Congestion Control Proposal in SDN With Random Early Detection

Luis Lezcano Airaldi
CINAPTIC - FRRe
Universidad Tecnológica Nacional
Corrientes, Argentina
luislezcano@frre.utn.edu.ar

Reinaldo J. R. Scappini
CINAPTIC - FRRe
Universidad Tecnológica Nacional
Corrientes, Argentina
rscappini@gfe.frre.utn.edu.ar

Sergio Gramajo
CINAPTIC - FRRe
Universidad Tecnológica Nacional
Resistencia, Argentina
sergio@frre.utn.edu.ar

Diego Bolatti
CINAPTIC - FRRe
Universidad Tecnológica Nacional
Resistencia, Argentina
dbolatti@frre.utn.edu.ar

***Abstract*— The emerging technology of SDN (Software Defined Networks) separates the data control plane from the forwarding plane while maintaining a centralized control of the network management. The SDN features require new traffic engineering techniques that exploit the global (centralized) network view, and the status and features of traffic flows. Our purpose is to perform traffic engineering in an SDN architecture, using the OpenFlow protocol capabilities and the potential of the SDN controller to collect operational data of the entire network, such as topology, latency, buffer utilization and frame sizes of the controlled devices in order to implement QoS. Considering that the performance of the network is an essential component of Quality of Service (QoS), and congestion is the main factor that affects it, this paper explores a method to search for alternative paths based on data from switches using the Random Early Detection (RED) congestion control mechanism.**

***Keywords*— SDN, Congestion Control, QoS, RED**

I. INTRODUCTION

Nowadays, a vast range of applications and services are provided to satisfy the growing and dynamic needs of society and Internet traffic has turned very dynamic and complex. With the continued and increasing demand, congestion prevention and control is a very important challenge in order to avoid deficient performance in collapsed networks.

Although traffic-engineering techniques have been widely exploited in the past to optimize the performance of communication networks, paradigms established by new generation networks carry out dynamically analyzing, predicting and regulating the behavior of data transmission due to several important reasons, such as real-time applications [1] [2].

Traditional networks have congestion control algorithms implemented across the whole protocol stack in packet switching networks [3]. This is not only to get efficient bandwidth usage and network stability but also to reach a satisfactory performance using the available resources such as RED algorithm [4] [5].

Multiple methods were applied to solve several network problems, especially QoS [6] [7]. In [8] the authors propose a theoretical model to search for alternative paths in a congested network.

In this way, network devices and data center infrastructure must respond to the growing demand with adequate scalability and SDN can be useful to improve network performance and congestion control [9].

SDN is revolutionary network paradigm that separates out network operation of the control plane above the infrastructure [10] [11], centralizing data routing decisions by means of a central controller that communicates with switching devices through the OpenFlow protocol [12].

The centralization of the SDN control plane and the possibility of developing applications has effect on network, thus, forwarding plane allows to add smart solutions that can be applied to improve routing algorithms such as congestion control. Therefore, many solutions have been proposed in literature that addresses this problem, such as studies over throughput and performance of data centers based on TCP proposals and SDN control avoiding severe collapses [13] [14] [15]. Also, some methods have been proposed to avoid congestion applied in Internet of Things (IoT) Applications and Data Centers using SDN [16]. In [17] authors propose a framework of congestion control decision based on global real-time network conditions information and occupied rate exchange information between nodes and controller [18] [19]. In [15] and [20] congestion control is calculated with link utilization in SDN controller and, if it necessary, rerouting algorithm is applied to choose a better route [21].

Following this approaches, each time t, the current state of the network can be analyzed to predict future behavior and update configuration parameters, such as routing tables or certain thresholds, to avoid network problems. To do that, this work proposes an approach using RED algorithm and SDN architecture. The scenario described makes certain assumptions to simplify the model: the state space is considered finite (a core network) with no uncertainty, since the paths between the nodes are known beforehand and the function to go from one state to another is simply a network link managed by the controller.

The rest of the paper is organized as follows. Section 2 reviews the main concepts SDN, QoS an RED and its standards; Section 3 presents our proposal integrating SDN and search algorithm to solve congestion problems over SDN forwarding plane. Section 4 contains an illustrative example. Finally, Section 5 presents the concluding remarks.

II. Fundamentals of SDN, QoS and RED

Before reaching our proposal, it is necessary to revise useful previous concepts of SDN, QoS with QoE and RED.

A. *Software Defined Networking*

SDN has emerged as an efficient network technology capable of supporting the growing of future networks and intelligent applications. In addition, it allows reducing the operating costs through simplified hardware, software, and management methodologies [22].

The SDN Controller is a logically centralized entity in charge of translating the requirements from the SDN Application layer to SDN Datapaths and providing SDN applications with an abstract view of the network, which may include statistics and events [11]. The SDN Controller has complete control of SDN Datapaths, subject only to the limit of their capabilities, and thus it does not have to compete with other elements in the control plane. This simplifies scheduling and resource allocation and allow networks to run with more precise policies, allowing for a greater resource utilization and guaranteed quality of service. This is done through a well-understood common information model (e.g. as the one defined by the OpenFlow Protocol [12] [23]).

Fig. 1 shows a general overview of the SDN architecture and its layers [24]. The Openflow protocol operates between the data plane (lower layer) and the control plane (upper layer). The controller makes its decisions based on calls from applications, which use the upper layer API that operates above the control plane.

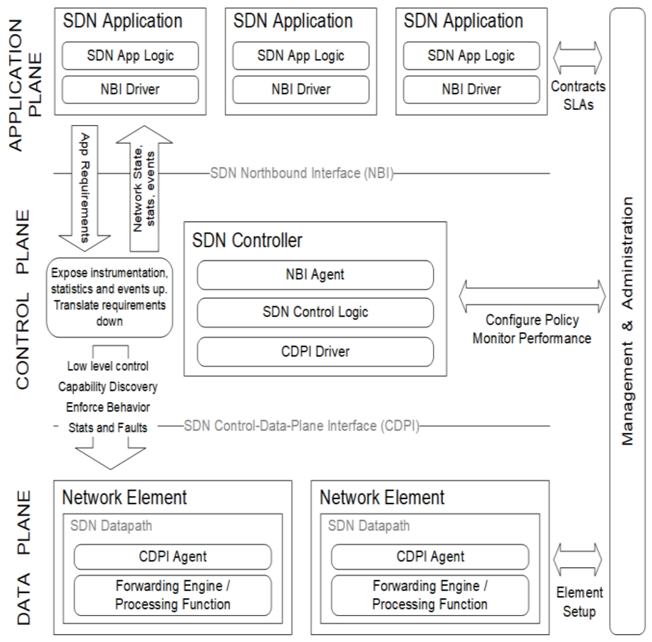

Fig. 1. Overview of Software-Defined Networking Architecture

B. *Quality of Service (QoS) and Quality of Experience (QoE)*

ITU-T Recommendations E.800 [25] and ITU-T E.804 [26] provide the basic definition of QoS and QoS services in networks. These recommendations cover the whole end-to-end aspects of telecommunication services (Fig. 2).

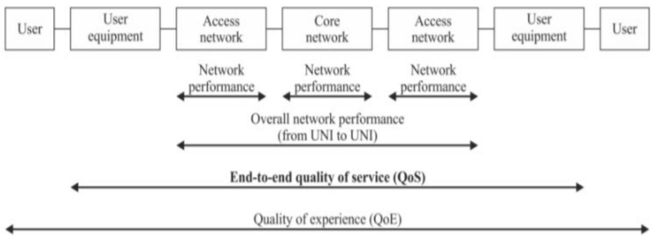

Fig. 2. End-to-end QoS (E.804)

Quality measures in telecommunications can be described in a hierarchical manner:

Network performance (NP): The network performance is assessed across a part of a network or a sub-network. Example parameters are bit error ratio, sending and receiving power, transmission delay.

Overall NP: If several network sections should be considered as being one integral part of the network (i.e. a black box), the overall network performance must be assessed. For example, the network performance of the whole transmission between two User Network Interfaces (UNIs) can be summarized in this way.

End-to-end quality of service (QoS): The assessment of the overall transmission chain from a user's perspective is considered to deliver the QoS in an objective manner. This implies that the most complete transmission chain without involving the user should be considered. Mostly, the measures rely on service-related characteristics without knowing any details about the underlying network sections, which are required to have an end-to-end service.

Quality of experience (QoE): The inclusion of the user to the overall quality in telecommunications extends the rather objective QoS to the highly subjective quality of experience (QoE). The QoE differs from user to user since it is influenced by individual experiences and expectations.

C. *Random Early Detection algorithm*

This algorithm, proposed by Floyd and Van Jacobson in the early 1990s, is a mechanism to address or avoid network congestion in a proactive (instead of reactive) way [27].

The buffer size and the delay of local queues at each node in the switches are the main contributors to the delay in the overall network that affect the transmission of packets and may result in a congestion [27]. A soft limit on the internal queue delay of the switch can be calculated using the maximum buffer occupancy of each switch together with the outbound link ratio. If queues have been assigned to the switch ports the OpenFlow controller can send a message "queue get config" to retrieve the relevant values of the queue, including length (in bytes), and minimum and maximum data rate (with Open Flow 1.2 and above). The local queuing delay can be obtained by dividing the maximum size of the buffer by the corresponding output link rate.

The Algorithm 1 Fig. 1 shows the basic outline of the RED algorithm in its simplest form, although there exist many variants [28] that were developed for specific purposes such as Weighted Random Early Detection, RED with the Unresponsive Flow Identification, Flow Random Early Detection, Balanced Random Early Detection.

The model proposed in this paper may be used with any implementation of RED assuming the controller has access to the necessary data about the queue's states.

```
1.  function RandomEarlyDetection($min_{th}$: minimum threshold,
    $max_{th}$: maximum threshold)
2.    for each packet arrival
3.      calculate new average queue size $avg$
4.      if $min_{th} \leq avg \leq max_{th}$
5.        calculate packet dropping probability $p_a$
6.        with probability $p_a$ drop the arriving packet
              or enqueue it with probability $(1 - p_a)$
7.      else if $avg \geq max_{th}$
8.        drop the arriving packet
9.      else
10.       enqueue packet
```

Fig. 3. Algorithm 1 – Random Early Detection

The basic behavior of RED, as illustrated in Algorithm 1, can be summarized as follows: every time a packet arrives to a queue in a switch, it has a probability p_a of being dropped. This probability starts at a minimum when the queue is empty, and it slowly increases proportionally to the queue length. This means that, as the congestion increases, so does the amount of dropped packets. If the queue size is above a certain predefined threshold max_{th}, the arriving packets are always dropped. In the same way, if the queue size is below a minimum threshold min_{th}, the arriving packet is always queued.

III. Proposal

This section describes the proposed model. It is divided into two subsections. The first mentions the tools and software used to carry out the simulation, and then, the second section describes the algorithm implementing the proposed solution.

A. Scenarios Simulation Tools

Multiple scenarios were tested with different topologies and operation methodologies. A working framework was set up to measure the performance of the network between any two nodes and from end-to-end nodes in order to obtain the NP (Network Performance) and QoS (Quality of Service) parameters.

The simulators used were Mininet [29] and GNS3 [30] with OpenVSwitch [31] appliances using, in both cases, an OpenDayLigh controller (with different versions) and OpenFlow protocol version 1.3. This software was chosen due to its availability and extended use in academic research.

The state of the queues in the SDN switches can be obtained using low-level API calls, one of which is described here due to its importance for the method proposed. The queue-stats switch command returns statistics for all queues on all ports in the switch, or only for the specified port and queue.

B. Proposed algorithm

```
1.  every time $t$:
2.    read queues state from all nodes
3.    obtain the $AQL$ for every node
4.    for each node $i$:
5.      if $AQL_i > AQL_{thres}$:
6.        find a new non-congested route for traffic that
            passes through node $i$
7.        if there is a new route:
8.          update the routing tables in the controller
9.          add node $i$ to $congestedNodes$ list
10.     if $i$ is in $congestedNodes$ and $AQL_i < AQL_{thres}$
11.       restore node $i$ to the previous route
```

Fig. 4. Algorithm 2 – Proposed model

The controller performs readings on all nodes in the network every certain time t to obtain operational data from the queues in the switches. The parameter of interest here is the average queue length (AQL), which is an indicator of possible congestion when using RED gateways. These data are then used to determine if a node is congested by analyzing if the AQL is above a certain threshold. If a congestion state is detected on a node, the controller will switch traffic through a different route, allowing the congested node to go back to a normal operating state after a period of time t.

To find a new route, the algorithm shown in Algorithm 2, checks each node along the path to see if it is operating below the predetermined congestion threshold (AQL_{thres}). The purpose of this is to find a new path which is not congested (lines 2 to 6).

If a route is found, the controller updates its routing tables and marks the congested nodes, so all new traffic that passes through them will switch to the new route (lines 7 to 9). In the next iterations, it periodically checks if the congested nodes can be restored to their previous path, allowing the network to recover from a congestion state and go back to the initial setup (lines 10 and 11).

IV. Illustrative Example

To illustrate the idea proposed in this paper, let us consider a simple network topology as it shown in Fig. 3. This method, however, can easily scale to more complex topologies. This network has seven nodes, which can be standard routers or a set of 5G nodes, managed by an SDN controller.

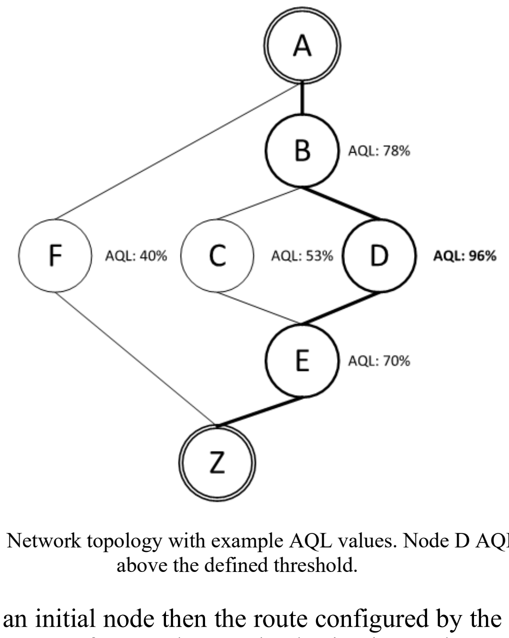

Fig. 5. Network topology with example AQL values. Node D AQL is above the defined threshold.

A is an initial node then the route configured by the SDN controller to get from node A to the destination node Z consist of the following path: A, B, D, E, Z. The controller reads the state of the nodes every certain predefined time interval t as can be seen in step 1 of Algorithm 2. After a period of time the node D reports a situation of "probable packet loss", that is, the congestion threshold or average queue length AQL is getting closer to the predefined value AQL_{thres}. This can cause the entire network to experience a performance degradation and, therefore, a decrease in the QoS for users. The AQL values are checked by the Algorithm in step 2.

In step 3, the model tries to find a new route for traffic that passes through the congested node D. If it finds this route, the controller proceeds to update its internal routing tables with the new route. In our example, all new traffic for the destination node Z will go through the route A, F, Z, and the node D will be marked as "switched" adding it to the list of congested nodes.

Being left only with the existing flows, the congested node D is expected to lower its AQL value over the next period, going back to a normal operating state after some time. Consequently, in step 4, the model checks the new AQL values of the nodes in the congested list. If it detects that a node's AQL dropped below a certain threshold, it is assumed that it is safe to restore that node to its previous route. Fig. 4 illustrates this behavior, showing the new data flows being routed through node F, and the consequent drop on node D AQL.

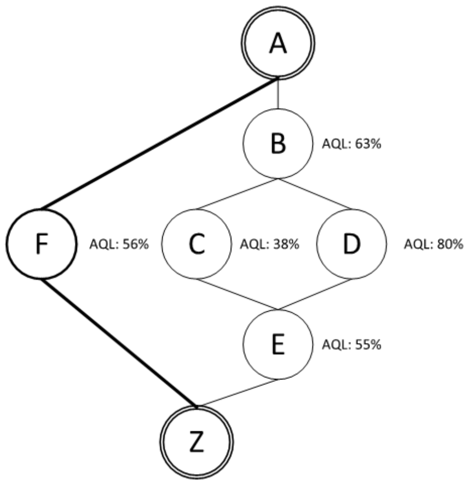

Fig. 6. Network topology showing the new route from A to Z and new AQL values due to the path change.

TABLE I. SHOWS A SUMMARY OF THE EXAMPLE DESCRIBED ABOVE, FOLLOWING THE STEPS IN THE ALGORITHM PROPOSED

SUMMARY OF THE EXAMPLE	
Node	***State***
S_0	Route from A to Z : $A - B - D - E - Z$.
S_1	AQLs: B 78%, D 96%, E 70%.
S_2	Congestion detected in node D ($AQL_d > AQL_{thres}$).
S_3	New route found to Z: $A - F - Z$. Add D to the congested nodes list.
S_4	Restore nodes in the congested list to their previous path if their AQL value dropped below the predefined threshold.

The tests results show that the controller switches routes when the congestion condition is met, but still further research is required with different conditions and evaluation of corner cases to better assess the proposed method. In addition, other considerations are necessary for real-world application, like flow identification in exit nodes and predefined priorities in queueing decisions.

V. CONCLUSIONS

The purpose of this paper is to explore the idea of controlling congestion reacting to certain network conditions. Taking advantage of the capabilities of a centralized network controller allows for a wide range of possibilities but they still require further research.

The goal of the proposed model is to detect congestion in an SDN network by reading data from switches running any variant of the RED algorithm, and use these data to act accordingly, modifying the routes to avoid the congested paths. The model finds a new path to the destination node verifying that it does not suffer from congestion.

The model could be extended by adding smart capabilities. These may range from modeling the network to consider more parameters than just the AQL to building a neural network that learns the traffic characteristics and adapt or change different parameters and even predict future congestion conditions considering data like peak hours, queue usage and congestion history.

REFERENCES

[1] H. Kopetz, Internet of things. In Real-time systems, ed: Springer, 2011, pp. 307-323.

[2] J. W. Guck and W. Kellerer: "Achieving end-to-end real-time Quality of Service with Software Defined Networking" IEEE 3rd International Conference on Cloud Networking (CloudNet), Luxembourg, 2014.

[3] D. E. Comer, Internetworking with TCP/IP, ed: Prentice Hall, 2005.

[4] S. Wu, X. Liu, & Z. Wang. "Enhanced random early detection algorithm BACnet router for congestion avoidance", 7th International Conference on Advanced Communication Technology IEEE, ICACT 2005, vol. 2, pp. 1156-1159, 2005.

[5] L. Enhai, L. Yan & Ruimin, P. "An improved random early detection algorithm based on flow prediction", Second International Conference on Intelligent Networks and Intelligent Systems IEEE, pp. 425-428, 2009.

[6] T. Szigeti & C. Hattingh, End-to-End QoS Network Design: Quality of Service in LANs, WANs, and VPNs. Cisco Press, 2004

[7] A. Barakabitze, et al., "QoE Management of Multimedia Streaming Services in Future Networks: A Tutorial and Survey," IEEE Communications Surveys & Tutorials, vol. 22, no. 1, pp. 526-565, Firstquarter 2020.

[8] Z. Cheng, X. Zhang, Y. Li, et. al., "Congestion-aware local reroute for fast failure recovery in software-defined networks", IEEE/OSA Journal of Optical Communications and Networking, vol. 9, no 11, pp. 934-944, 2017.

[9] D. Thomas, Nadeau, K. Gray, SDN: Software Defined Networks: An Authoritative Review of Network Programmability Technologies. ed: O'Reilly Media; 1st edition, 2013.

[10] S. Azodolmolky, Software Defined Networking with Openflow, ed: Packt Publishing, 2013.

[11] R. Scappini, R. Calcagno, F. Alarcon, S. Gramajo and D. Bolatti, Trabajando con SDN Y OPENFLOW. 1ra Edición. E-book, ed: Sergio Gramajo, 2019.

[12] Open Networking Foundation. OpenFlow Switch Specification. https://www.opennetworking.org/software-defined-standards/specifications/ , last access date: August, 2020.

[13] T. Hafeez, N. Ahmed, B. Ahmed & A. Malik. “Detection and mitigation of congestion in SDN enabled data center networks: A survey”. IEEE Access, vol. 6, pp. 1730-1740, 2017.

[14] Y. Lu, & S. Zhu. “SDN-based TCP congestion control in data center networks”. IEEE 34th International Performance Computing and Communications Conference (IPCCC), pp. 1-7, 2015.

[15] J. Bao, J. Wang, Q. Qi & J. Liao, “ECTCP: An Explicit Centralized Congestion Avoidance for TCP in SDN-based Data Center”. IEEE Symposium on Computers and Communications (ISCC), pp. 00347-00353, 2018).

[16] Y. Lu, Z. Ling, S. Zhu, & L. Tang, “SDTCP: Towards datacenter TCP congestion control with SDN for IoT applications”, Sensors, vol. 17, no. 1, pp. 109, 2017.

[17] S. Hertiana, A. Kurniawan & U. Pasaribu, “Path associativity centralized explicit congestion control (PACEC) for SDN”, International Conference on Control, Electronics, Renewable Energy and Communications (ICCREC), pp. 18-23, 2017.

[18] J. Gruen, M. Karl & T. Herfet, “Network supported congestion avoidance in software-defined networks”, 19th IEEE International Conference on Networks (ICON), pp. 1-6, 2013.

[19] M. Kao, B. Huang, S. Kao & H. Tseng, “An effective routing mechanism for link congestion avoidance in software-defined networking”, IEEE International Computer Symposium (ICS), pp. 154-158, 2016.

[20] S. Song, J. Lee, K. Son, H. Jung, & J. Lee. “A congestion avoidance algorithm in SDN environment”. IEEE International Conference on Information Networking (ICOIN), pp. 420-423, 2016.

[21] Z. Abdullah, I. Ahmad & I. Hussain, “Segment routing in software defined networks: A survey”, IEEE Communications Surveys & Tutorials, vol. 21, no. 1, pp. 464-486, 2018.

[22] S. Sezer et al., "Are we ready for SDN? Implementation challenges for software-defined networks", IEEE Communications Magazine, vol. 51, no. 7, 2013.

[23] Open Networking Foundation, SDN Technical Specifications: https://www.opennetworking.org/software-defined-standards/models-apis/, last access date: June 2020.

[24] Open Networking Foundation, SDN Architecture, TR-502: https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/TR_SDN_ARCH_1.0_06062014.pdf, last access date: June 2018.

[25] Recommendation ITU-T E.800, Definitions of terms related to QoS: https://www.itu.int/rec/T-REC-E.800-200809-I, last access date: June 2018.

[26] Recommendation ITU-T E.804, QoS aspects for popular services in mobile networks: https://itu.int/ITU-T/E.804, last access date: June 2018.

[27] Sally Floyd and Van Jacobson “Random Early Detection Gateways for Congestion Avoidance” IEEE/ACM. Transactions on Networking, vol. 1, no. 4, pp. 397-413,1993.

[28] P. Singh, S. Gupta, Variable Length Virtual Output Queue Based Fuzzy Adaptive RED for Congestion Control at Routers. In: Aluru S. et al. (eds) Contemporary Computing. IC3, pp. 123-134, 2011.

[29] Mininet: An Instant Virtual Network on your laptop: http://mininet.org, last access date: June 2020.

[30] GNS3: https://www.gns3.com, last access date: June 2020.

[31] Open vSwitch, Production Quality and Multilayer Open Virtual Switch: https://www.openvswitch.org, last access date: June 2020.