IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 4, April 2017

Enhancing Traffic Management and Load Balancing in SDN

Prof. Swati Joshi[1], Arnab Dutta[2], Neeraj Chavan[2], Gaurav Dhus[2], Pratik Bharsakle[2]

Associate Professor, Department of Computer Engineering, Sinhgad College of Engineering, Pune, India[1]

Student, Department of Computer Engineering, Sinhgad College of Engineering, Pune, India[2]

Abstract: This document The Internet has connected many of the devices in the world. By the end of 2015, we already had9 billion connected things. And some predict that by 2020, the number of Internet-connectedthings will reach or even exceed 50 billion. Many network connected devices can't be updated.Considering the range of applications, it's impossible to determine exactly which services areback-ending any particular product or service. This is the problem in a nutshell: One can'tdetermine exactly where the data is being processed and stored.Load balancing and Traffic management methods are adopted in the design of switching androuting algorithms. However, the requirement of load balancing in conventional networks cannotbe fully satisfied by traditional approaches. The quantityof data flowing in these systems isconsiderably high. The network topology and switching mechanism can affect the performanceand latency significantly. The layer 2 switching has some significant disadvantages which includelack of hardware which increase their susceptibility to broadcast storms. The main reasonis the lack of efficient ways to obtain network traffic statistics from network devices. Propertraffic management and load balancing will help take some extra effects towards the cause ofmaking data flow easier. SDN separates the control plane from the data plane and this centralizedcontrol of data layer and control layer will make the management of data packets easier.As a solution, applications to manage the traffic and balance the load at switch based uponSDN architecture is developed to provide improved routing and networking performance fornetwork.

Keywords: Software-Defined Networking (SDN), controller, OpenFlow, OpenDayLight.

I. INTRODUCTION

The adoption of software-defined networking (SDN) initiative has dramatically changed thenetworking landscape over the past few years. SDN deployment will enable devices to sharenetwork resources efficiently and reliably, and further cut hardware investment.

A. **OpenFlow:** OpenFlow is a protocol that enables network controllers to determine the path of network packets across a network of switches. The controllers are distinct from the switches. This separation of the control from the forwarding allows for more sophisticated traffic management. The Open Networking Foundation (ONF), a user-led organization dedicated to promotion and adoption of software-defined networking (SDN), manages the OpenFlow standard. ONF defines OpenFlow as the first standard communications interface defined between the control and forwarding layers of an SDN architecture. It is the absence of an open interface to the forwarding plane that has led to the characterization of today's networking devices as monolithic, closed, and mainframe-like. A protocol like OpenFlow is needed to move the network control out of proprietary network switches and into control software that's open source and locally managed [1].

B. **Software Defined Networking (SDN):**Software-defined networking (SDN) is an approach to computer networking that allows network administrators to manage network services through the abstraction of lower-level functionality [1, 2]. SDN is meant to address the fact that the static architecture of traditional networks doesn't support the dynamic, scalable computing and storage needs of more modern computing environments such as data centres. This is done by decoupling or disassociating the system that makes decisions about where traffic is sent (the control plane) from the underlying systems that forward traffic to the selected destination (the data plane) [1]. The Software Defined Networking method centralizes control of the network by separating the control logic to off-device computer resources. All SDN models have some version of an SDN Controller, as well as southbound APIs and northbound APIs:

a. **Controllers:**The 'brains' of the network, SDN Controllers offer a centralized view of the overall network, and enable network administrators to dictate to the underlying systems (like switches and routers), how the forwarding plane should handle network traffic.

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 4, April 2017

- **OpenDayLight:** OpenDaylight is a highly available, modular, extensible, scalable and multi-protocol controller infrastructure built for SDN deployments on modern heterogeneous multivendor networks. OpenDaylight SDN Controller has several layers. The top layer consists of business and network logic applications. The middle layer is the framework layer and the bottom layer consists of physical and virtual devices. The middle layer is the framework in which the SDN abstractions can manifest. This layer hosts northbound and southbound APIs. The controller exposes open northbound APIs which are used by applications. OpenDaylight supports the OSGi framework and bidirectional REST for the northbound API. The business logic resides in the applications above the middle layer. OpenDaylight provides a model-driven service abstraction platform that allows users to write applications that easily work across a wide variety of hardware and southbound protocols. These applications use the controller to gather network intelligence, run algorithms to perform analytics, and then use the controller to orchestrate the new rules, if any, throughout the network [3].

b. **Southbound APIs:** Software-defined networking uses southbound APIs to relay information to the switches and routers below. OpenFlow, considered the first standard in SDN, was the original southbound API and remains as one of the most common protocols. Despite some considering OpenFlow and SDN to be one and the same, OpenFlow is merely one piece of the bigger SDN landscape.

c. **Northbound APIs:** Software Defined Networking uses northbound APIs to communicate with the applications and business logic above. These help network administrators to programmatically shape traffic and deploy services.

C. **Mininet Emulator:** Mininet is a software emulator for prototyping a large network on to a single machine [4]. Mininet emulator is an inexpensive and quickly configurable network testbed. So far, it is the most well-known tool supporting SDN OpenFlow network research emulator, as observed from the ONS 2013 conference [3]. It allows the user to quickly create, interact with, customize and share a software-defined network (SDN) prototype to simulate a network topology that uses OpenFlow switches. Mininet uses virtual hosts, switches, and links to create a network on a single OS kernel, and uses the real network stack to process packets and connect to real networks. In addition, Unix/Linux-based network applications are also allowed to run on virtual hosts. In an OpenFlow network emulated by Mininet, a real OpenFlow controller application can run on an external machine or on the same machine where virtual hosts are emulated.

II. RELATED WORK

Traffic Management and Load Balancing is an important subject for network performance optimization by dynamically analysing, predicting, and regulating the behaviour of the transmitted data [6]. Today's Internet applications require the underlying network architecture to behave in real time (or near real time) and to scale up to a large amount of traffic. Highly efficient network management is desirable to significantly improve resource utilization for optimal system performance. However, all the existing Traffic Engineering technologies rely on closed and inflexible architectural design, where the control and data planes are tightly coupled and integrated. Such inflexible and close architectures prevent the existing Traffic Engineering technologies providing truly differentiated services to adapt, to increasingly growing, uneven, and highly variabletraffic patterns. On the contrary, the unique features of SDN, including visibility, programmability,openness, and virtualizability, pave the way for the development of new Traffic Engineeringtechniques that are inherently flexible, adaptive, and customizable [2].

Load Balancing is a technique used to distribute large number of requests across multiple devices.Load balancer increases the network performance by properly using the available resourcesand helps in improving response time and transactions per second. The hash-basedEqual-Cost Multi-Path (ECMP) is a load balancing scheme to distribute flows across availablepaths using flow hashing methods [6]. However, a main constraint of ECMP is that two or more large, long-lived flows can collide on their hash and thus share the same output port, therebycreating a bottleneck in the network. OF switches use flow-match wildcards to aggregate trafficflows [6]. OF is a great concept that simplifies network and traffic management by enablingflow-level control over switches and providing a global network view [1]. However, the centralcontrol management and the global view of overall flows require the controller to set up all flows for the critical path in the entire network, which is not sufficiently scalable and results inboth bottleneck and latency. In Round Robin Strategy, the requests are equally distributed to theservers. The requests are assigned on round robin basis [7]. However, it does not always resultin the most accurate or efficient distribution of traffic, because many round-robin load balancersassume that all servers are the same: currently up, currently handling the same load, and withthe same storage and computing capacity.

To reduce the number of interactions between the controller and the switches, the SDN TrafficEngineering approaches implement wildcard OF rules at the switches, and the switches canmake local routing decisions that handle mice flows to

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 4, April 2017

avoid involving the controller, while thecontroller maintains the control over only targeted elephant flows, especially for quality of service(QoS)-significant flows. To fully utilize the flexible control and global view promised bySDN, SDN Traffic Engineering demands a dynamic load balancing mechanism that is adaptiveto time-varying network states and adjustable based on fine-grained traffic characteristics suchas path cost i.e. transmission cost of payload at switch level [6]. How to significantly reduce thenetwork overhead when SDN controller(s) or monitoring devices collect the network statisticswith high accuracy is a topic for further study.

An SDN controller has complete view of the network. The controller also has the ability to change the network structure and services at run time. Several SDN controllers have beendeveloped and deployed. As of now, almost all the SDN controllers are based on OpenFlowprotocol. SDN controllers can have different properties. Selecting a controller using a singleproperty is trivial [8].

However, selecting a controller based on multiple properties is a Multi-Criteria Decision Making problem. Among them, the top five list the SDN controllers are:POX, Ryu, Trema, FloodLight, and OpenDaylight [8]. OpenDaylight is an open source projectsupported by IBM, Cisco, Juniper, VMWare and several other major networking Vendors [3].OpenDaylight is an SDN controller platform implemented in Java. As such, it can be deployedon any hardware and operating system platform that supports Java.

For evaluation of the performance of a network efficiently, instead of building a large experimental testbed, there are two common methods — simulation and emulation, to mimic theenvironment of an actual network without real deployment [9]. The former approach uses asoftware program to execute the operations of real devices and the interactions between them.Running a simulation is inexpensive, flexible, controllable, and scalable than using real devicesto run real operating systems and applications. Two tools that can simulate or emulate an OpenFlow network are: EstiNet, which can be used as a simulator or an emulator, and the otheris Mininet emulator [4].

EstiNet emulator uses its unique methodology, “kernel reentering,”to run simulations and emulations [10]. EstiNet emulator shows its correctness, accuracy, andscalability but it needs more time to simulate more OpenFlow switches. Mininet emulator generallyworked well but comparatively the EstiNet ousted the Mininet. Considering trade-offsthe Mininet emulator is an open-source software and is available free of cost while EstiNetemulator is a paid software.

III. METHODOLOGY

The project consists of two modules viz. Traffic Manager, Load Balancer.

A. **Traffic Manager:** The traffic manager component eyes on mapping of the MAC address of the target node to the appropriate Egress connector ID at switch. For this it maintains the MAC table and performs lookup at it for every incoming flow. The operational flow of traffic manager module as shown in Fig. 1 is discussed as follows:

a. A packet_in is sent from the switch to controller
b. The OpenFlow plugin parses the packet and translates it into an MD-SAL formatgoverned by the OpenFlow protocol model; the OpenFlow 1.3 Plugin creates anMD-SAL Notification and publishes it into the MD-SAL.
c. MD-SAL routes the notification to all registered consumers, in this case the TrafficManager application
d. If the Traffic Manager application does not yet have the mapping between the macaddress and the port, it floods the packet. Flooding is done as pkt_out to the switch. The application issues an MD-SAL RPC (with input parameter that constitutes thepkt_out PDU payload).
e. MD-SAL routes the RPC to the OpenFlow 1.3 Plugin.
f. The OpenFlow 1.3 Plugin processes the packet, which is eventually translated intoan OpenFlow pkt_out PDU.
g. The OpenFlow pkt_out PDU is sent to the switch.
h. When the application wants to program a flow, it creates a new flow in the MD-SAL.A new flow is created in the appropriate table, which is in the appropriate flowcapablenode (i.e. switch, openflow:1 in the example), which is in the config space ofthe MD-SAL inventory database. The inventory database is a subtree (namespace)of the MD-SAL database.
i. MD-SAL generates a change notification, which is received by the Forwarding Rules Manager (FRM). The FRM listens on flow updates - i.e. it registered for MD-SAL change notifications on the config namespace subtree that corresponds to flows.

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 4, April 2017

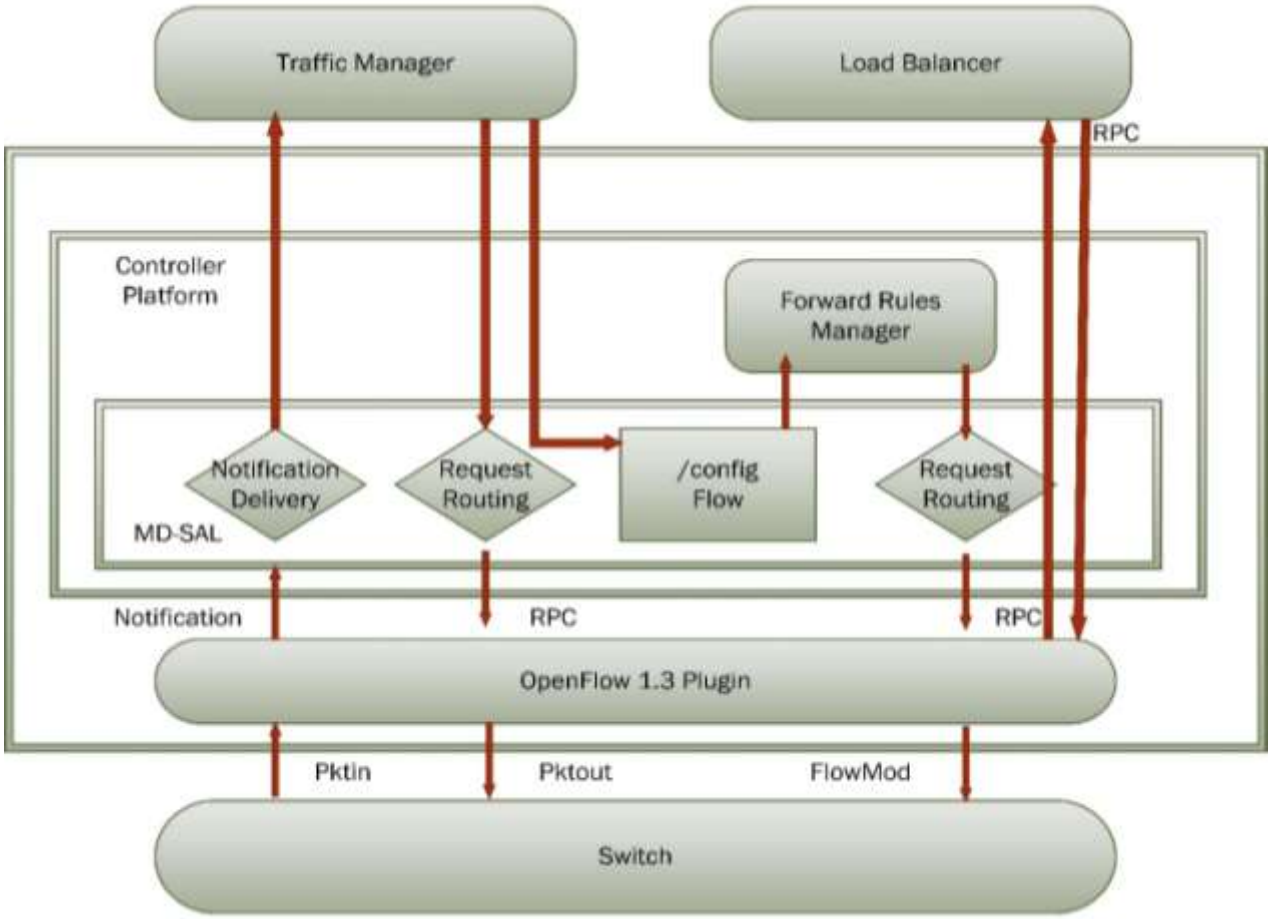

Fig.1 Operational Diagram

The FRM programs the flow. It issues an MD-SAL RPC flow programming operation.

j. MD-SAL routes the RPC to the OpenFlow 1.3 Plugin.
k. The OpenFlow 1.3 Plugin processes the packet, which is eventually translated intoan OpenFlow FlowMod PDU in the OpenFlow Java library.
l. The OpenFlow FlowMod PDU is sent to the switch.

B. **Load Balancer:** The load balancer component has task of optimizing the network resources. In order to meet this objective the component module needs to keep track of the current network context and find the optimal path for the traffic between the two hosts.

The operational flow for Load Balancer module as shown in Fig. 1 is as follows:

a. The administrator provides host IDs of the two devices between which to balancethe load.
b. Load Balancer sends RPC to controller to access the current network statistics.
c. The Dijkstra's algorithm is used to compute all possible shortest between the two hosts.
d. The shortlisted path are then sequenced using transmission cost of each path. Lesser the cost, optimal the path.
e. The optimal path is then updated to controller via RPC.

IV. RESULTS AND DISCUSSION

The proposed system is fully implemented and is ready to be deployed. The ping time before the installation of traffic manager module is much greater.

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 4, April 2017

Fig.1 Ping Time Before and After Traffic Manager Application

Fig.2 Bandwidth Before and After Traffic Manager Application

When the traffic manager module is installed, the ping time is reduced by 40-60%. The jitter in the network is reduced considerably as shown in the Fig. 1. In the load balancer the payload is routed optimally and there is no packet flow between unintended hosts. This results in a balanced network. The bandwidth achieved between the hosts before installation of our system was 110-150 Mbps and after installation it increases drastically to about 16-17 Gbps as depicted in Fig. 3. The increase in bandwidth can be used to manage network having high volumes of data traffic. As the ping time is reduced we can expect faster responses from the servers. Our system helps in managing the network traffic dynamically by balancing the load at respective switches and making the network congestion free.

V. CONCLUSION

SDN represents a new, flexible, and open architecture that allows the dynamic and timely regulationof the behaviour of network switches in complex and large-scale computer networks. AsSDN is accelerating the innovation and evolution of modern data networks; it requires a highlyscalable and intelligent Traffic Management and Load Balancing system. Applying the novelconcept of SDN, the response time is reduced. The network traffic is handled by the LoadBalancer which in turn optimizes the resources in the network.

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 4, April 2017

ACKNOWLEDGMENT

We would like to express our sincere gratitude towards our guide **Prof. S. A. Joshi** for her invaluableguidance and supervision that helped us in our research. She has always encouragedus to explore new concepts and pursue newer research problems. We credit our project contribution to her. Collectively, we would also like to thank our project committee members **Prof.H. A. Bhute** and **Prof. V. M. Manga** for their time, suggestions, and for graciously agreeing tobe on our committee, and always making themselves available. We cannot thank them enough.

REFERENCES

[1] "Onfsdn evolution - a white paper," 2016.
[2] D. Kreutz, F. M. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," Proceedings of the IEEE, vol. 103, no. 1, pp. 14–76, 2015.
[3] "The opendaylight platform | opendaylight." https://www.opendaylight.org/.
[4] "Mininet: An instant virtual network on your laptop (or other pc) - mininet." http://mininet.org/.
[5] "Maven – welcome to apache maven." https://maven.apache.org/.
[6] F. Akyildiz, A. Lee, P. Wang, M. Luo, and W. Chou, "Research challenges for traffic engineering in software defined networks," IEEE Network, vol. 30, no. 3, pp. 52–58, 2016.
[7] S. Kaur, K. Kumar, J. Singh, and N. S. Ghumman, "Round-robin based load balancing in software defined networking," in Computing for Sustainable Global Development (INDIACom), 2015 2nd International Conference on, pp. 2136–2139, IEEE, 2015.
[8] R. Khondoker, A. Zaalouk, R. Marx, and K. Bayarou, "Feature-based comparison and selection of software defined networking (sdn) controllers," in Computer Applications and Information Systems (WCCAIS), 2014 World Congress on, pp. 1–7, IEEE, 2014.
[9] S.-Y. Wang, "Comparison of sdn openflow network simulator and emulators: Estinet vs. mininet," in Computers and Communication (ISCC), 2014 IEEE Symposium on, pp. 1–6, IEEE, 2014.
[10] "Estinet technologies inc.."http://www.estinet.com/.
[11] T. D. Nadeau and K. Gray, SDN: Software Defined Networks: An Authoritative Review of Network Programmability Technologies. " O'Reilly Media, Inc.", 2013.