International Research Journal of Computer Science (IRJCS)
Issue 11, Volume 4 (November 2017)
ISSN: 2393-9842
www.irjcs.com

APPLICATION OF OS PRINCIPLES TO DESIGN SOFTWARE DEFINED NETWORKING CONTROLLER

SALIL JAGTAP
Computer Science Department, TAMUK
Saliljagtap12@gmail.com;

ROSHINI KARUNAMURTHY
Computer Science Department, TAMUK

MUJAHID AKBAR ALI SYED
Computer Science Department, TAMUK

ASHISH SHAHANE
Computer Science Department, TAMUK

Manuscript History
Number: **IRJCS/RS/Vol.04/Issue11/NVCS10089**
DOI: 10.26562/IRJCS.2017.NVCS10089
Received: 09, October 2017
Final Correction: 23, October 2017
Final Accepted: 07, November 2017
Published: November 2017
Citation: JAGTAP, S., KARUNAMURTHY, R., SYED, M. A. A. & SHAHANE, A. (2017). APPLICATION OF OS PRINCIPLES TO DESIGN SOFTWARE DEFINED NETWORKING CONTROLLER. IRJCS:: International Research Journal of Computer Science, Volume IV, 37-42. doi: 10.26562/IRJCS.2017.NVCS100089
Editor: Dr.A.Arul L.S, Chief Editor, IRJCS, AM Publications, India

Abstract— In this paper, we use an existing operating system to enhance features of SDN controller rather than creating Yet Another Network Controller (YANC). YANC, a controller system uses state as a file system which enables user and system application to interact through standard I/O. In this paper we represent the goals, design and features of Open Network Operating System (ONOS).

Keywords— SDN; SDN Controller; ONOS; YANC; YANC File System;

I. INTRODUCTION

The advancement of computing has been swift over the past three decades, but there has virtually been no change in networking. The networks themselves have become a building block of all infrastructures in society and an important part of the emerging public and private clouds. However, the complexities of traditional networking approaches have increased [1]. They have become an obstacle to creating new, novel services within a single data centre, on interconnected data centres, or within enterprises, and an even larger barricade to the continued growth of the Internet [11]. The network is built using switches, routers, and other devices which is the cause of its limitations, that have surpassed complexities because they implement an ever-increasing number of distributed protocols and use closed and proprietary interfaces. Innovation in this environment, it is too difficult, for network operators, third parties, and even vendors to innovate [11]. Customized and optimized solutions and networks cannot be offered for their use cases that are relevant to their business by the operators.With SDN, the introduction of new features becomes automatic, error-free, and faster to implement. The introduction of Software defined network has generated monumental buzz in the past few years as it promises to simplify many of the network management issues that have been serious concerns for network operators. Software-defined networking uses a logically centralized control plane to manage a collection of packet processing and forwarding nodes in the data plane [3].

The genesis of software-defined networking began shortly after Sun Microsystems released Java in 1995 [10]. In April and May 2001, Ohio State University and OARnet, collaboratively ran the first SDN test and developed the first practical SDN use case. Software-defined networking was continued with work done in 2003 by Bob Burke and Zac Carman developing the Content Delivery Control Network patent application. CableLabs specified Digital Cable and CableCARD using what we now know as SDN, which debuted in 2007 [10]. SDN was again moved ahead in work done at UC Berkeley and Stanford University around 2008. The Open Networking Foundation was founded in 2011 to promote SDN and Open Flow [1]. At the 2014 Interop and Tech Field Day, software-defined networking was demonstrated by Avaya using shortest path bridging and OpenStack as an automated campus, extending automation from the data center to the end device, removing manual provisioning from service delivery.

A SDN controller is an application in software-defined networking (SDN) that governs flow control to enable intelligent networking. SDN controllers are based on protocols, such as Open Flow, that allow servers to tell switches where to send packets [3]. The core of an SDN network is the controller. The core lies between network devices at one end and applications at the other end. The controller manages any communications between applications and devices. Different network tasks are done by SDN Controller platform, which typically contains a collection of "pluggable" modules [3]. Some of the basic tasks including inventorying what devices are within the network and the capabilities of each, gathering network statistics, etc. Furthermore, to strengthen functionality and support more developed capabilities extensions can be inserted, some examples of extensions are running algorithms to perform analytics and adapting new rules throughout the network. The protocols such as Open Flow to configure network devices and choose the optimal network path for application traffic are used by the controller.

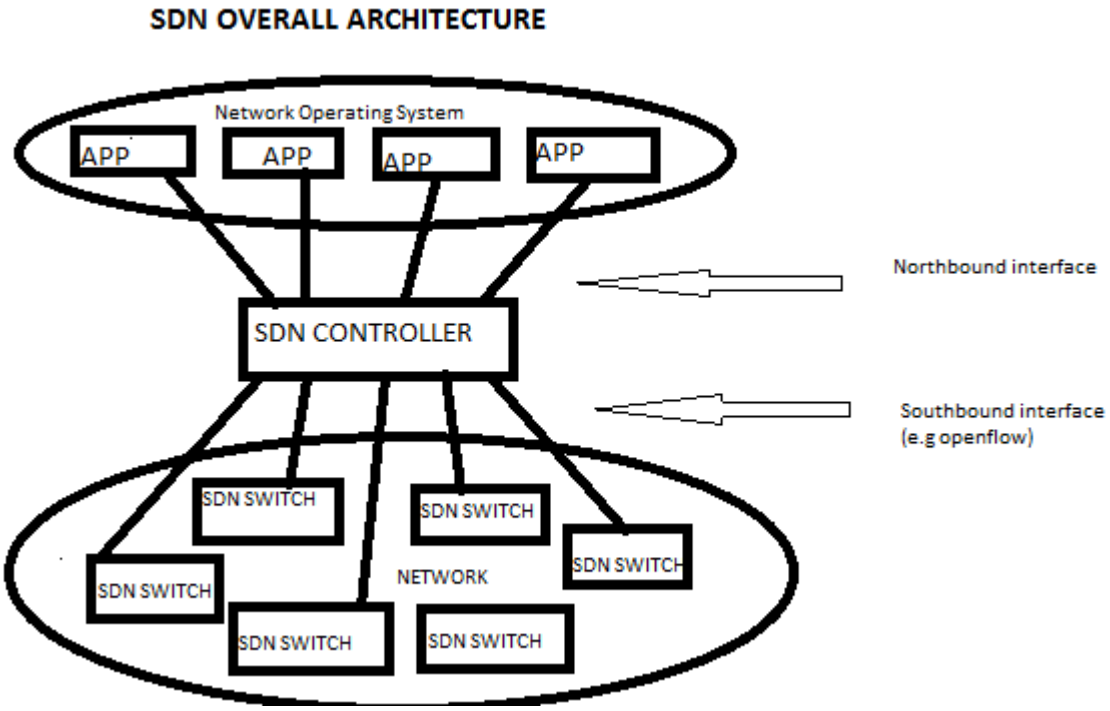

Fig I.1: SDN Overall Architecture.

Management of large network enterprise networks is difficult; hence controlling of management problem is done by introducing operating systems as an instructional example. The SDN controller serves as a sort of operating system (OS) for the network [3]. The control plane is taken off the network hardware and run as software instead. Automated network management is made possible by the controller and it eases the makes it easier to secure and operate business applications. Vendors of SDN controllers include Big Switch Networks, HP, IBM, VMWare and Juniper.

II. OPEN NETWORK OPERATING SYSTEM(ONOS)

Networks have become a very critical infrastructure for society. To resolve this complexity AT&T and NTT communications have started the project called ONOS and the source code of the project made open source for the community. The main goal of Open Network Operating System project is to design Software Defined Networking Operating System for communication service providers that are designed for high performance, high availability and scalability [7].

A communication service provider is a service provider which transfers the information electronically, for example a telecommunication service provider. ONOS provides the control plane for a software-defined network (SDN), managing network components, such as switches and links, and running software programs or modules to provide communication service to end hosts and neighbouring networks [8]. SDN separates the control plane from the data plane freeing software innovation cycles to become independent of hardware innovation cycles. Control Plane makes the decision where traffic is sent, and its packets are originated by router itself. Data Plane forwards traffic to the next hop along the path to the selected destination network according to control plane logic. The ONOS project is written in Java [2].

The ONOS will

- Bring carrier grade features (scale, availability and performance) to the SDN control plane.
- Enable Web style agility.
- Help service providers migrate their existing networks to white boxes.

The goals of an ONOS are as follows [6]:

- It frees network application developers from knowing the complexities associated with the hardware.
- Re-enable innovation to happen for both network hardware and software, independently, on their own time scales.
- The ONOS GUI provides the view of the multi-layer network and allows the user exploration of network state, network errors.

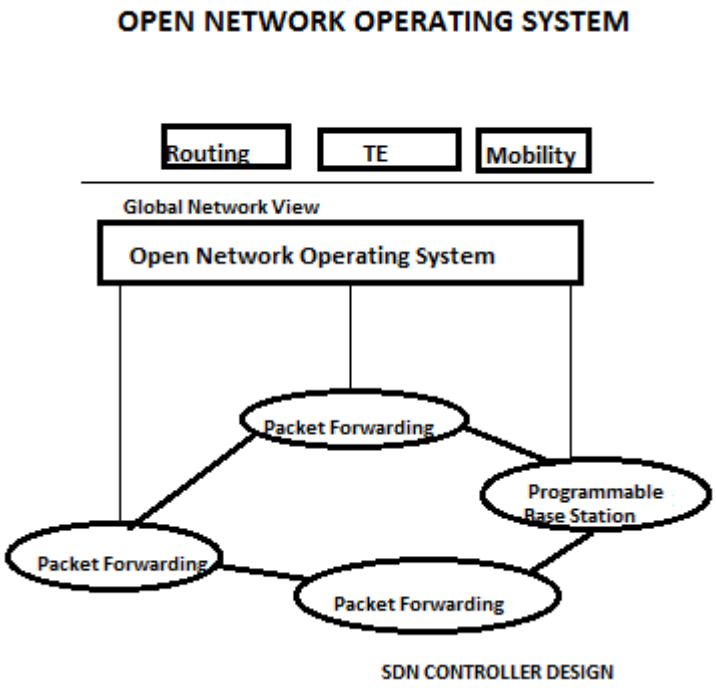

Fig II.1: Open Network Operating System Structure.

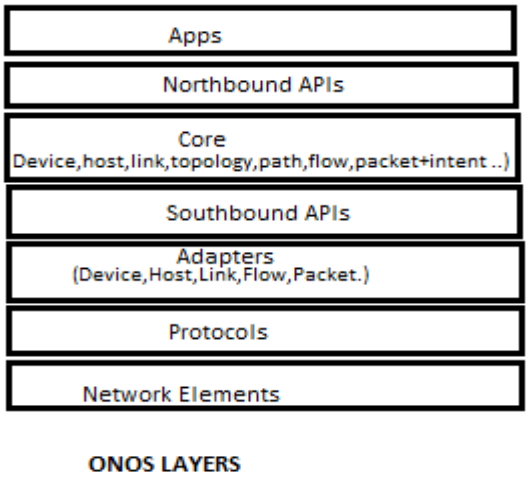

Fig II.2: ONOS Layers.

ONOS designing features:

1. Distributed Core: It mainly provides high availability, scalability and high performance.
2. Northbound APIs: Northbound interface allows a particular component of a network to communicate with the higher-level components. This API provides graphs of Network and applications which tries to ease development of controls, management and configuration service.
3. Southbound APIs: It enables the pluggable southbound protocol for controlling Open Flow.
4. Software Modularity: It makes easy to develop, debug, maintain and upgrade the ONOS as a software system [8].

III. YET ANOTHER NETWORK CONTROLLER (YANC)

Yanc is a collection of programs which make up a software defined network (SDN) control plane [9]. The core component of Yanc is a schema-based filesystem [4]. The yanc is a controller platform for software-defined networks which exposes the network configuration and state as a file system. In yanc, network applications are separate processes and it may be written in any language. In this paper we present the design of yanc. The yanc architecture, illustrated in Figure, builds off a central abstraction using the file system. As illustrated in Figure , the yanc architecture consists of a file system which represents the network and is the interface to the network, device drivers which inter-face between the file system and the physical network, a messaging system in Linux as the SDN controller event system [5], and leverages Linux's run-time system to exe-cute applications (i.e. ,as Linux executables) and users pace environment for extending the tools(e.g., grep)and distribution mechanisms (e.g., apt) available for network controller. With yanc, the configuration and state of the network is exposed as file I/O allowing running application software in a variety of forms and developing in any language. Much like modern operating systems, system services interact with the real hardware through drivers, and supporting applications can provide features such as virtualization, or supporting libraries such as topology discovery [2]. One special example that is made possible by building yanc into an existing operating system is that distributed file systems can be layered on top of the yanc file system to realize a distributed controller. Finally, while we mostly discuss yanc in terms related to the Open Flow protocol for ease of understanding, we believe the design of yanc extends into more recent research, going beyond Open Flow [5]. We believe that once fully implemented, yanc will enable researchers to focus on value-added applications instead of yet another network controller.

IV. THE YANC FILE SYSTEM

The focus of yanc is that network configuration and state is managed by file system. Therefore, network can be managed by standard file I/O and no special applications are needed. Interaction with network configuration and state through file I/O enables a robust environment for network administration since file systems are the backbone for modern operating systems [4]. A common interface is provided to a wide range of hardware, system state, remote hosts, and applications.

- File System Layout – File systems usually contain files with information on them, directories group the files in a specific format. The yanc file system is ascended on /net as shown in the figure. Hierarchal nature of network configuration and state is copied by the yanc file system and used as its directory structure [4]. Entities such as switches/ and hosts/ represent top level directories, they are also alternate representations of the state further they are also called as views. A combination of files and subdirectories represent the actual configuration (e.g. port status) and state (eg counters). Importantly in yanc directories and files contain semantic information, each directory contains a list of objects which creates object of appropriate type on system call.

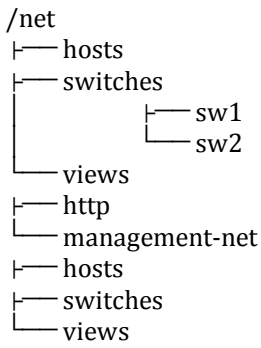

Fig IV.1: The Yanc File System Hierarchy.

- Switches – Switch is a rough entity and each switch will be represented with its own directory. Switches can be created, deleted, and renamed with the standard file system calls (mkdir (), rmdir (), and rename (), respectively). There is no need of removing children of the object before removing the object, hence the rmdir () automatically removes the children before the object itself.

- Ports and Topology – Each switch has a set of ports. Each port is represented with a directory and within the directory are files with information about the given port. Symbolic links are used by yanc for manipulation and re-organization of directory structure while still transporting important link information [4]. A symbolic link named peer is contained in each link, which may or may not exist. Physical links are represented by pointing the peer at another port. Pointing the peer to anything else other than a port is considered an error.
- Flows – A flow entry is an entry in a table which shows how the switch should handle traffic, OpenFlow and many other protocols use the same technique. Many pieces of information are present in flows, each yanc flow is represented by a directory. Each matched field is a separate directory. If a file is not matched it is called a wildcard. File named action.* describe actions and files such as priority are used for setting priorities of a flow entry. Version file is used to set multiple values atomically. Applications use inotify or fanotify to check the yanc file system for new, changing or deleted flow entries. Another application may modify the flow written by another application before its written-on hardware for performance, security or monitoring purposes.
- Packet In – Applications must be notified of events started on switches. I/O file also handles events in yanc. Drivers handle specific events relating to the switch hardware status. The packet-in message in open flow is the main event for network management applications. Table-misses are the main reason for packet-in events, but they are sometimes also caused by explicit rules to trigger software processing. When one application needs to change a packet-in before it is received by another application views are used. Each application interested in packet-in events creates a directory in the events/ subdirectory (creating a private buffer). New packet-in messages appear as sub-directories within each private buffer, with files of information regarding the message.

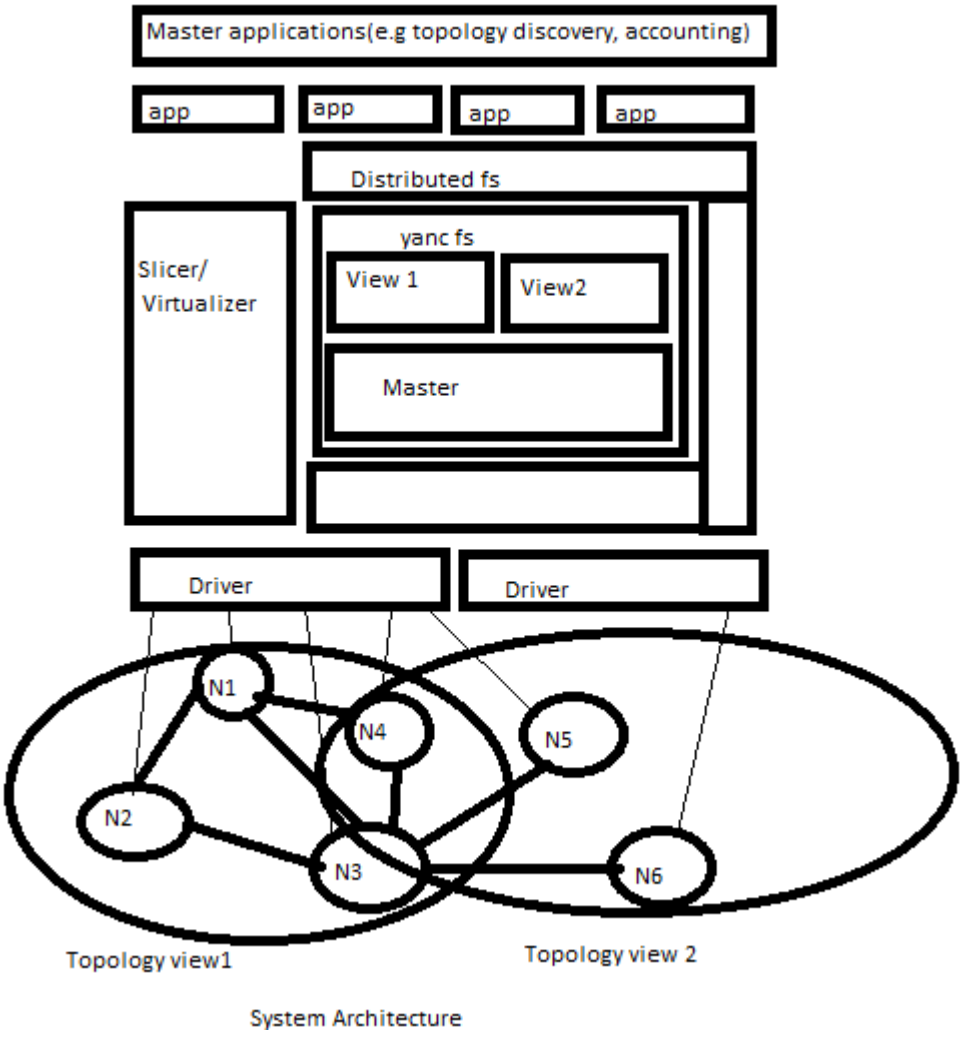

Fig IV.2: The Yanc File System Architecture.

V. CONCLUSION

The yanc was designed by keeping operating system in mind, as in how operating system mechanisms and principles can be applied to SDN. Yanc takes advantage of network operating system which can be used as a hold in transformation of operating system. Thus, more focus can be put on specific control-plane-centric topics such as load balancing, congestion control, and security. This prototype is only a introductory stride toward realizing many of the aspects of yanc that we introduced in this paper.

VI. REFERENCES

1. Network functions virtualization. In SDN and OpenFlow World Congress Oct.22-24 2012.
2. R. Sherwood, G. Gibb, K.-K. Yap, G. Appenzeller, M. Casado, N. McKeown, and G. Parulkar. Can the production network be the testbed? In Proc. USENIX conference on Operating systems design and implementation (OSDI), 2010. "https://www.sdxcentral.com/sdn/definitions/sdn-controllers/open-source-sdn-controllers/."
3. Matthew Monaco, Oliver Michel, Eric Keller in University of Colorado at Boulder Applying Operating System Principles to SDN Controller Design Nov 23, 2013.
4. Yanc: Yet another Network Controller by Alex Tsankov. "Yanc: Yet Another Network Controller | Alex Tsankov - Academia.edu"
5. Introducing ONOS - a SDN network operating system for Service Providers. "Microsoft Word - Whitepaper-ONOSv1-Brian.docx - Whitepaper-ONOS-final.pdf"
6. Onos: Open network operating system. "http://es.slideshare.net/umeshkrishnaswamy/" open-network-operating-system."ONOS - A new carrier-grade SDN network operating system designed for high availability, performance, scale-out."
7. Z. Cai.Maestro: Achieving Scalability and Coordination in Centralized Network Control Plane. PhD thesis, Rice University, 2011. Software Defined Networking "https://en.wikipedia.org/wiki/Software-defined_networking."
8. https://community.arm.com/processors/b/blog/posts/an-introduction-to-sdn-software-defined-networking

