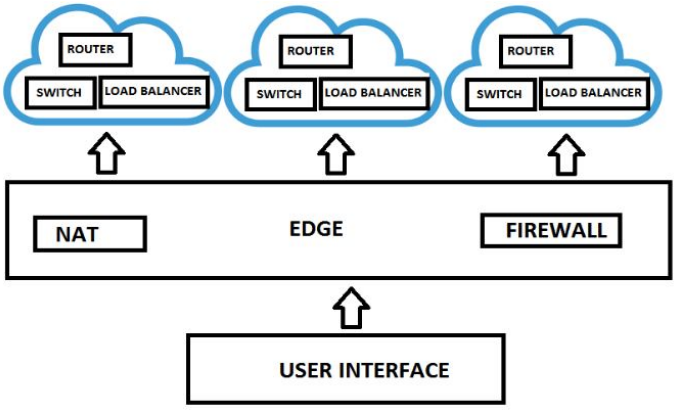

Fig. 3. Proposed Design

The NFV Management and Orchestrator is further subdivided into

1) NFV Orchestrator
2) VNF Manager
3) VIM (Virtual Infrastructure Manager)

The NFV Orchestrator is responsible for the installation of new VNF Packages and new Network Services. The VNF Manager on the other hand monitors the life-cycle of all the VNF instances. The VIM on the other hand controls and manages the NFVI compute, storage and network. The NFVI consists of the hardware resources which could be any standard and well known X86 server. Hypervisors are then implemented above this hardware infrastructure. The compute, storage and the network part of the virtual functions are pulled up, on top of the Hypervisor. The virtual network functions that we would like to implement can be now built above this infrastructure. Since we are implementing a hybrid model, the X86 servers are located both in the customer premises and the data-centers. The decision of placing the network functions in either of these locations can be based on pragmatic requirements of the user. If the users wish that none of their data should leave their premises without proper security and protection, then it makes sense to implement the firewall on the edge. In our case, we have implemented the Firewall and NAT on the Edge, while the router, switch, and load-balancers have been implemented on the data-center (Cloud).

VII. Design and Implementation

Before we get into the design specifics, we would like to state the assumptions we made for the simulation. We assume that we are an Internet Service Provider who has control over the edge as well as the cloud. The design we propose, involves virtual machines running services and acting as edge networking devices. At the cloud level, we are using three Amazon EC2 instances, each in California, Virginia and Ohio to simulate a distributed cloud system. Virtualized functions would be present in the distributed cloud and would be invoked into the edge networking devices upon request.

We have done preliminary implementation of both edge devices as well as the cloud distributed systems on a Linux CentOS running an Intel i7-5500U CPU. We have developed and implemented network functions of switch, router, load-balancer, firewall and network address translation. Among these functions, we have placed the firewall and NAT functions on the edge and the rest on the cloud. The network services are present on multiple cloud servers for redundancy purposes. The client who can be the system administrator of an ISP, has a menu driven program which specifies all the available networking services that can be offered upon his request as shown in Fig 3. The client can request for a functionality by forwarding it to the edge device which acts as a coordinator between the client and cloud. The edge device is aware of which cloud servers possess the requested function. When it receives a request from the client, the edge device will first look for whether the particular function is already present at the edge. If it is not present, then the edge will look at the list of cloud servers that possess the function and choose the best cloud server to download the code from amongst the list of servers, by considering the round trip time and location. As soon as the best cloud server has been selected, the edge requests for the particular function from the server which sends back the desired code to the edge device. Once the edge device receives the function, it starts acting as the virtualized network device until the client requests for something else. Following are the details of all the network functions that we have implemented.

A. Switch

A network switch is a Layer 2 device which looks and MAC addressess and forwards data between devices connected to it over a common Local area Network (LAN) segment. The switch parses the Source MAC and destination MAC and builds a CAM table based on the Source MAC and incoming interface. Then it checks for a match in the CAM table for the Destination MAC. If no match is found, it makes a copy of

the data and broadcasts the input data on all interfaces other than the one on which it received the request from. Once the destination client replies, the switch appends the MAC onto it's CAM table such that if there are more requests for the same destination, the switch will directly unicast the message instead of a broadcast.

B. Router

We have implemented a simple router which builds the routing table and forwards packets from one network to another and chooses the best path to reach the destination. The Router parses the Source IP address and Destination IP address and builds a routing table based on the Source IP and incoming interface. Now, it scans the routing table and checks for a match to reach the destination IP address. If a match is found, the router ARP's for the destination's MAC address and once it receives a reply, it recreates the packet header with it's own MAC as source MAC and destination's MAC as DMAC, and the packet is forwarded to the respective destination. If there is no match for the destination IP, the ARP process is repeated for obtaining the MAC address of the next hop IP, the header is recreated with next hop MAC as DMAC and packet is forwarded to the next hop address through the router's connected interface.

C. Firewall

We have implemented a basic firewall which compares the incoming addressess against some preset rules and determines whether to accept the packet or drop it. These rules will decide the action to be performed on the incoming tuples. The rules are matched sequentially and it gives result as per "first-match wins" rule. To demonstrate the feasibility of implementation, we have used simple rules which will match the source IP, protocol and state of connection present in tuple against the rules. If no rule is matched, the default drop policy is applied.

D. Network Address Translator

In some cases, a networks internal addressing needs to be kept private from the external network. Moreover, the internal IP addresses cannot be used outside the network as they are invalid outside for use. Thus, the need for network address translation arises where the NAT router acts as an agent between the public network and a local private network where a single unique IP address is required to represent an entire group of hosts. The basic NAT functionality works such that the private client IP is translated to one public IP by the NATting router and the destination communicates to the router's public IP which is then remapped to the private client IP by the router. Port addresses are used as a unique identity to differentiate between multiple services for the same Client IP.

E. Load-Balancer

We have implemented a computational load balancer which uses a Virtual IP for servicing client requests but there are multiple physical servers located behind the virtual IP that the client is not aware of. Whenever the client wants to send any data, it sends it to the VIP. The load balancer receives this request and in our implementation, based on the mod of hash value of the filename, the load balancer forwards it to one of the physical servers also known as backend servers by establishing TCP connection with each of them. In this way, the load is distributed and handled by multiple servers. We use the edge as the load balancer and the cloud instances as the backend servers.

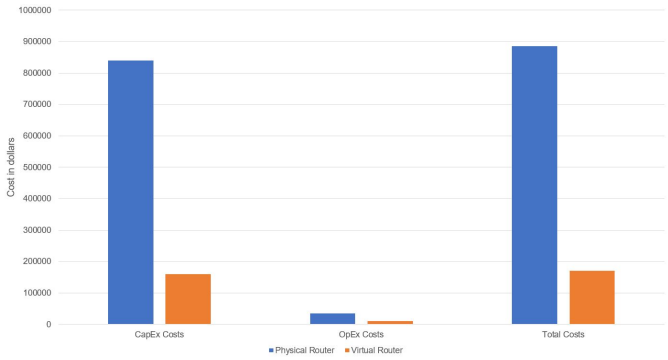

Fig. 4. Per Server Router Deployment Cost over Five Years

VIII. Evaluation

There are two parts to the evaluation. The cost evaluation basically compares the cost for deploying a physical hardware specific device versus a virtual one. The delay evaluation is a comparison between centralized and distributed Network Function Virtualization where we compare the latency and time involved in having all the functions present in the cloud versus functions being distributed over the entire architecture.

A. Cost Evaluation

Complex infrastructures are very expensive in terms of maintenance and operation. Network Function Virtualization will thus completely reshape the infrastructure of telecom providers by contributing towards a lesser CAPEX/OPEX. The Infrastructure flexibility thus helps service providers to dynamically change the service offerings thereby satisfying market needs.

The graph in Fig. 4 shows a comparison of costs for deployment of a physical router versus that of a virtual router.The X axis here indicates the CAPEX,OPEX and total costs while the Y axis indicates the total cost in dollars. The graph differentiates Capital Expenses, Operational Expenses and Total costs of a physical and virtual router over a period of five years. From the graph, it is evident that there is a significant reduction in the cost investment for a virtual router versus that of a physical one. Thus we can say that having network functions virtualized would prove to be economical as well as cost efficient especially for service providers who need to invest on deploying network devices almost all year round as per their increasing number of customers and demand for services respectively.

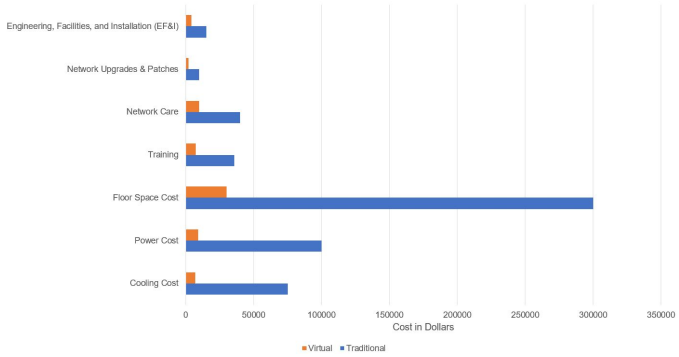

Fig. 5. Five Year Cumulative OpEx

The graph in Fig. 5 shows a comparison of the factors involved in hardware costs for example, cooling cost, installation charges, power space costs etc for physical and virtual devices. The X axis indicates the cost in dollars while the Y axis indicates the various factors.From the graph, we can see that hardware costs for general purpose x86 standard server is much lower than that of the hardware specific one. This is because the standard server possesses an industry standard x86 processor, storage capacity and processing power requiring minimal power, cooling etc. as compared to a bulky device with greater storage space and higher power requirements respectively.

B. Delay Evaluation

In this project, we mainly aim at creating virtualized environments for various network services like routing, switching, firewall service, load balancing and network address translation. In general scenarios, the number of hops, time taken to traverse every hop and the time taken for the reply usually determines the delay with which packets are delivered across the network.

A centralized NFV architecture relies on a centralized cloud server where the respective functions are stored. At any point of time a client will have to contact this centralized server for any service. However, in a Distributed NFV environment the edge stores the respective codes. In our case,few functions are located in the edge and the rest are in the cloud. In cloud, we have replicated these functions on multiple instances for redundancy purposes and have a mechanism by which the edge selects the best of these instances based on RTT. We have evaluated the latency involved in fetching a service from the cloud versus fetching it from the edge.

To compute latency, we use PCAP. PCAP running on the edge measures the time for the first outgoing packet and the first incoming packet. This is done for both the distributed versus centralized scenarios. PCAP consists of an application programming interface for capturing network traffic.

Using MATLAB, the comparison sketches were plotted.The graph here indicates the delay evaluation analysis for two services that are stored in edge versus two services that are stored in cloud. It should be noted that we have also implemented redundancy in the cloud,by replicating the services on all three instances. The X axis here represents the data packet at every instance and y axis represents the RTT. In the below Fig. 6, blue curve represents RTT computed by PCAP, which includes the time for data to travel from edge to the cloud servers. The peak in the graph indicates the time taken to select one of the three servers in the cloud in addition to the time taken for the reply from the nearest one that is selected. A constant blue line after the peak indicates that after the initial selection, the same server is selected for every packet and thus it just indicates the time for a reply from it.The green curve on the other hand indicates a case when the worst server gets selected especially in cases where the best one fails. This blue curve and green curves are a clear representation of the centralized model. The orange line on the other hand shows the time taken for the case where the function is implemented on the edge itself. This time is the processing time for the edge to search its own memory for the desired code. This is a clear representation of the distributed model. These scenarios were experimented using Amazon EC2 Clusters. We can observe that for any particular data packet at any instance, the difference in RTT between a centralized system and the distributed system is around 16ms to 17ms when the best server is selected for the centralized scenario, else it is around 40ms to 50ms when there is just one cloud server. Thus, a distributed NFV produces lesser latency in comparison with a centralized one in any case. However, due to the lack of power,space and resources it is not feasible to place all the services on the edge. Thus, there is always a trade-off between storing the service in the edge versus placing them in data-centers.

IX. Challenges

One of the major challenges is crash failure in white box. In order to reduce the frequency of such failures, tools such as MRTG (Multi Router Traffic Grapher) and PRTG (Paessler Router Traffic Grapher) can be used to monitor traffic load on the network. MRTG generates HTML pages containing images which provide live visual representation of this traffic. If the traffic exceeds a certain threshold and starts to load the network, we can use a backup white box and distribute the load over to the other box, thus avoiding a crash. As white boxes are cheap, multiple white boxes can be kept as backup. Also, we can monitor CPU utilization based on the OS from time to time and shift load based on a threshold value for CPU utilization respectively. PRTG is a server up-time and CPU utilization, network monitoring and bandwidth usage software package for server infrastructure. It monitors and classifies bandwidth usage in a network using SNMP, packet sniffing and Netflow. Few other challenges are ability to adapt to changing corporate culture, complex Ecosystem and integration, speed of operation in white box versus traditional hardware specific network devices and security. These challenges are mainly due to insufficient growth of our infrastructure to accept NFV on a large scale. There is a wide concern that existing switches, routers manufacturing companies would be reluctant to provide such virtualized functionalities, so there is need for

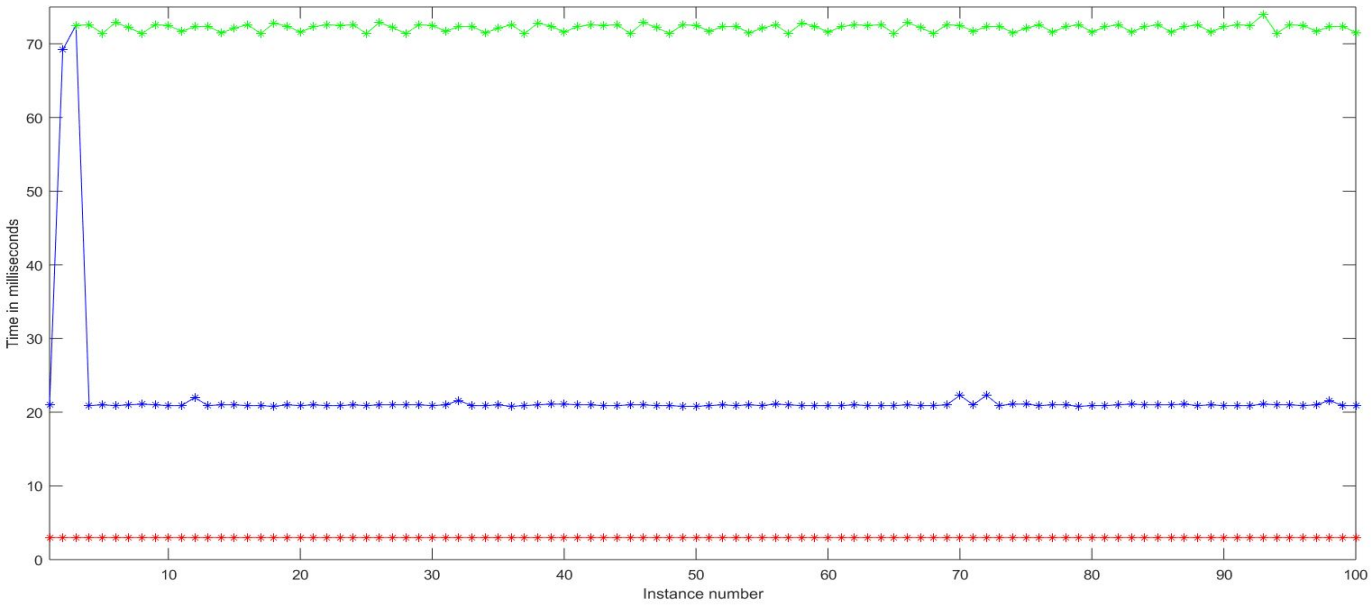

Fig. 6. Comparison of Centralized vs Distributed NFV (Best Case)

a proper platform to be established before distributed NFV's can be used commercially. Another major challenge for service providers is the need to ensure proper orchestration of the virtualized capabilities and functionalities while harmonizing IT/cloud and network resources. So, its better to have early deployment of distributed NFV for service providers to build infrastructure and mitigate their business.

X. Conclusion

We have tried to integrate Network Function Virtualization and Distributed systems in such a way that instead of having virtual functions located on the data centers or points of presence, it is now distributed over the entire architecture by having few functions that need customer control deployed near customer premises and few functions that would be seldom used, on the cloud. Our system eliminates vendor dependency and provides flexibility. By introducing a mechanism to choose the best cloud server to dwnload the functions not present in the customer premises, and replicating the functions on multiple cloud servers, we have implemented a system for redundancy as well as fault tolerance. Even though we can say that by having a general purpose x86 hardware implement all the functions there might be some compromise on the speed of computation, search and storage, as compared to traditional hardware specific devices, there is a tradeoff here since the speed is not significantly affected while there is a significant decrease in the cost. We have herewith proposed a system for increased efficiency, decreased overload, decreased latency, and reduced cost for functions. We believe that with emerging applications such as Internet of Things and Software Defined Networking, balancing load and computing data efficiently would become a necessity. With our system, these issues can be well addressed.

Acknowledgment

We would like to extend our thanks to Dr. Shivakant Mishra, Professor and Associate chair in the Department of Computer Science at the University of Colorado Boulder. We would also like to thank all our peers in Distributed Systems course for their support and valuable feedback. We also extend our gratitude to Amazon Web Services and AWS Educate for providing free instances for academic purposes.

Additional Resources

For the purpose of testing our work, we have created a GitHib repository. Our scripts can be cloned or downloaded from the following link.

https://github.com/DistributedNFV/Implementation-Analysis

References

[1] Aditya Saroja Hariharakrishnan and Abhirami Shankar, "Edge Based SDN Solution for Handling Increasing IoT Data," *https://github.com/AdvNetSys-EdgeComputing/Edge-Based-SDN-Solution-for-Handling-Increasing-IoT-Data*

[2] Yuri Gittik, "Distributed Network Functions Virtualization An Introduction to D-NFV", *White Paper RAD*

[3] Juniper Networks, "An Architecture for echnology Transformation," *White Paper Juniper Networks*

[4] Akamai, "Network Function Virtualization," *White Paper Akamai*

[5] European Telecommunications Standard Institute, "Network Functions Virtualization: Architectural Framework," *ETSI GS NFV 002, 2013*