Flow Monitoring Scheme for Reducing Congestion and Packet Loss in Software Defined Networks

Sminesh C. N.
Dept. of Computer Science and Engineering
Govt. Engineering College Thrissur
Thrissur, India

Dr. E. Grace Mary Kanaga
Dept. of Computer Sciences Technology
Karunya University
Coimbatore, India

Ranjitha K.
Dept. of Computer Science and Engineering
Govt. Engineering College Thrissur,
Thrissur, India

***Abstract*— Software Defined Networking (SDN) is a new approach in network architecture which decouples the control plane from data plane. The SDN controller has the global view of the underlying network, enabling enormous research works in the area of traffic engineering (TE). Existing congestion control solutions for SDN TE are either reactive or proactive in nature. In reactive schemes when congestion event occurs the source is informed which then reacts by rate adjustment and in proactive scheme pre computation of protection path demands additional resources. We propose a proactive flow monitoring method to reduce congestion and packet loss in SDN. This technique reduces packet loss by automatically re-routing congested link's traffic through a dynamically computed alternate path. Simulation result shows that proposed scheme reduces average packet loss leading to overall increase in network efficiency.**

***Keywords*— Traffic Engineering; SDN; Data Plane; Control Plane**

I.INTRODUCTION

With the advent of cloud services and server virtualization, Internet traffic has grown tremendously. Inefficiencies in existing architecture to support the growing traffic lead to the invent of various Internet architecture models called Future Internet [7]. SDN has become one of the most promising and popular candidates of Future Internet by its unique features like global visibility, openness, vendor independence and programmable networking devices. With an intelligent controller and a programmable data plane, SDN with OpenFlow leads to various innovative traffic engineering (TE) and load balancing techniques that are inherently flexible, adaptive, and customizable [6].

SDN TE aims to optimize the network performance by analyzing, predicting and regulating the behavior of transmitted data. Efficient traffic monitoring and routing algorithms are necessary. Routing algorithms for finding shortest paths in network finds the shortest path from the source to the destination and leads traffic through this route. However, there are many problems in the current routing system in terms of efficiency and fairness [1]. One of the inefficiencies of the existing routing architecture is that it can forward packets along non-optimal routes or it can spread load unequally, over-utilizing some links while leaving others idle. Thus it does not ensure network fairness. Since there is no open algorithm for doing network load balancing automatically, the administrator of the network should manage network parameters to disperse network load manually. With network traffic fairness issue, when network congestion occurs, packet loss occurs and the optimal route becomes the bottleneck link. This not only decreases the network throughput but also worsens average delay with high probability of packet loss, affecting the whole network efficiency.

In this paper, we propose a proactive method for reducing packet loss in SDN. This technique reduces packet loss by automatically re-routing the congested link's traffic through a dynamically computed alternate route. Controller monitors the load on each link periodically and if the link utilization exceeds a fixed threshold eventually those links can cause congestion and packet loss and are identified as bottleneck links. Once bottleneck links are identified, controller removes all these links from network virtual topology. New alternate route is computed from this updated topology and newly computed route is applied to flow tables. Further traffic uses this newly applied flow thus decreasing congestion and packet loss.

The simulation is done using Mininet [9], OpenIris controller[11] andD-ITG packet generator[10]. Performance evaluation is done to in the perspective of average packet loss for single and multiple flows. It is observed that when the network load increases the proposed proactive re-routing performs better than the existing system by reducing congestion and average packet loss leading to overall network efficiency.

In this paper, section II explains the concept of SDN and OpenFlow and discusses the related works in the area of traffic analysis in SDN. Section III discusses the proposed proactive strategy for data plane load balancing in SDN, section IV discusses the performance analysis of the proposed system and the paper is concluded in section V.

II.Background and Relate Work

SDN and OpenFlow

SDN is an emerging network architecture where the control plane logic is decoupled from the forwarding plane and is directly programmable [2]. One can control, change, and manage network behaviour dynamically through programming using open interfaces instead of relying on closed boxes and vendor defined interfaces [14]. Network intelligence is logically centralized in SDN controllers which maintain a global view of the network.

This centralized up-to-date view makes the controller suitable to perform network management functions. By having vendor independent control over the entire network from a single logical point, SDN greatly simplifies network design, network devices and its operation. Network operators and administrators can programmatically configure this simplified network abstraction rather than individually configuring each network device manually, making network management easier. It also gives network managers the flexibility to configure, manage, secure and optimize network resources dynamically.

OpenFlow is the most common standard protocol interface defined to facilitate interaction between the SDN controller and data plane [4]. OpenFlow allows direct software based access and manipulation of flow tables that instruct OpenFlow switches in the infrastructure layer to direct the network traffic. Using these flow tables, administrators can quickly change network layout and traffic flow. Since OpenFlow allows the network to be programmed on a per-flow basis, an OpenFlow based SDN architecture provides extremely granular control, enabling the network to respond to real time changes at the application, user and session levels.

The unique features of SDN, including global visibility, programmability and openness, pave way for the development of new Traffic Engineering techniques that are inherently flexible, adaptive, and customizable[1]. Following literature mainly derives solutions for SDN TE in the perspective of four thrust areas including flow management, fault tolerance, topology update and traffic characterization.

Related Work

In [6], hash-based Equal-Cost Multi Path (ECMP) load-balancing scheme to split flows across available paths using flow hashing technique is discussed. When a packet with multiple candidate paths arrive, it is forwarded to the one that corresponds to a hash of selected fields of that packet's headers modulo the number of paths, thus splitting load to each subnet across multiple paths. A key limitation of ECMP is that two or more large, long-lived flows can collide on their hash and end up on the same output port, creating a bottleneck. This static mapping of flows to paths does not account for either current network utilization or flow size, thus resulting in collisions that overwhelm switch buffers and degrade overall switch and link utilization.

In [5], Kanagevlu Renuga et al. proposed a local re-routing mechanism in SDN based data centre networks to effectively manage congestion in the event of link congestion or failure. A fault tolerant adaptive routing algorithm which balances the load across multiple paths and minimizes the occurrence of congestion is proposed. The key idea behind the algorithm is to route the packets along the lightly loaded alternate path.

In [13], Mohan et al. discuss and analyze two proactive local rerouting algorithms namely Forward Local Re-routing(FLR) and Backward Local Rerouting(BLR) to compute backup paths for a primary path for faster failure recovery. By rerouting the failed traffic from the point of failure, local re-routing enables fast recovery. FLR and BLR algorithms choose backup paths so as to reduce the number of forwarding table entries with improved sharing of forwarding rules at the switches along the primary and backup paths.

Sang Min et al. proposed a new routing architecture called Automatic Re-routing with Loss Detection (ARLD) [3] which is enabled by SDN and OpenFlow protocol. Basic idea of ARLD is that in wired network, the most common cause for packet drop is network congestion, so a packet drop occurs at a certain link, controller treats that link as a bottleneck link. Once packet drop is detected, SDN controller's re-routing module is invoked which further initiates by-pass or alternate route computation. Re-routing module finds out alternate route, if any, and then controller update switch flow tables with this alternate route. Further traffic gets routed through this newly added route until the flow table entry expires.

Another issue with ARLD method is that alternate route computation does not consider the existing network utilization. All the available paths in the virtual topology is taken as input for by-pass route computation. If the newly computed alternate path happens to be an already fully utilized link, this re-routing can again result in network congestion at that link, leading to congestion propagation starting from one node to the whole network degrading the whole network efficiency. Also in ARLD approach discussed above, once packet loss in detected, the node itself is removed from the topology. This makes all other paths through this node unavailable during alternate route computation.

A reactive method initiates re-routing and alternate route computation once network congestion and packet drop happens. By this time, as the network is already congested, good number of packets already gets dropped by the time controller computes and applies alternate route to flow tables. In the proactive approach involves computation of primary path and back-up path for faster failure recovery results in additional resource reservations and flow table entries which is an overhead to the SDN controller.

III.SYSTEM ARCHITECTURE

The proposed system consists of mainly two modules. The modules are integrated inside the SDN Controller.

- Port Monitoring Module
- Re-routing Module.

Port Monitoring Module

Port monitoring module collects the port statistics through periodic monitoring of each port on each switch. The OpenFlow protocol makes use of request and reply messages for collecting port statistics. Controller then computes port utilization for each port using transmitted bytes count from port statistics message using the formula[12].

$$\text{PortUtilization} = \frac{\text{BytesTransmitted} * 8}{\text{PortSpeed} * \text{TimeInterval}} * 100 \qquad (1)$$

If utilization exceeds a fixed threshold for any of the port, links associated with those ports are identified as bottleneck links. Figure 1 shows the computation of port utilization and identification of bottleneck link.

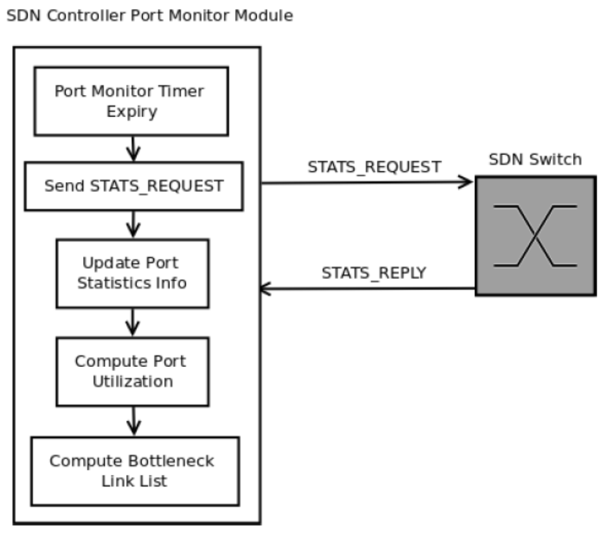
SDN Controller Port Monitor Module
Port Monitor Timer Expiry
Send STATS_REQUEST
Update Port Statistics Info
Compute Port Utilization
Compute Bottleneck Link List
STATS_REQUEST
STATS_REPLY
SDN Switch

Figure 1: Computation of Bottleneck Link

Re-routing Module

Re-routing module is responsible for updating network virtual topology and alternate route computation. Figure 2 shows the module architecture.

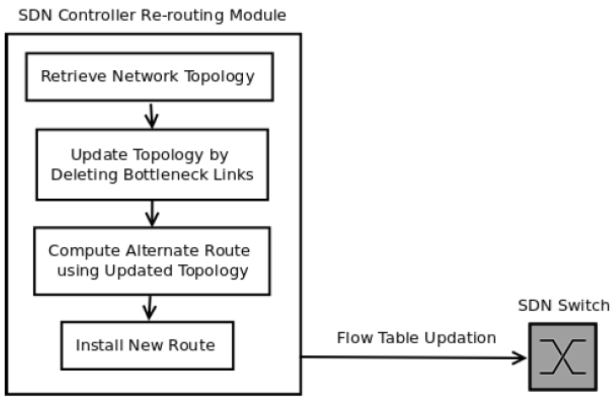
SDN Controller Re-routing Module
Retrieve Network Topology
Update Topology by Deleting Bottleneck Links
Compute Alternate Route using Updated Topology
Install New Route
Flow Table Updation
SDN Switch

Figure 2 : Re-routing Module

Re-routing module retrieves the network topology from topologyManager module in the controller. Retrieved topology is maintained locally as an adjacency matrix of size NxN (AdjMatrix[N][N]), where N is the number of switches under the controller. Now for each link SiSj present in the topology, AdjMatrix[i][j] and AdjMatrix[j][i] is set to one. Absence of a link between two nodes is marked by setting corresponding AdjMatrix value to zero. Bottleneck links are removed from the topology by updating the adjacency matrix. For any bottleneck link SiSj, it is removed from the topology by setting AdjMatrix[i][j] and AdjMatrix[j][i] to zero.

Once the local topology is updated, Dijkstra's algorithm is used to compute alternate path for congested flows. The updated adjacency matrix is given as input to Dijkstra's algorithm which returns alternate path.

IV.SIMULATION AND RESULTS

Mininet[9] is used to simulate the experiment topology given in Figure 3 with 11 hosts and switches. The SDN controller is implemented using java based OpenIris SDN controller[11] developed by ETRI. Traffic generation for performance evaluation is done using the log files generated by the D-ITG packet generator[10].

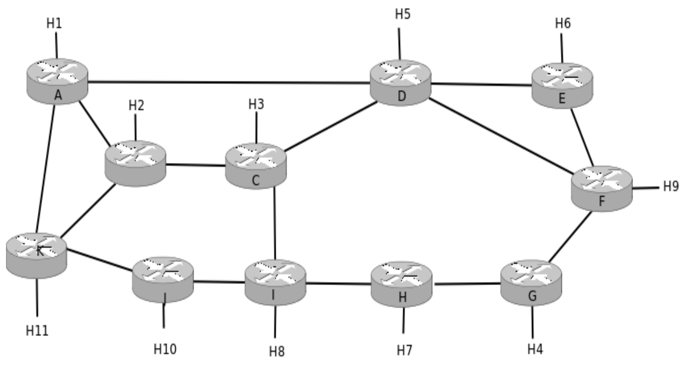

Figure 3 : Experiment Topology

Proposed system evaluation is done in two scenarios

- Single flow with varying packet rate
- Multiple flows with constant packet rate.

For the first scenario, link AB is configured at 10Mbps bandwidth and all other links at 20Mbps bandwidth. Using D-ITG packet generator, 1000 bytes UDP packets are sent from host H1 to host H2 at varying packet rate like 1000 packets per second, 2000 packets per second etc.

For the second scenario, each link in the network is assigned a bandwidth of 20Mbps. 1000 bytes UDP packets are then sent from host A to every other host(H2-H11) at constant packet rate of 1000 packets per second for 60 seconds. This resulted to having a maximum of 10 flows in the network.

Performance Analysis

The performance of the proposed system is analyzed in terms of loss percentage[12], is measured for the same topology by varying parameters like packet rate and number of flows.

Single flow with varying Packet Rate

The variation of packet loss with varying packet rate is as shown in Figure 4. We can observe that the proposed system reported much lesser packet loss than existing system. By re-routing the packets via alternate path, proposed system considerably reduced the number of packets dropped.

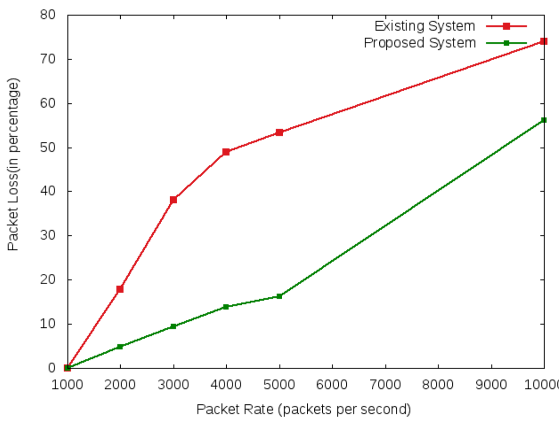

Figure 4: Average Packet Loss with varying packet rate

Multiple flows with constant packet rate

The variation of average packet loss with varying number of flows is as shown in Figure 5.

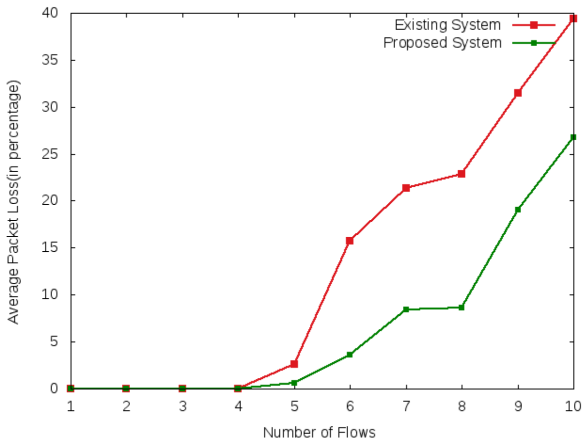

Figure 5: Average Packet Loss with varying Number of Flows

We can observe that the proposed system reported much lesser packet loss than the existing system when number of flows is varied from 1to10. Initially loss remained the same for both the system as the network was not congested. But as the number of flows increased above 5, proposed system reported much lesser packet loss than existing system. This is achieved by re-routing packets from congested path via non-congested alternate paths. We can also observe that proposed system always reported less loss than existing system which shows that there was no congestion propagation in the proposed system.

V.CONCLUSION AND FUTURE WORKS

In this paper we proposed a proactive traffic analysis method for reducing network congestion in SDN through periodic monitoring and automatic re-routing of the network traffic from congested path to a non-congested alternate path. Simulation results shows that proposed pro-active traffic analysis method to reduce congestion and packet loss outperformed the reactive method in the perspective of packet loss.

As a future work we are planning to provide a solution for congestion propagation through efficient call admission. Performance comparison can be made between proposed proactive scheme and existing reactive schemes. We also plan to extend the performance analysis to other network performance metrics like average throughout and average end-to-end delay.

REFERENCES

[1]. Ian F. Akyildiz, Ahyoung Lee, Pu Wang, Min Luo, and Wu Chou, May/June 2016 "Research Challenges for Traffic Engineering in Software Defined Networks", published in IEEE Network ,.

[2]. Kreutz, Ramos, Esteves Verissimo P, Esteve Rothenberg C, Azodolmolky S, and Uhlig S 2015, "Software-Defined Networking: A Comprehensive Survey", Proceedings of the IEEE 103.1: 14-76.

[3]. Sang Min Park, Seungbum Ju, Jaiyong Lee 2014, "Efficient routing for traffic offloading in Software-

defined Network", Procedia ComputerScience 34 : 674-679.

[4]. https://www.opennetworking.org/images/stories/downloads/sdnresources/onf-specifications/openflow/openflow-spec-v1.4.0.pdf "Openflow switch specification V1.4.0 (Wire Protocol 0x05),October 14, 2013"

[5]. Kanagavelu Renuga, and Khin Mi Mi Aung 2015, "SDN Controlled Local Re-routing to Reduce Congestion in Cloud Data Center", International Conference on Cloud Computing Research and Innovation (ICCCRI) IEEE.

[6]. Akyildiz, I., Lee, A., Wang, P., Luo, M. and Chou, W. (2014). A roadmap for traffic engineering in SDN-OpenFlow networks. Computer Networks, 71, pp.1-30.

[7]. Braun, W. and Menth, M. (2014). Software-Defined Networking Using OpenFlow: Protocols, Applications and Architectural Design Choices. Future Internet, 6(2), pp.302-336.

[8]. https://www.opennetworking.org/images/stories/downloads/sdnresources/white-papers/wp-sdn-newnorm.pdf,"Software-Defined-Networking: The New Norm for Networks ONF White Paper" April 13,2012

[9]. Lantz B, Heller B, & McKeown N (2010 October) A network in a laptop : rapid prototyping for Software Defined networks In Proceedings of the 9th ACM SIGCOMM WORKSHOP ON Hot Topics in Networks

[10]. Alessio Botta, Walter de Donato, Alberto Dainotti, Stefano Avallone, and Antonio Pescape , "D-ITG 2.8.1 Manual" University of Napoli Federico II http: // traffic. comics. unina. it/ software/ ITG, October 28, 2013

[11]. Byungjoon Lee, Sae Hyong Park, Jisoo Shin, and Sunhee Yang, "IRIS: The Openflow-based Recursive SDN Controller" ICACT 2014 :1237-1240

[12]. Larry L. Peterson & Bruce S. Davie Computer Networks a Systems Approach 4th Edition, 2007, Elsevier

[13]. Mohan, Purnima Murali, Tram Truong-Huu, and Mohan Gurusamy 2015, "TCAM-aware Local Rerouting for Fast and Efficient Failure Recovery in Software Defined Networks", IEEE Global Communications Conference (GLOBECOM). IEEE.

[14]. Pfaff B, Pettit J, Koponen T, Jackson E, Zhou A, Rajahalme J, Gross J, Wang A, Stringer J, Shelar P, Amidon K 2015, The design and implementation of open switch, 12th USENIX symposium on networked systems design and implementation (NSDI 15).