ABSTRACT:

SDN is a new approach to the current world of networking. SDN varies from other kinds of networks because of the separate the network control plane from the data plane in it. The control plane constitutes the controller and hence the control plane is called as the intelligence of the network. The data plane constitutes the switches, routers and other networking devices that are commonly used in networking. Software Defined Networking (SDN) mainly concentrates on providing an open and programmable network where the administrators are allowed to design and implement their own algorithms and load balancing strategies. In SDN, the switches carry out packet forwarding while decision making is done by the controller since the controller is the intelligence (brain) of the SDN network. OpenFlow protocol is used to carry out the messages between switches and the controller. It is a communication protocol through which the controller directs the route of the network packet via the switches. In the research article, mainly employed Dijkstra's algorithm for finding the best path among the multiple paths of the same length based on the transmitted data rate using Open Daylight Controller.

Keywords: Software Defined Networking (SDN), Open Flow, Open Daylight Controller, Mininet.

1. Introduction

There are a lot of developments in the current technology of networking. Various kinds of networking techniques have been adapted recently since the earliest kind of networking i.e. traditional networking and centralized networking has been very complicated and difficult to understand and administrator. In case of a traditional network, it consists different set of protocols for network layer and hardware devices. The hardware devices and protocols have been strongly integrated into modular fashion that causes a major problem in the implementation of new network policies(conflict-free and innovative). Whereas centralized network, the control plane consists of an overall view of network topology information and status update[4]. The network administrator is responsible for configuration of each and every network device [3]. The devices used to provide network services in a traditional network are vendor locked in and they are not flexible to changes, very expensive and requires high operational resources for running
the network[5][9]. Since all the networking devices are not from the same vendor it becomes even very complex to operate such different products of different vendors [5][9].

To overcome these complexities a new routing technique called Software Defined Networking (SDN) has been evolved. It is believed to provide a conflict-free confined set of policies for the communication within and between different network devices and applications [1]. Generally forwarding and routing devices i.e. switches and routers are tightly integrated as controlling and forwarding logic into a single plane. And this kind of distributed network management creates complexity [1]. To reduce the management complexity SDN separate divides the entire plane into two different planes as control plane and data plane where the control plane acts as a decision make in routing while data plane acts as forwarding device which forward packets based on a defined set of rules and policies that are decided in the control plane.

SDN provides an open, standard and programmable platform on which the researchers will be able to run their own and new approaches in production networks. The SDN controller resides in the control plane and it determines the forwarding path through which the packet transfer takes place in the network. The data plane consists of networking devices like switches, routers and other basic network appliances [1]. SDN is basically an experimental platform where the researchers are provided with the opportunity to create, manage (change) and implement their own network environment. SDN architecture provides a global view of its network environment with three basic layers namely Application Layer, Control plane Layer & Data plane Layer [1].

ARCHITECTURE OF SDN

![SDN Architecture Diagram](image-url)
The interaction between these integrating layers is done through APIs, called as Northbound Interface and Southbound Interface APIs[1][2][9]. The communication between Control plane and Application Layer takes places through Northbound API whereas the communication between Control plane and data plane layers takes place through Southbound API with the help of Open Flow protocol. Basically, the instructions are given to the data plane about certain rules or policies by Open Flow protocol over a secure channel decided by the controller [4][9]. Open Flow ensures the implementation of the policies defined by the controller on the basis of information gathered from the network [3][9][11].

2. Related works in SDN

Various routing techniques have been evolved in SDN concentrating on various issues. Some routing techniques are developed to achieve scalability while some other techniques concentrate on QOS, the multipath deduction for routing in order to reduce load, dynamically adjusting the parameters based on the real time information.

A new concept in SDN is developed by separating routing from the routers in order to achieve access to compute path cost thereby establishing a separate controller for routing and some virtual topology where we can have a global view of the network. This concept can be modified and tested on various applications by generating new rules, policies or strategies. This concept is developed in order to achieve the common attributes of routing technique such as scalability, load balancing, and simplicity. There is further scope for development in this area where we can develop new strategies to achieve a simple and scalable network that meets the requirements of SDN.

Also, a routing technique called explicit routing has been developed in SDN in order to reduce the events processed by the control plane and also to achieve scalability. It concentrates mainly on three issues like selecting all the transit routers in the network, implementation of the Explicit Routing technique and then evaluating its effect and results on SDN controller.

In SDN, several works have been implemented using a simple algorithm such as shortest and widest routing algorithm to optimize routing in SDN [1]. For example, the widest path is used to find a path between switch and server in the data center. While in the widest or shortest path algorithm, it was used after notifying the controller which path is more suitable for the current flow of the packet. A recent work is developing this routing technique in Floodlight Controller by changing the algorithm from shortest path to widest path algorithm. From its results, we can
conclude that this work doesn't conclude much about the features and requirements of SDN.[8][12].

The emulation of the shortest path between two nodes has been made using Bellman-Ford algorithm in Software Defined Networking (SDN) environment using mininet and POX controller [2][12].

The Box covering-based routing (BCR) algorithm has been implemented to achieve low-latency. It is used a renormalization box-covering method and is applied to test on large-scale networks by dividing the entire SDN network into some subnets using box covering method. Later the shortest path in every subnet and between subnets is found by embedding the proposed BCR algorithm code inside the SDN controller [3]. This algorithm is implemented in Mininet using Ryu controller. Results showed that the BCR algorithm gives good low-latency performance and can be applied to large-scale SDN networks [3].

There are also works done for SDN satellite onboard switching networks. The deployment of a centralized routing algorithm on SDN controller enables more rational network traffic allocation and enhances QoS supporting capability. During the packet routing process, this algorithm integrates multiple parameters including delay, bandwidth, BER, handoff, and load etc., resulting in a better-balanced network flow, improved network utilization, and reduced loss and delay jitter by the ISL handoff. Compared with currently available algorithms, this algorithm exhibits superior performance in terms of packet loss ratio, throughput, average transmission delay and delay jitter.

Currently, existing routing algorithms in SDN environments mainly concentrates on reducing routing load in multiple paths, dynamically adjusting the parameters of routing algorithms using real-time data [3] without giving much importance to network topology.

3. PRELIMINARIES

3.1. Mininet

Mininet acts as an emulator in SDN environment [4][15]. It provides an open source experimental platform to develop, test and run various network topologies using different strategies with the help of Open Flow and Software-Defined Networking systems[9][13][14]. Mininet provides various built-in features, user interfaces to create any kind of network topologies [2][11]. It’s software can emulate both the network devices and controllers in SDN environment and is freely available. The controllers (POX, Ryu, Open Daylight etc.) Used for stimulation in SDN are run by Mininet [10].

3.2. Open Daylight Controller

Open Daylight is mainly Java-based open source SDN Controller. It is used for customizing and automating networks of any size and scale [4][8]. Southbound protocols (Open Flow and others),
Northbound protocols (REST and Java RPC) are used for communication between controller and data plane, controller and applications respectively [2][3][8][12]. It includes exposure with a REST API and a web-based GUI i.e. DLUX.

3. 3. OpenvSwitch

Open vSwitch is considered as an implementation of virtual multilayer switch in computer networking. It supports any kind of protocols and is used mainly to manage the traffic [4][9][13].

3.4. Dijkstra’s Algorithm

Using Dijkstra’s, each host in the switch network or topology obtains the path information from one host to another.

4. Discussion

In recent years, Software-defined networking (SDN) is highly emerging computer networking architecture. Many strategies have been developed in SDN for efficient load balancing in SDN [1][4][6]. Also, many routing techniques have been developed such as multiple path routing algorithms, dynamic adjustments of route parameters [3].

In this paper, we use Dijkstra’s algorithm to develop an optimized routing strategy based on fat-tree topology information. To develop this strategy, first, we need to collect statistical data i.e. information about hosts connected to the switch network (MAC, IP addresses, ports etc). We use Dijkstra’s to find paths available between host1 and host2. Later we find the path cost for all the routes between hosts [2]. A fat-tree topology is thereby created by finding all the paths between the hosts in the switch network.

The fat-tree topological information of this switch network (hosts, switches, ports) is stored by Open Daylight controller in Model-driven Service Abstraction Layer (MDSAL) and YANG data modeling languages. Operational information i.e. switch related information (switch id, tables, flows in the switch) will be stored in inventory manager of MDSAL (REST API) while YANG is used to model configuration and store data manipulated by the NETCONF protocol. YANG-UI module helps you to interact with the YANG-based MDSAL data store. DLUX is used for displaying and reading of the topological information.

We calculate transmitted data rate of various paths of the same length between the hosts for an interval of time. After calculating transmission rate at subsequent intervals, the transmission rate of hosts is updated in the flow table 1 in switches. Based on the transmitted data rate at every interval, the path with low transmitted data rate is selected as best path and packet is transferred to the host for processing. The Open Daylight controllers access this information and direct the path to forward the packet.
### Table 1. Transmitted data rate from host1 to host4 in two different paths at an interval of 2ms

<table>
<thead>
<tr>
<th>Switch5 (No. of packets)</th>
<th>Switch6 (No. of packets)</th>
<th>Best path</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>7</td>
<td>Switch6</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>Switch6</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>Switch6</td>
</tr>
<tr>
<td>9</td>
<td>3</td>
<td>Switch6</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>Switch5</td>
</tr>
<tr>
<td>0</td>
<td>4</td>
<td>Switch5</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>Switch6</td>
</tr>
</tbody>
</table>

### 5. Experimental Analysis

![Fat tree topology](image)

Fig 2. Fat tree topology

The fig. 2 shown above represents the fat-tree topology of Open flow Switch network. Suppose we need to transfer a packet from host1 to host4. We has two paths through which we can transfer the packet. One path is from Switch1 via Switch5 through Switch2 to host4, whereas another path is from Switch1 via Switch6 through Switch2 to host4. In this case, we have two paths of same path cost. In such cases, the controller decides the best path to transfer the packet by comparing the transmitted data rate of both the switches i.e Switch5 and Switch6.
The transmitted data rate of both the switches, Switch5 and Switch6 is represented in graphical format in the above figure. In the first scenario, the transmitted data rate of Switch5 is 20 while Switch6 is 7. Since Switch6 has less transmitted data rate, it indicates that there is less traffic rate via the route of Switch6. Hence the controller decides the best path between the two available paths is through Switch6. Similarly, in second, third and fourth scenarios, the situation are same as that of the first scenario. But in the fifth and sixth scenario, the transmitted data rate of Switch5 is less when compared to that of Switch6. In such cases, the controller decides the best path is through Switch5.

6. Conclusion And Future Scope

In the research article, Applied methodology was Dijkstra’s algorithm to efficiently develop a routing technique to find the best path among the multiple alternative paths of same path cost between a single pair of hosts in a fat-tree topology. Implementation of Dijkstra’s algorithm in SDN environment with Open Flow Switch network using Mininet and Open Daylight controller. The performance of the dynamic load balancing concept may vary with different controllers like POX/NOX, Floodlight, Ryu, Beacon etc. The performance may also vary with different topologies of different sizes other than the fat-tree topology hence their limitations of this algorithm. Another scope of research can be of extending this algorithm to traditional and hybrid networks with both Open Flow and regular switches.

References


