Designing a Router for Software Defined Network (SDN)

Alpana Ghosh
BE Computer Engineering
MMCOE Pune
Pune- 52, India

Komal Tilekar
BE Computer Engineering
MMCOE Pune
Pune- 52, India

Ankita Singh
BE Computer Engineering
MMCOE Pune
Pune- 52, India

***Abstract*— Networks today are driven by a number of hardware devices, thus they are called as Hardware Defined Network. On the other hand are Software Defined Network or SDN, these networks unlike its counterpart are programmable, flexible, easy to maintain and cost effective. This paper proposes an algorithm for routing packets in a Software Defined Network (SDN). The algorithm is adapted to make the most of SDN technology and at the same time addresses the dynamic needs of today's networks. The proposed algorithm has its roots from multiple concepts such as principle of optimality, multistage graph and Link State Routing algorithm.**

***Keywords*— Software Defined Network (SDN), OpenFlow Protocol, router.**

I. INTRODUCTION

Modern computer networks perform a bewildering array of tasks, from routing and traffic monitoring, to access control and server load balancing. However, managing these networks is unnecessarily complicated and error-prone, due to a heterogeneous mix of devices (e.g., routers, switches, firewalls, and middle boxes) with closed and proprietary configuration interfaces.

Software defined networks are poised to change this by offering a clean and open interface between networking devices and the software that controls them.

In software-defined networks (SDNs), a logically centralized controller manages the packet processing functionality of a distributed collection of switches. SDNs make it possible for programmers to control the behaviour of the network directly, by configuring the packet forwarding rules installed on each switch. SDNs can both simplify existing applications and also serve as a platform for developing new ones.

OpenFlow is an SDN technology proposed to standardize the way that a controller communicates with network devices in SDN architecture. OpenFlow provides a specification to migrate the control logic from a switch into the controller. It also defines a protocol for the communication between the controller and the switches.

This paper proposes an algorithm for the implementation of router functionality at the controller of the SDN architecture. The controller runs the algorithm and takes routing decisions which are then communicated to the forwarding device using the OpenFlow protocol.

II. PROBLEM STATEMENT

The explosion of mobile devices, server virtualization and the advent of cloud services are among the trends driving the networking industry to re-examine the traditional network architecture. Some of the key computing trends driving the need for a new network paradigm include:

A. Changing Traffic Pattern

B. The "Consumerization of IT"

C. Rise of Cloud Services

D. Big Data means More Bandwidth

Software Defined Network (SDN) is dynamic and capable of protecting existing investments while supporting possibilities of future work. With SDN today's static network can evolve into an extensible service delivery platform capable of responding rapidly to changing business, end-user, and market needs. Understanding the importance of this technology we have come forward to work on designing a router for SDN enabled network which we feel is our future.

III. SYSTEM ARCHITECTURE

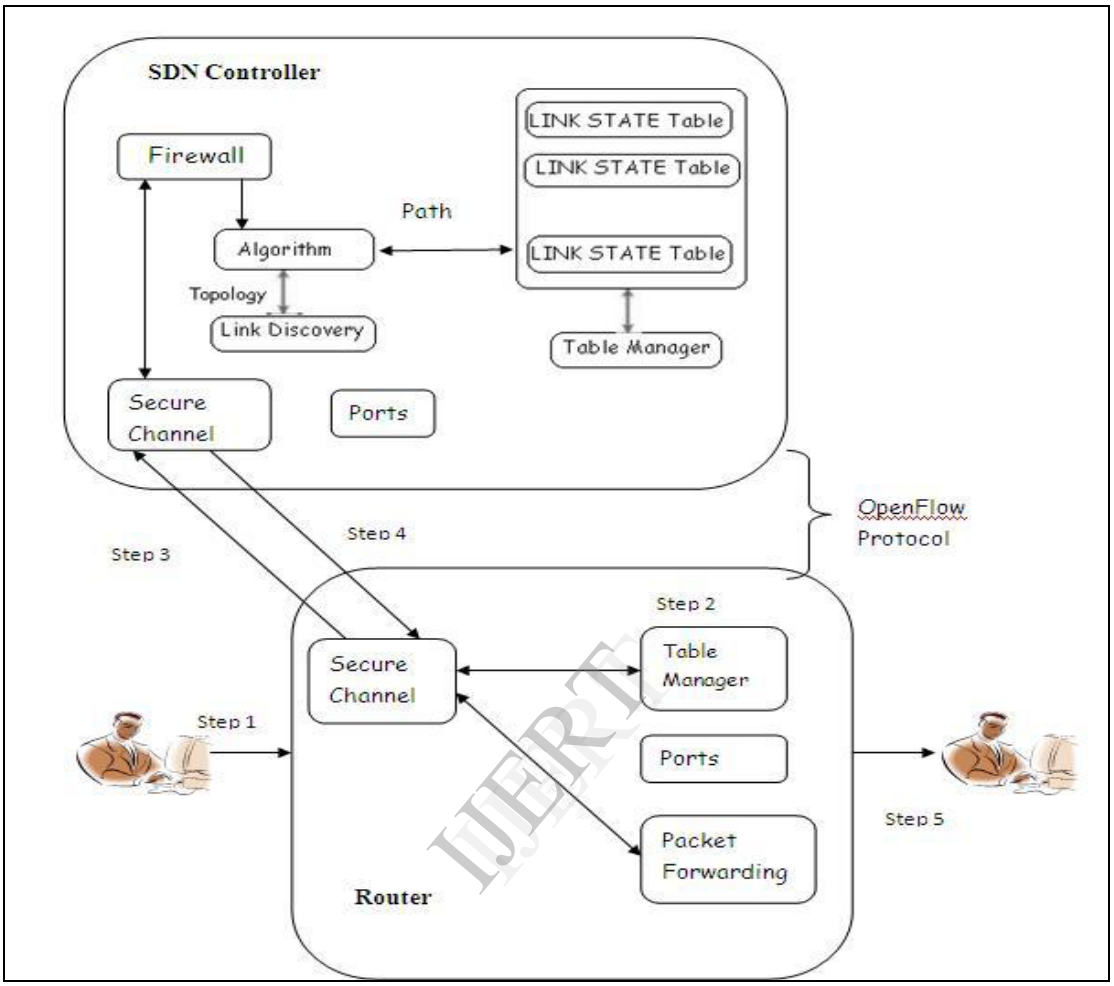

Fig. 1 System Architecture of the implemented project

Steps in Fig. 1 is explained below

1) *Step 1* – Sender wishes to communicate with the receiver, thus it sends a packet.
2) *Step 2* – The multiport device checks for a flow rule for this packet in the Flow Table. If a matching entry is found, the instructions associated with the specific flow entry are executed and goto to step 5.
3) *Step 3* – If no match is found in the Flow Table, only the packet header is forwarded to the controller via the ports.
4) *Step 4* - The controller executes the routing algorithm, and adds a new forwarding entry to the Flow Table in the switch and to each of the relevant switches along the flow path.
5) *Step 5* – The switch then forwards the packet to the appropriate port to send the packet to the receiver.

IV. PREREQUISITES FOR THE ALGORITHM

The proposed algorithm uses the following concepts

A. Optimality Principle

It states that "If router J is on the optimal path from router I to router K, then the optimal path from J to K also falls along the same route." Whenever the algorithm calculates a path from source to destination it updates the flow table of all the routers falling in this path. Thus for a single flow the path is calculated only once.

B. *Multistage Graph*

A multistage graph is a graph **G = (V, E)**

1) with **V** partitioned into **K >= 2** disjoint subsets such that if **(a,b)** is in **E**, then **a** is in $\mathbf{V_i}$, and **b** is in $\mathbf{V_{i+1}}$ for some subsets in the partition
2) and $|\mathbf{V_1}| = |\mathbf{V_K}| = \mathbf{1}$.

In order to obtain the shortest path from source to destination there are 2 approaches

1) *Forward Approach:* In this approach we start from the destination and expand through the shortest link between two stages up to the source.

2) *Backward Approach:* Similar to forward approach except this time we start from the source and expand the path up to the destination.

The proposed algorithm uses the forward approach. The advantage to doing so it that once we freeze our destination multiple nodes wanting to communicate with the same destination and among those nodes the ones that fall in the shortest path calculated previously need not recalculate their shortest path. If we were to use the backward approach for each source we would have to calculate the path thereby reducing the speed of the processor.

C. *Link State Routing*

In this routing approach each node monitors neighbours/local links and advertises them to the network. Along with that each node maintains the full graph by collecting the updates from all other nodes.

In the SDN architecture the presence of a central controller eliminates the requirement for each node to maintain the topology of the network. This job is now handed over to the controller.

V. PROPOSED ALGORITHM

1) Controller creates a HELLO packet for each connected router and directs the router to send the same to its immediate neighbours.

2) Depending on all the HELLO packets received at a router the controller creates a unique LINK STATE table for each router.

TABLE I

LINK STATE TABLE OF A ROUTER

Neighbour's IP address	**Neighbour's MAC address**	**Weight**
….. …… ……		

3) The flow tables of each router are initially empty. The flow table advices a router to forward an incoming packet to a respective output port in order to reach its destination.

TABLE II
FLOW TABLE OF A ROUTER

Destination router's IP/MAC address	**Next Router's IP address**	**Output Port Number**
….. …… …….		

4) The header of a packet received at a router is sent to the controller. The controller extracts the destination IP address. Set "len" to infinity & path length (PL) = 0.

5) The destination router's LINK STATE table is referred to create a number of potential optimal paths (POP). To avoid looping don't use routers already present in the POP. Also record the PL of these POP. (expand from destination to source)

```
For (LINK STATE tables of destination router) do
{
  if (neighbor router absent in POP)
   {
     create duplicate POP;
     annotate neighbor router to      duplicate POP;
      PL = PL + link cost of annotated router.
    }
}
```

Delete the original POP & retain the duplicate POP as original

6) If the source is reached update len = PL. Keep this POP aside Delete all the POP with PL >= len. Also delete all the POP that cannot be expanded further due to the constraint of looping.

7) If there is only one POP left & it contains the source router then terminate the algorithm. Jump to step 9.

8) Assume the destination to be the most recently added entry in the POP and jump to step 5.

9) Update the flow tables of all the routers present in the optimal path.

VI. Advantages and Disadvantages

A. *Advantages*

1) This technology has the potential to make significant improvements to service request response times, security, and reliability.
2) It could also reduce costs by automating many processes that are currently done manually and by allowing IT departments to replace (at least in some cases) high-margin devices with commodity hardware.
3) It centralizes and simplifies control of the network by making networks programmable and more agile.
4) Its architectural configuration of splitting the data plane from the control plane gives it the ability to incorporate faster, more powerful hardware.

B. *Disadvantages*

1) Many organizations simply do not have the time, expertise, or capital to invest in a completely new networking architecture, particularly smaller organizations with limited IT staff and budgets.
2) The networking device present in the software defined network must be SDN enabled. They should understand the OpenFlow protocol.
3) Since the control of the entire network is at a single point (controller), failure of the controller will result in huge time lost to recover the system.

VII. Future Scope

1) The proposed algorithm could be used in all household networks rather than larger organizations.
2) Since the control of the entire network resides in a software we can program it to run as any of the multiple network devices available in today's market. This means we can have one hardware capable of running as multiple network devices all at the same time.

VIII. Conclusion

This paper thus proposes an algorithm for routing packets in a SDN network. The algorithm provides separation of the control plane from the data plane making it less complex. The centralized controller provides a view of the network and bestows programmability of the network by external applications.

It is difficult for the world to shift from the current network configuration to a Software Defined Network but never the less this algorithm will find its usage in most of the cloud services, private networks as well as network virtualized environment.

Acknowledgment

We wish to acknowledge our external project guide in Company, Mr. Ashok Raut, the project guide in college, Prof. Geetha S. B. for their valuable suggestions and expertise.

References

[1] http://networkstatic.net/tutorial-to-build-a-floodlight-sdn-openflow-controller-module/
[2] Software Defined Networking, Jennifer Rexford, COS 461: Computer Networks Lectures: MW 10-10:50am in Architecture N101
[3] OpenFlow tutorial, Theophilus Benson
[4] http://archive.openflow.org/wp/documents
[5] http://www.openflow.org/wk/index.php/OpenFlow_Tutorial
[6] http://www.openflow.org/videos/
[7] www.csd.uoc.gr/~hy490-31/links.html
[8] CS 490.31: Software Defined Networks, Xenofontas Dimitropoulos ETH Zurich