IOSR Journal of Electronics and Communication Engineering (IOSR-JECE)
e-ISSN: 2278-2834,p- ISSN: 2278-8735.Volume 11, Issue 1, Ver. IV (Jan. - Feb .2016), PP 05-13
www.iosrjournals.org

SDN Market

Dr. Anitya Kumar Gupta

Abstract: *Software-Defined Networking (SDN) is a new and emergent model which had changed the conventional networks, by breaching their perpendicular incorporation, their network control logic by their underlying routers & switches, by supporting the unification of network control,& also providing the ability to program the network. The separation of concerns among the network policies, controlling hardware, and promoting the traffic, is the key for flexibility. In this manuscript we present a comprehensive overview on SDN, focusing on several technologies, attaining attention and also the advantages they offer for the cloud-computing suppliers & users. This manuscript comprises with introduction, motivation for SDN, and also explains their main concepts and differentiations from conventional networking and the key building blocks of an SDN infrastructure. This manuscript also provides the in-depth analysis for the hardware infrastructure, northbound & southbound APIs, SDN controllers, network virtualization layers, network Software languages, & the network applications along with the in progress research efforts & challenges. Finally, we also plan to examine the situation of SDN, the key enabler for a software-defined environment.*

***KeyWords**: Cloud-Computing, Conventional Networks, Network Controllers, Network Virtualization, Software-Defined Networking (SDN), Software Defined Environment, SDN infrastructure.*

I. Introduction

The Internet has prompted the making of a computerized society, where (just about) everything is associated and is open from anyplace. Notwithstanding, in spite of their far reaching selection, customary IP systems are intricate and difficult to oversee. It is both hard to arrange the system as indicated by predefined strategies, and to reconfigure it to react to blames, load and changes. To make matters significantly more troublesome, current systems are additionally vertically incorporated: the control and information planes are packaged together. The disseminated control and transport system conventions running inside the switches and switches are the key advances that permit data, as computerized bundles, to go the world over. In spite of their far reaching selection, conventional IP systems are intricate and difficult to oversee [1]. To express the fancied abnormal state system strategies, system administrators need to design every individual system gadget independently utilizing low-level and frequently merchant particular orders. Notwithstanding the design unpredictability, system situations need to persevere through the progress of issues and adjust to load changes. Programmed reconfiguration and reaction systems are for all intents and purposes non-existent in current IP systems. Authorizing the required arrangements in such a dynamic domain is in this manner very difficult.

To make it much more confused, current systems are additionally vertically coordinated. The control plane (that chooses how to handle system movement) and the information plane (that advances activity as per the choices made by the control plane) are packaged inside the systems administration gadgets, lessening adaptability and blocking advancement and development of the systems administration base. The move from IPv4 to IPv6, began over 14 years back and still to a great extent deficient, demonstrates the veracity of this test, while truth be told IPv6 spoke to just a convention upgrade. Because of the dormancy of current IP arranges, another directing convention can take 5 to 12 years to be completely composed, assessed and sent. In like manner, a fresh start way to deal with change the Internet design (e.g., supplanting IP), is viewed as an overwhelming undertaking – just not attainable practically speaking [2], [3]. At last, this circumstance has expanded the capital and operational costs of running an IP system.

The Software Defined Networking (SDN) [4], [5] is a mounting systems administration worldview that offers plan to change the constraints of current system frameworks. To start with, it breaks the vertical mix by isolating the system's control rationale (the control plane) from the fundamental switches and switches that forward the movement (the information plane). Second, with the partition of the control and information planes, system switches get to be basic sending gadgets and the control rationale is actualized in a legitimately brought together controller (or system working system1), disentangling strategy requirement what's more, system (re)configuration and development [6]. A disentangled perspective of this design is appeared in Fig. 1. Emphasize that a sensibly unified automatic model does not propose a physically brought together framework [7]. Actually, the need to ensure sufficient levels of execution, adaptability, and unwavering quality would block such an answer. Rather, creation level SDN system outlines resort to physically conveyed control planes [7], [8]. The partition of the control plane and the information plane can be acknowledged by method for a very much characterized Software interface is resided among the SDN controller and switches. The controller activities direct control over the state in the dataplane components by means of this all around characterized application Software interface (API), as delineated in Fig. 1. The most striking case of such an API is OpenFlow [9], [10].

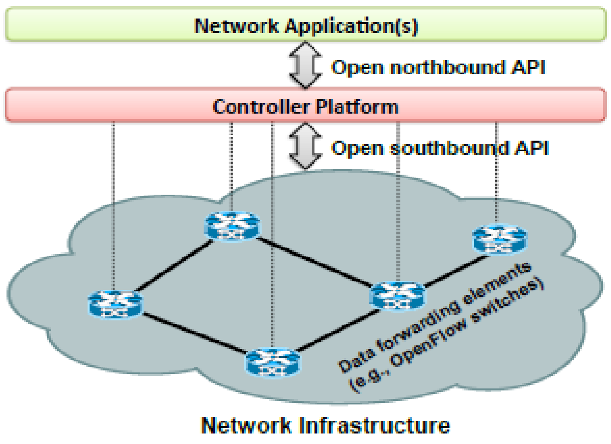

Fig.1. SDN Architecture Simplified View.

An OpenFlow switch has one or more tables of bundle taking care of tenets (stream table). Every standard matches a subset of the activity and performs certain activities (dropping, sending, altering, and so forth.) on the movement. Contingent upon the tenets introduced by a controller application, an OpenFlow switch can – trained by the controller – act like a switch, switch, firewall, or perform different parts (e.g., load balancer, movement shaper, and as a rule those of a middlebox). An imperative result of the product characterized organizing standards is the division of concerns presented between the meaning of system arrangements, their usage in exchanging equipment, and the sending of activity.

This partition is critical to the fancied adaptability, breaking the system control issue into tractable pieces, and making it less demanding to make and present new reflections in systems administration, streamlining system administration and encouraging system development and advanced.
Although SDN and OpenFlow started as educational experiments [9], they gained vital traction within the trade over the past few years.

Most vendors of business switches currently embrace support of the OpenFlow API in their instrumentality. The SDN momentum was robust enough to form Google, Facebook, Yahoo, Microsoft, Verizon, and Deutsche Telekom fund Open Networking Foundation (ONF) [10] with the most goal of promotion and adoption of SDN through open standards development. because the initial issues with SDN measurability were self-addressed [11] – specially the parable that logical centralization tacit a physically centralized controller, a difficulty we are going to come back to soon – SDN ideas have matured and evolved from an instructional exercise to a poster success. Google, for instance, has deployed a software-defined network to interconnect its knowledge centers across the world.

This production network has been in preparation for three years, serving to the corporate to boost operational potency and considerably scale back prices [8]. VMware's network virtualization, NSX is one more example. NSX may be a business answer that delivers a totally practical network in package, provisioned freelance of the underlying networking devices, entirely primarily based around SDN principles. As a final example, the world's largest IT firms (from carriers and instrumentality makers to cloud suppliers and financial-services companies) have recently joined the SDN consortium like the ONF in addition to also the OpenDaylight proposal.

A few recent manuscripts have specific some aspects of SDN. An outline of OpenFlow and a brief literature review are often found in and. These OpenFlow-oriented manuscripts gift a comparatively simplified three-layer stack composed of high-level network services, controllers, and also the controller/switch interface.. However, equally to the previous works, the manuscript is restricted in terms of scope as well as it doesn't give an in-depth treatment of basic aspects of SDN. In essence, existing manuscripts lack a radical discussion of the essential components of AN SDN like the network operative systems, package languages, and interfaces. They conjointly disappoint on the analysis of cross-layer problems like measurability, security, and reliableness. A a lot of complete summary of current analysis efforts, challenges, and connected standardization activities is additionally missing.

In the above Fig.2 represents the high-end SDN- Software Defined Networking architecture, here SDN have controller which provide a centralized or centered control plane for operating the switches. It also permits the SDN based applications network based operations. By, the outcomes there are visible undemanding changes relating to the conventional network architectures. The SDN ought to thoroughly boost the swiftness of the network improvement and progress the concert, scalability, flexibility, ease of management, cost and security.

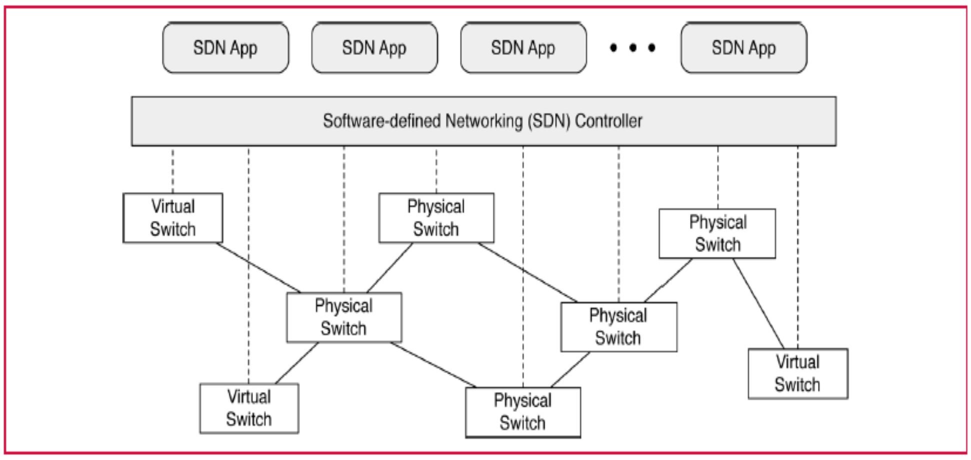

Fig.2. SDN- high-end Architecture.

The Software Defined computer- SDN and IBM's software defined environment (SDE) allocate the computerization and optimization of complete computing provides analogous benefits. In the center, the SDN provides an advanced abstraction based specification of the network connectivity & the services that can be automatically and dynamically mapped to a set of underlying network resources. We organize this manuscript into 6 sections which are described below.
I. Introduction
II. What Is Software Defined Networking?
III. Virtualization & Abstraction.
IV. Software-Defined Networks
V. Current Research
VI. Conclusion
VII. Acknowledgements
Our aim within the early part of the manuscript is to elucidate that SDN isn't a work of fiction as an advancement of technology. The existence of SDN is the intersection of a series of legacy ideas, along with technology drivers, and includes the current and future desires. The thoughts that are underlying SDN – the are the separation of the management and their dataplanes, the flow abstraction is upon the forwarding choices are created, along with the logical centralization of network management, and also the ability to program the network [17]. With the recent trends in networking– that specifically have the advancements like switch atomic conversion and also the curiosity in the possible styles of network virtualization – the resulting is the paradigm shift in networking technologies. By the results of the high business interest and also the potential to vary the standing quo of networking from multiple views, variety of standardization efforts around SDN are on-going, as we tend to additionally discuss in Section III these virtualization and other technologies more detalied.

Section IV is that the nucleus of this manuscript, presenting an in depth Associate relating to the Software Defined Networks. The comprehensive analysis also represents the essential components of an SDN infrastructure employing a bottom-up, bedded approach. The alternative for a bedded approach is grounded on the very fact that SDN permits thinking of networking on two of the elementary ideas, which are widespread in different disciplines of science they are: A) separation of considerations – relating to the leveraging the construct of abstraction and B) Rule. Relating to the, bottom-up approach it divides the networking down side into eight parts they are: 1) hardware infrastructure, 2) south interfaces, 3) network virtualization -hypervisor layer between the forwarding devices and also the network in operation systems, 4) network in operation systems- SDN Controllers and control platforms, 5) northward interfaces which are used to provide a typical computer code abstraction to the upper layers, primarily the network applications, 6) virtualization discrimination based slicing techniques that provide special purpose libraries or computer code languages and compilers, 7) network computer code languages, and eventually 8) network applications. Additionally, we tend to additionally consider cross-layer issues like debugging and troubleshooting mechanisms.

The discussion in Section V on the current Research efforts made by the researchers, challenges that are currently objecting the SDN, and future work that specifies the upcoming research concerns relating to the SDN. The section VI finally concludes and provides a conclusion of the author and finally it ends our manuscript.

II. What Is Software Defined Networking?

Software-Defined Networking- SDN was initially coined to signify the thoughts and effort around OpenFlow at University of Stanford [24]. It was initially defined as, a novel network architecture in which their advanced state in the dataplane is administrated by a remote control plane which is detached from the previous architectures. In the networking technologies and considering it from many situations it has shifted from the original outlook of SDN, by referring to anything that involves software as being SDN. We define an SDN as a network architecture that consists of 4 stakes:

1) **Control & dataplanes are detached**. Control functionality is unconcerned from the network devices that will be converted into simple packet based forwarding components.

2) **Forwarding decisions are flow-based, as an alternative of target- based**. The flow can be broadly defined as a set of packet field values that are acting as a match criterion and also as a set of instructions or actions. With the context of SDN/OpenFlow, the flow is a succession of packets flanked by a source along with a destination. All the packets of the stream receive indistinguishable service procedures at the promoting devices. Flow abstraction allows a coalesce behavior of various sort of network devices, switches, which also includes routers, firewalls, & middleboxes. The Flow programming also enables unparalleled flexibility which is limited to the abilities of the applied flow tables.

3) **Control logic** is stimulated to an exterior unit, known as Network Operating System (NOS) or SDN controller. NOS is a software platform which runs on service server knowledge and also provides the necessary resources & abstractions that make possible the indoctrination of a variety of devices which are based on a plausibly federalized, intangible network outlook. Their purpose is consequently parallel to the established operating system.

4) **Network is programmable-** The software applications are running over the NOS which interacts with underlying dataplane strategies. It is the basic feature of SDN.

The consistent centralization of control logic, are in meticulous, it offers quite a few additional benefits. Initial it is simple and less error-prone to modify network policies through high level languages and software components, compared with low level device specific configurations. Subsequently, a control program can automatically react to spurious changes of the network state and thus maintain the high-level policies intact. Thirdly, the centralization of the control logic in a controller with global knowledge of the network state simplifies the development of more sophisticated networking functions, services and applications.

Following the SDN thought introduced in associate degree SDN will be outlined by 3 basic abstractions: (1) forwarding, (2) specification, and (3) distribution. Abstractions area unit essential tools of analysis in technology and knowledge technology, being already associate degree omnipresent feature of the many laptop architectures and systems.

Ideally, the forwarding abstraction ought to permit any forwarding behavior desired by the network application, the management program whereas concealment details of the underlying hardware. OpenFlow is one realization of such abstraction, which may be seen because the cherish a “device driver” in associate degree software package. The distribution abstraction ought to protect SDN applications from the vagaries of distributed state, creating the distributed control downside a logically centralized one. Its realization requires a standard distribution layer that in SDN resides in the NOS. This layer has 2 essential functions. Initially, it is liable for putting in the management commands on the forwarding devices. Subsequently, it collects standing data about the forwarding layer (network devices and links), to offer a global network read to network applications.

The last abstraction is specification, which ought to permit a network application to precise the required network behavior without being liable for implementing that behavior itself. This could be achieved through virtualization solutions, as well as schedule languages. These approaches map the abstract configurations that the applications specific based on a simplified, abstract model of the network, into a physical configuration for the worldwide network read exposed by the SDN controller. Fig. 3 depicts the SDN design, concepts and building blocks. As antecedently mentioned, the sturdy coupling between control and knowledge planes has created it troublesome to feature new functionality to ancient networks, a reality illustrated in
Fig.4. the coupling of the management and knowledge planes (and its physical embedding within the network elements) makes the development and also makes the routing algorithms ready, as it'd imply a modification of the management plane of all network devices – through the installation of recent code and, also hardware upgrades in some cases.

The latest networking options area unit normally introduce via costly, specialized and hard-to configure equipment (aka middleboxes) like load balancers, intrusion detection systems (IDS), and firewalls, among others. These middleboxes ought to be placed strategically within the network, creating it even tougher to later modification the network topology, configuration, and practicality. In distinction, SDN decouples the management plane from the network devices associate degreed becomes an external entity: the network operating system or SDN controller.

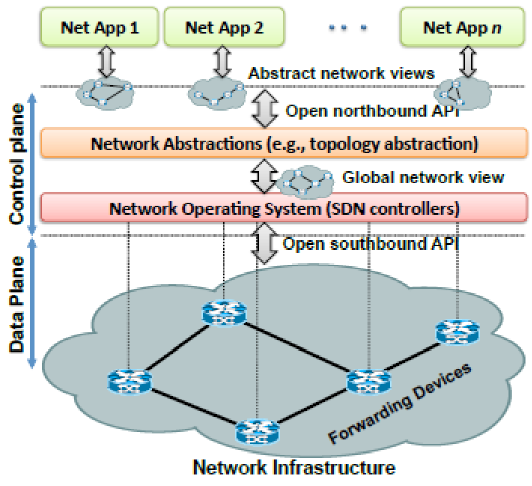

Fig.3. SDN Architecture and their Abstractions-fundamental.

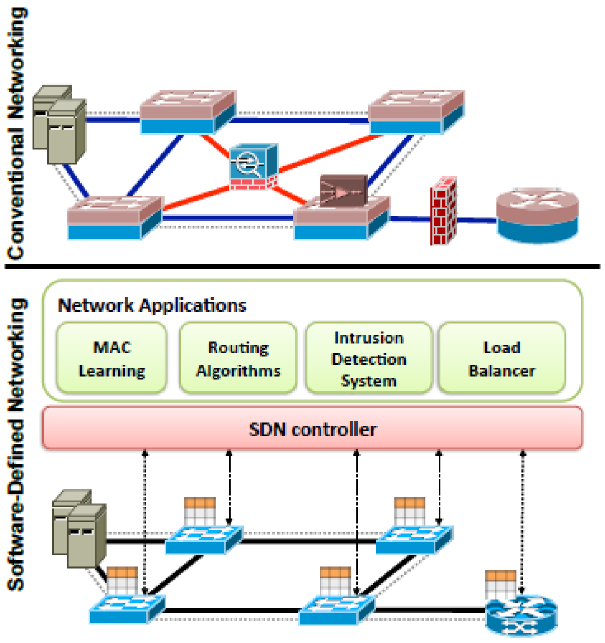

Fig.4. SDN and conventional Networking- Diffrences

II.I Advantages-Sdn:

The advantages of SDN are:

1. It is trouble-free to program the application as the abstractions provided by means of the control policy and by which the network based programming language can also be shared.

2. All applications be capable of taking benefit of the alike network information, leading to additional dependable and efficient policy decisions, while reuse control plane software modules.

3. These applications be capable of seize actions from any piece of the network. Therefore there is no need to invent a specific policy about the locality of the latest functionality.

4. Load balancing and the routing applications be capable of combined successively, with the load balancing choices having the priority over the routing strategies.

III. Virtualization & Abstraction:

SDN defines open, standard abstractions for networks that hide the details of the underlying infrastructure which is similar to the operating system. It abstracts the complexity of underlying hardware by exporting common application programming interfaces (APIs) to services such as file systems, virtual memory, sockets, and threads. These abstractions provide new tools to enable the richer networking functionalities demanded by recent industry trends including dynamic virtual server which has dynamic workload, along with the multi-resident cloud computing, in addition to warehouse-scale data centers.

Existing standards and abstractions have proven inadequate for delivering this functionality Bat scale, for example, 12-bit virtual local area network (VLAN) identifiers allow for up
to 4,096 isolated tenants. To address this, networks have become increasingly complex, including proprietary routing, traffic engineering mechanisms, and labor-intensive configuration of network appliances used to secure and optimize multi-tier systems.

SDN offers the potential to reverse this trend by addressing these problems in the controller software running on commodity servers that programs network hardware using open protocols. The dominant use of SDN that enables solutions to these problems is network virtualization. Network virtualization involves abstracting the physical network in two ways:
(1) Isolating multiple tenants and giving them a "view" and
(2) Presenting an abstract topology that may differ from the physical topology, e.g., an abstract topology with all hosts attached to a single, large switch. A related concept is Network Functions Virtualization (NFV), which replaces specialized appliances such as firewalls, load balancers, and intrusion detection systems with virtual machines (VMs) running on conventional servers [4–7] connected to the network. In the server world, virtualization has enabled new applications and revenue streams that would not have been technically possible or economically feasible otherwise, it is anticipated the same will be true for networking.

IV. Software - Defined Networs:

SDN design will be pictured as a composition of various layers, as shown in Fig. 5 (b). Every layer has its own specific functions. whereas a number of them ar forever gift in associate SDN preparation, like the south API, network operational systems, north API and network applications, others could also be gift solely specially deployments, like hypervisor- or language-based virtualization. Fig.5 presents a tri-fold perspective of SDNs. The SDN layers are diagrammatical within the center (b) of the figure, as explained higher than. Fig. 5(a) along with 5 (c) depict a plane familiarized read and a system style perspective, severally. The subsequent sections introduce every layer, following a bottom-up approach. For every layer, the core properties and ideas ar explained supported the various technologies and solutions.

IV.I. Layers

The various layers that are resided in the SDN architecture are discussed below.

Infrastructure

An SDN infrastructure, equally to a standard network, consists of a group of networking instrumentation. the most distinction resides within the undeniable fact that those ancient physical devices area unit currently easy forwarding components while not embedded management or code to require autonomous choices. The network intelligence is far away from the info plane devices to a logically-centralized system, i.e., the network software system and applications, as shown in Fig.5 (c).

Southbound Interfaces

Southbound interfaces (or southbound APIs) are the connecting bridges between control and forwarding elements, thus being the crucial instrument for clearly separating control and data plane functionality. However, these APIs are still tightly tied to the forwarding elements of the underlying physical or virtual infrastructure. Typically, a new switch can take two years to be ready for commercialization if built from scratch, with upgrade cycles that can take up to nine months. The software development for
a new product can take from six months to one year. The initial investment is high and risky. As a central component of its design the southbound APIs represent one of the major barriers for the introduction and acceptance of any new networking technology. In this light, the emergence of SDN southbound API proposals such as OpenFlow [9] is seen as welcome by many in the industry. These standards promote interoperability, allowing the deployment of vendor-agnostic network devices.

Network Hypervisors

One of the attention-grabbing options of virtualization technologies nowadays is that the proven fact that virtual machines is simply migrated from one physical server to a different and may be created and/or destroyed on-demand, sanctioning the provisioning of elastic services with versatile and straightforward management. Different workloads need completely different network topologies and services, like flat L2 or L3 services, or maybe additional complicated L4- L7 services for advanced practicality.

Therefore, it's exhausting to stay the first network configuration for a tenant, virtual machines cannot migrate to discretional locations, and also the addressing theme is mounted and exhausting to vary

Network Operating Systems / Controllers

Traditional operational systems offer abstractions (e.g., high-level programming APIs) for accessing lower-level devices, manage the synchronal access to the underlying resources (e.g., hard drive, network adapter, CPU, memory), and supply security protection mechanisms.

SDN is secure to facilitate network management and ease the burden of resolution networking issues by means that of the logically-centralized management offered by a network software package (NOS).

Northbound Interfaces

Northbound and the Southbound interfaces are the two key abstractions relating to SDN environment. Southbound interface had a accepted policy-OpenFlow, the universal northbound interface is tranquil an unwrapped issue. At this moment it may still be a bit too early to define a standard northbound interface, as use-cases are still being worked out. An abstraction that would allow network applications not to depend on specific implementations is important to explore the full potential of SDN. The northbound interface is mostly a software ecosystem, not a hardware one as is the case of the southbound APIs.

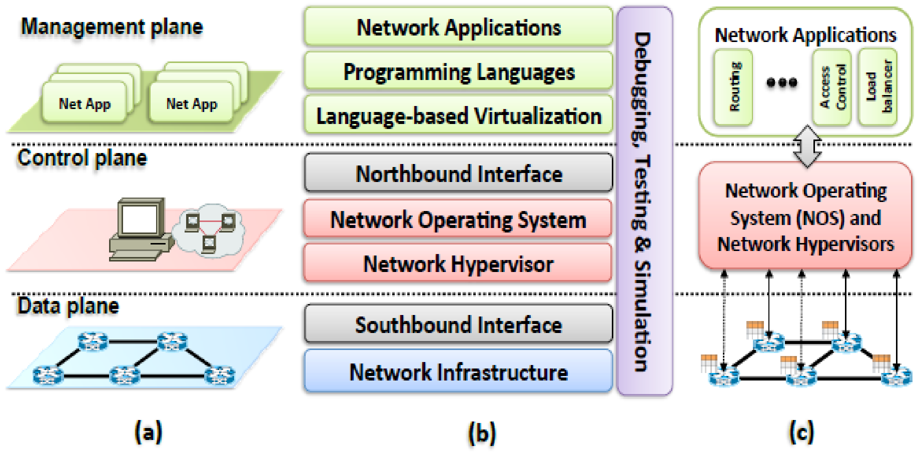

Fig.5. SDN in (a) planes, (b) layers, & (c) system design architecture.

Language-Based Virtualization

There are 2 essential uniqueness of virtualization solutions square measure the aptitude of expressing modularity and of permitting totally different levels of abstractions whereas still guaranteeing desired properties like protection. Pyretic is a motivating example of a artificial language that provides this sort of high-level abstraction of constellation. SDNs, high-level programming languages are used to design:

1) Form an abstraction of higher level to abridge the task of programming for the forwarding components or devices.

2) Facilitate added prolific and problem-focused environment designed for network based software programmers, towards speeding up the development in addition to innovation.

3) promote the software modularization in addition to code reusability in network control plane;

4) Advance the expansion of network virtualization.

Network Applications

Network applications will be seen because the “network brains”. They implement the control-logic which will be translated into commands to be put in within the knowledge plane, dictating the behavior of the forwarding devices. Software-defined networks will be deployed on any ancient network atmosphere, from home and enterprise networks to knowledge centers and net exchange points. Such kind of environments has light-emitting diode to a good array of network applications.

Accessible network applications carry out ancient practicality like routing, load equalization, and security policy social control, however conjointly explore novel approaches, like reducing power consumption. different examples embody fail-over and dependableness functionalities to the info plane, end-to-end QoS social control, network virtualization, quality management in wireless networks, among several others. the variability of network applications, combined with real use case deployments, is predicted to be one amongst the most important forces on fostering a broad adoption of SDN.

Despite the wide selection of use cases, most SDN applications will be classified in one amongst 5 categories: traffic engineering, quality and wireless, measure and watching, security and responsibleness and knowledge center networking.

V. Current Research:

Currently out there OpenFlow switches area unit terribly various which separates the appliance expectations from the switch of multiple virtual flow tables and switch drivers. Flow tables area unit supposed to fulfill the expectations of applications switch flow tables. Effort to develop a standard library to implement OpenFlow one.0 and 1.3 protocol endpoints (switch agents and controllers) a collection of the parameterized table properties every|for every} table certain each flow table, and also the data mask that may be passed Flow Table capability Flow matching rules area unit hold on in flow tables within network giant and economical flow tables to store the foundations. The actual capability in terms of OpenFlow table size has OpenFlow one.0 understood state explosion owing to its flat table consequently, saving up to fourth thousand flow table entries.

Shadow MACs propose label switch for resolution 2 enforced by straightforward hardware tables rather than these days, the outturn industrial|of economic|of business} OpenFlow switches shall be addressed within the switch style method support of central processing unit power of current commercial OpenFlow switches. These are powerful CPUs into the switches, as projected in between external controllers and also the OpenFlow agent within effective approach forward could be a native style of SDN switches evolving Switch styles &amp; Hardware Enhancements SDN switch styles are unit showing during a myriad of hardware SDN switch devices. with flow tables of up to 1M actual match entries and up to 1K applied to SDN to scale back prices in switch and routing Recent proposals on cache-like OpenFlow switch arrangements limitations of flow table sizes with clever switch styles. The application flow table area unit alternative approaches towards evolving switch styles embody deliver superior SDN software package switches
OpenFlow 1.0 switches [435], providing a artifact SDN knowledge planes proposes to enhance switches with FPGA algorithmic abstraction of OpenFlow controllers wherever every controller sees the controllers below as OpenFlow switches. Modularity in most SDN managementlers forces developers to re implement attempt to deliver the goods modularity in SDN control programs.

VI. Conclusion

Software Define Networking is a upcoming networking technology and the traditional networks are complex and hard to manage. One of the reasons is that the control and data planes are vertically integrated and vendor specific. Another, concurring reason, is that typical networking devices are also tightly tied to line products and versions. In other words, each line of product may have its own particular configuration and management interfaces, implying long cycles for producing product updates or upgrades. All this has given rise to vendor lock-in problems for network infrastructure owners, as well as posing severe restrictions to change and innovation. Software-Defined Networking created an opportunity for solving these long-standing problems. Some of the key ideas of SDN are the introduction of dynamic programmability in forwarding devices through open southbound interfaces, the decoupling of the control and data plane, and the global view of the network by logical centralization of the "Network brain".

While data plane elements became dumb, but highly efficient and programmable packet forwarding devices, the control plane elements are now represented by a single entity, the controller or network operating system. Applications implementing the network logic run on top of the controller and are much easier to develop and deploy when compared to traditional networks. Given the global view, consistency of policies is straightforward to enforce. SDN represents a major paradigm shift in the development and evolution of networks, introducing a new pace of innovation in networking infrastructure.

In spite of recent and interesting attempts to survey this new chapter in the history of networks the literature was still lacking, to the best of our knowledge, a single extensive and comprehensive overview of the building blocks, concepts, and challenges of SDNs. Trying to address this gap, the present manuscript used a layered approach to methodically dissect the state of the art in terms of concepts, ideas and components of software-defined networking, covering a broad range of existing solutions, as well as future directions. We started by discussing the new paradigm along with traditional networks. Following a bottom-up approach, we provided an in-depth overview of software-defined networking

SDN has successfully managed to pave the way towards a next generation networking, spawning an innovative research and development environment, promoting advances in several areas: switch and controller platform design, evolution of scalability and performance of devices and architectures, promotion of security and dependability. We will continue to witness extensive activity around SDN in the near future. Emerging topics requiring further research are, for example: the migration path to SDN, extending SDN towards carrier transport networks, realization of the network as- a-service cloud computing paradigm, or software-defined environments.

Acknowledgements

We would like to thank the reviewer and our professor guide who had given us support and kept efforts in suggesting us and reviewing the article to achieve the high end quality and also to the reviewers who are going to review this manuscript.

References

[1]. Wenfeng Xia,Yonggang Wen, Chuan Heng Foh, Dusit Niyato, Haiyong Xie. (2015). A Survey on Software-Defined Networking. IEEE COMMUNICATION SURVEYS & TUTORIALS . 17 (1), 27-51.

[2]. Diego Kreutz,Fernando M. V. Ramos, Paulo Verissimo, Christian Esteve Rothenbermb, Siamak Azodolmolky Steve Uhlig. (2014). Software-Defined Networking: A Comprehensive Survey. IEEE COMMUNICATION SURVEYS . 17 (2), 1-61.

[3]. C. Dixon D. Olshefski V. Jain C. DeCusatis W. Felter J. Carter M. Banikazemi V. Mann J. M. Tracey R. Recio. (March/May 2014). Software defined networking to support the software defined environment. IBM Journal of Resedarch & Development. 58 (2/3), 3:1-3:14.

[4]. T. Benson, A. Akella, and D. Maltz, "Unraveling the complexity of network management," in Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation, ser. NSDI'09, Berkeley, CA, USA, 2009, pp. 335–348.

[5]. B. Raghavan, M. Casado, T. Koponen, S. Ratnasamy, A. Ghodsi, and S. Shenker, "Software-defined internet architecture: Decoupling architecture from infrastructure," in Proceedings of the 11th ACM Workshop on Hot Topics in Networks, ser. HotNets-XI. New York, NY, USA: ACM, 2012, pp. 43–48.

[6]. A. Ghodsi, S. Shenker, T. Koponen, A. Singla, B. Raghavan, and J. Wilcox, "Intelligent design enables architectural evolution," in Proceedings of the 10th ACM Workshop on Hot Topics in Networks, ser. HotNets-X. New York, NY, USA: ACM, 2011, pp. 3:1–3:6.

[7]. N. Mckeown, "How SDN will Shape Networking," October 2011. [Online]. Available: http://www.youtube.com/watch?v=c9-K5O qYgA

[8]. 7.S. Schenker, "The Future of Networking, and the Past of Protocols," October 2011. [Online]. Available: http://www.youtube.com/watch?v= YHeyuD89n1Y

[9]. H. Kim and N. Feamster, "Improving network management with software defined networking," Communications Magazine, IEEE, vol. 51, no. 2, pp. 114–119, 2013.

[10]. N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: enabling innovation in campus networks," SIGCOMM Comput. Commun. Rev., vol. 38, no. 2, pp. 69–74, Mar. 2008.

[11]. ONF, "Open networking foundation," 2014. [Online]. Available: https://www.opennetworking.org/

[12]. T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama, and S. Shenker, "Onix: a distributed control platform for large-scale production networks," in Proceedings of the 9th USENIX conference on Operating systems design and implementation, ser. OSDI'10. Berkeley, CA, USA: USENIX Association, 2010, pp. 1–6.

[13]. Mark Zuckerberg – founder of Facebook

[14]. Researchers of University of Washington

[15]. Bill Gates – founder of Microsoft

[16]. Dr. Anitya Kumar Gupta – Member of Campus London and Student Ambassador of Firefox.