Advances in Computer Science and Information Technology (ACSIT)
p-ISSN: 2393-9907; e-ISSN: 2393-9915; Volume 3, Issue 3; April-June, 2016, pp. 220-224
© Krishi Sanskriti Publications
http://www.krishisanskriti.org/Publication.html

A Hybrid Channel Access Method to Optimize Congested Switched Network

Shilpa Mahajan[1] and Nisha Sharma[2]

[1]Computer Science And Engineering North Cap University Gurgaon, Haryana
[2]Mtech Student Computer Science And Engineering North Cap University Gurgaon, Haryana
E-mail: [1]shilpamahajan@itmindia.edu, [2]nishasharma21.91@gmail.com

Abstract—*From, the last few years the network acquirement is shifting frequently due the quantity of traffic have been mounting exponentially and additional challenging end-to-end goals are imposed for completion. On the other hand, due to unbothered networks architectures the complication and hindering of the design have been increasing resulting in reduced utilization of resources. In the most recent time, a latest networking loom called Software-Defined Networking(SDN) is budding quick which is based on the division of data and control planes. Such, loom helps the network supervisor to encompass additional dynamic run of the network behavior.*
The principle of this paper is to examine the potential which SDN provides to increase resources distribution beside the network when supplementary route are present linking source and destination.

Keyword*: SDN, loom,dynamic*

1. INTRODUCTION

This section describes about the common sight of the presented networks, and how latest developing technologies can facilitate to progress the network potential in order to attain a more elastic loom and adjust to today's requirements. In order to adjust to the fresh requirements, latest network paradigms such as Software-Defined Networking (SDN), cognitive networks or involuntary networks are budding quickly due the benefit of carriers and Internet Service Providers ISPs.

The initial effects to do is to analyze which are the harms of the presented networks as they have turn out to be a obstacle to create fresh, new services, and an even superior obstacle to the sustained development of the Internet. As protocols are liable to be distinct in separation and solves a particular problem without the profit of any basic thought resulted in one of the prime restrictions of today's networks: complexity. For example, to add much mechanism, IT must handle numerous switches, routers, firewalls, Web authentication portals, etc. In addition other factors like network topology, vendor switch model, and software version all ought to be taken into description. Due to all these type of complexity, today's networks are comparatively fixed as IT seeks to minimize the threat of service interruption. The claim that currently wanted in the diverse services that user insists, such as VoIP or streaming video in High Quality, where beyond belief when the architecture was designed.

All the above factors have lead to greater provisioning of the networks, raising the price, and wasting resources due the complexity to optimize, organize and adjust the physical resources existing to the necessities of every moment. So its moment to move forward and advance to a finest architecture, easy to handle, develop and understand.

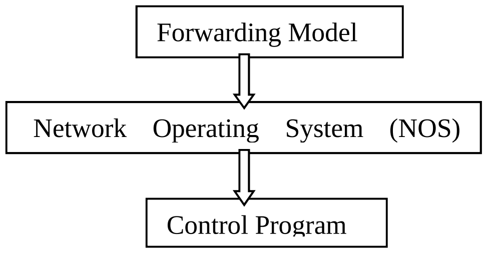
All the above issues can be solved by SDN loom which have a apparent abstraction layers and a centralized control plane, it's much easier to make an additional and dynamic administration and control of the network due to universal outline of the network, and control over its totally. The inspiration of this paper is to take benefit of the fresh centralized networking loom of SDN to build up a load balancing algorithm which will adjust the route of every flow depending on the present condition of the network in order to attain an enhanced resource sharing, dropping the overall cost and adjusting its behavior to the traffic enlargement [8, 9].

The rest of the paper is organized as follows:

Section 2 describes about the SDN and Open Flow Protocol. Dynamic Load Balancing Algorithm applied to SDN is presented in Section 3 and, finally Section 4 provides the conclusions and some suggestions for future work.

2. DESCRIPTION OF SDN AND OPENFLOW PROTOCOL

SDN is a modern network technology that offers high interoperability and cost competent behavior for client control and network programmability [1]. It provide an abstraction of the forwarding task separating the data plane from control plane, resulting in a liberty to supervise diverse topologies, protocols without numerous limitations from the physical layer.i.e why huge companies, such as Google, are already using it[2,3]. In SDN the control plane is divided into three major layers:



Forwarding model: mainly consists of the Network Elements (NE) (i.e. switches) and a committed communicated control from every NE with the NOS defining the forwarding condition in a ordinary approach.

Network Operation System: portion of software running in servers (controllers) which gives information about the present status of network such as the topology or the state of each port.

Control program: along with operative goal it computes the forwarding state of each NE.

Advantages:

SDN try to progress the recent networks. The major advantages of the SDN paradigm are:

OPEX drop: due to administrative control physical communication with the hardware is reduced thus enhancing the uptime of the network.

CAPEX drop: decoupling the data plane from the control plane provides effortless hardware and brings more competition between hardware manufacturers, as the devices don’t depend on the proprietary software.

Quickness it helps network to adjust fastly with changes like failures or latest traffic pattern as the control layer can work together with the infrastructure layer continuously. And many more.

2.1 SDN Controllers

Currently many projects are running in SDN controllers [4], and the selection of the controller can be based on many aspects, such as the programming language, the current activity on its development, the amount of documentation or the set of built-in components.

“Table 1” shows a summary of features of the most known controllers, specifying the language in which they are written and other factors.

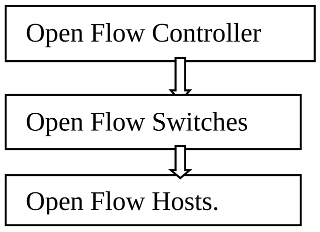
Table1: Different Types and Features of SDN Controller

Feature	Ryu	POX	Beacon	Floodlight	Daylight
Written In	Python	Python	Java	Java	Java
Supported Language	Python	Python	Java	Python, Java	Java
Actively Developed	Yes	Yes	No	Yes	Yes
Rest API	Yes	Yes(limited)	Yes	Yes	Yes
Well Documented	Yes	No	Yes	Yes	Yes
Utility Function	Yes	No	No	Yes	No

2.2 OPENFLOW PROTOCOL

In the non SDN network the forwarding(data path) and routing (control path) decision are made by switches and routers only while in an Open Flow switch, this mechanism of decision making is separated among the switch and the controller. Due to this separation the data path on the network element, and advanced routing are moved to the NOS. Thus, we can say that Open Flow is the protocol which allows communication between the controller and all the switches. It is the most frequent protocol used in SDN networks, and is frequently puzzled with the SDN concept, but they are diverse things. As SDN is the structural design isolating the layers and Open Flow is only a protocol used to transmit the messages from the control layer to the network fundamentals [5, 6].

Open Flow network composed of three components:



Where, Open Flow Controller send message to switches, keep informed flow tables, spot network congestion and reroute the network path and it should be noted that every switches maintain a flow table having routing information [7].

3. DYNAMIC LOAD BALANCING ALGORITHM APPLIED TO SDN

The main aim is to balance dynamically the load depend upon the traffic condition in order to achieve the best resource profit. The procedure starts with following stages:

Table 2: Stages of dynamical load balancing algorithm

STAGES	DESCRIPTION
New Flow Detected	When packet arrive at switch its header is matched with the rules of the switch and if not matched trigger flow various step of algorithm 1.

Rerouted Flow	It has two things to do: • To find the path where to reroute the traffic when congestion is detected. • What to do if no path is found to reroute the traffic and rerouting is done where there is lowest traffic.

For description of any algorithm, we need to reveal the various data structures drawn in it. Such structure helps in achieving a competent load balancing together with reduction of computational cost and time. The four foremost data structures are explain below:

DATA STRUCTURES	**DESCRIPTION**
Flow	It specifies a definite traffic flow from one host to another one. The structure is as follows: ***Flow={<FID>,<SIP>,<DIP>,<SPort>,<DPort>,<UBndwdth>}*** where**,** **FID(Flow ID):** recognize every flow with a single ID. **SIP(Source IP):** it contain the IPv4 of the source host who initialized a flow. **DIP(Destination IP):** IPv4 of the destination host. **SPort(Source Port):** port number of the source. **DPort(Destination Port):** port number of the destination. **UBndwdth(Used Bandwidth):** broadcast speed of a definite flow, in Mbps.
Flows Collection	Bundle of flows are put jointly in collections, holding a numeral of flows with familiar trait (e.g flows going by identical link). ***FlowsCollection = {< Flow1 >,< Flow2 > ... < Flown >}***
Path	It is composition of the information about a particular path linking two hosts, and poised as shown below: ***Path= {<PID>, <Hops>, <Links>, <Endpnts>,<Cpcty>,<Flows>,<UBndwdth>, <FCpcty>}*** where, **PID(Path ID):** gives every distinct feasible path with a unique ID. **Hops:** contain a identifier of every of the switches contained by the path. **Links:** provides all the links concerned in the path. Every of the links is collected by a couple of Switch-Port identifiers. **Endpnts(Endpoints):** identifier of the switch with the source and destination host is coupled to. **Cpcty(Capacity):** indicate the highest capability of the path. Which correspond to the capability of the link with minimum capability beside the path. **Flows:** listing the flows which are routed throughout this path. **FCpcty(FreeCapacity):** capacity offered in this path. Is the lowest amount of capacity of the *Links* that figure the path?
Paths Collection	Listing of all the potential paths of the hosts that have initiated a contact among them, and information about every of the paths. ***PathsCollection = {< Path1 >,< Path2 > ... < Pathn >}***

3.1 Algorithm Description

The endeavor of the algorithm is to equilibrium dynamically loads depending on traffic environment in order to attain the finest resource return. In order to achieve such objective, it is crucial to maintain footpath of the existing state of the network in terms of traffic.

3.1.1 New flow detected

This is the primary step of algorithm and works through the following various steps:

STEP1: In this step, it is checked whether a packet header which arrive at the switch in the network matches with any of the convention that the switch has or not, if not then it will generate the Algorithm 1(“see Figure1”)

STEP 2: In this step, we find all the potential paths among the points which have been computed already, to avoid recalculation for the entire path. If in the case that the paths linking the endpoints of the switches have not been compute yet, a role to determine is trigger. For every fresh path exposed, a fresh *Path* is store into the *Path Collection*, having as a only one of its kind identifier through which the path goes. After finding all the potential paths the next step is to:

STEP3: Selection of a route, and writing all the convention in the flow tables of every switch inside the preferred route. For route selection, the algorithm look for the Path with superior *Free Capacity* cost. After the route selection process, and using the *Hops* and *Links* of the definite *Path*, it is desirable to drive the appropriated communication to the switches to promote the packets all the way through that path. Therefore, each switch within the selected route will have the compulsory flow entry to carry out the message between the two end points.

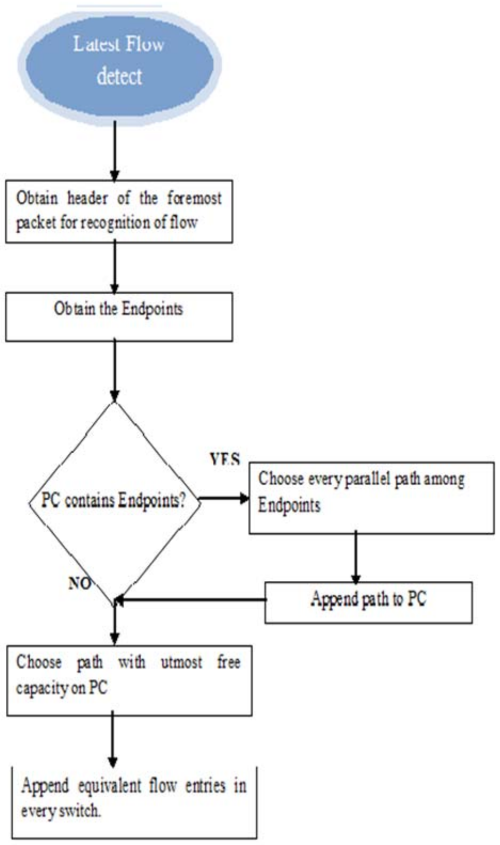

Latest Flow detect
Obtain header of the foremost packet for recognition of flow
Obtain the Endpoints
PC contains Endpoints?
YES
Choose every parallel path among Endpoints
Append path to PC
NO
Choose path with utmost free capacity on PC
Append equivalent flow entries in every switch.

Fig. 1: Flow chart for new flow detected (algorithm 1)

3.1.2 Rerouting flows

This is the second step of algorithm and its main aim is: To find the path where to reroute the traffic when congestion is detected. For finding the path various steps are followed ("see Figure 2"):

STEP 1: Detection of the overcrowded link and then we move forward by establishing a *Flows Collection* having all the flows that are with that link. Consequently, the flow with minimum bandwidth convention is selected (named as *Pending Flow*),

STEP 2: In this step it is checked whether some additional equivalent route for this flow with adequate free capacity to hold its traffic exists or not .If yes then the flow is routed all the way through that path transferring the consequent flow entries to every of the Open Flow switches are repeated, stirring the lightest flows along a different paths.

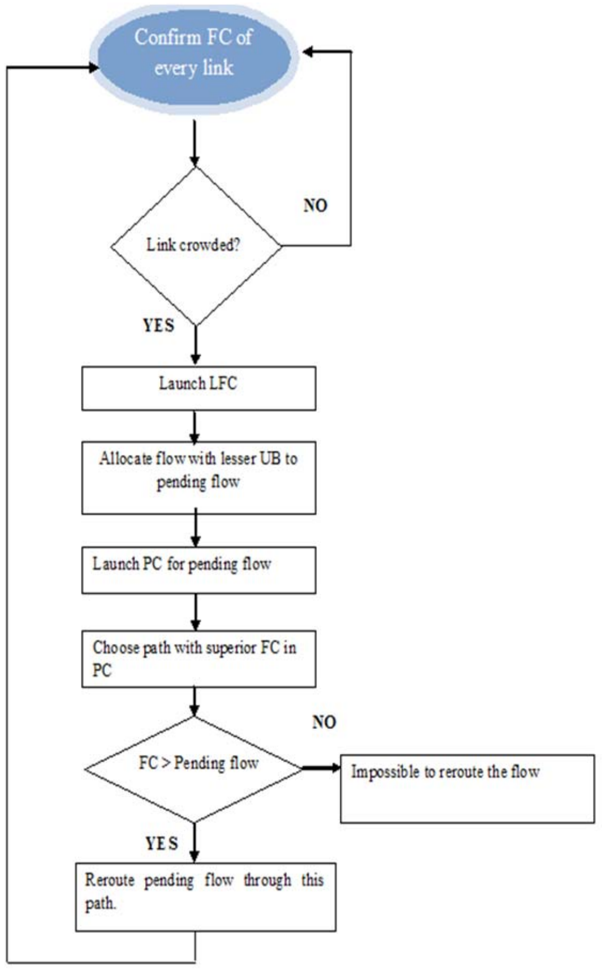

Confirm FC of every link
Link crowded?
NO
YES
Launch LFC
Allocate flow with lesser UB to pending flow
Launch PC for pending flow
Choose path with superior FC in PC
FC > Pending flow
NO
Impossible to reroute the flow
YES
Reroute pending flow through this path.

Fig. 2: Flow chart for Rerouting flows

STEP 3: In this the entire process starts again, and, if the congestion still exist, above two steps are repeated, stirring the lightest flows along a different paths.

4. CONCLUSION AND FUTURE WORK

Using the algorithm, we can easily allocate the dissimilar traffic flows approved by a network through the dissimilar parallel routes among source and destination. The algorithm has been implemented over a Software-Defined Network, demanding to discover the capability which brings to us innovative paradigm of networking. SDN provide a sight of every basic of the network as well as control over them. Thus, allowing a dynamic power of the procedures to be done in each promising circumstances. Future work will be dedicated to exploit the total throughput of the network when there is more than one path between the source and destination and what are the effects of rerouting a flow in terms of throughput, delay, jitter and packet losses? It is possible to maintain the quality parameters within a certain margin using mininet and floodlight controller[10,11].

REFERENCES

[1] N.McKeown et al, “OpenFlow Innovation in Campus Networks,” in SIGCOMM Computer. Communication.Review.,vol 38, no 2, pp.69-74,Mar. 2008

[2] (2012). Inter-datacenter wan with centralized te using sdn and openflow. White paper, Google, Inc.

[3] Sushant Jain, Alok Kumar, S. M. (2013). B4: Experience with a globally-deployed software defined wan. Technical report, Google Inc.

[4] Khondoker, R., Zaalouk, A., Marx, R., and Bayarou, K. (2014). Feature-based comparison and selection of software defined networking (sdn) controllers. In International Conference on Computer Software and Applications, 2014 Proceedings of.

[5] H.Kem and Feamster: **“**Improvement in network management with SDN**”**:2013 Communication magazine IEEE,vol.15.

[6] Y.Jarraya ,T.Manadi,M.Debbabi: **“**A survey and layered taxonomy of SDN**”**:2014 Communication surveys tutorials IEEE.

[7] Open network foundation, “OpenFlow Switch Specification,”version 1.0.

[8]Fang, S., Yu, Y., Foh, C. H., and Aung, K. M. M. (2012). A loss-free multipathing solution for data center network using software-defined networking approach. IEEE.

[9] Egilmez, H. E., Dane, S. T., Bagci, K. T., and Tekalp, A. M. (2012). Openqos: An openflow controller design for multimedia delivery with end-to-end quality of service over software-defined networks.

[10] Mininet(http://mininet.org).

[11] FloodLight(http://www.projectfloodlight.org/floodlight).