Optimization of proactive routing with latency minimization in SDN networks

Armel KEUPONDJO[1,2,3], Nogbou Georges ANOH[3], Ferdinand ATTA[1,2,3], Souleymane OUMTANAGA[1,2,3]

1- *Ecole Doctorale Polytechnique de l'INP-HB UMRI 78 : EEA, Laboratoire de Recherche en Informatique et Télécommunications (LARIT). Email: armelkeupondjo@gmail.com*
2- *Institut National Polytechnique Félix Houphouët Boigny de Yamoussoukro*
3- *Laboratoire de Recherche en Informatique et Télécommunications (LARIT)*

Abstract: SDNs networks opt for a centralized approach, in which the controller is the element that defines the overall network management policy and communicates it to openflow switches via the OpenFlow protocol whenever it is requested by them or he considers it opportune. This has a strong impact on latency and packet loss, parameters that have very important for multimedia applications. To optimize the transmission delay in data networks with SDN, two approaches are generally proposed; a responsive approach and a proactive approach. However, we notice that the reactive approach, even if it allows to better evaluate the quality of the optimal paths, increases the latency time. While the proactive approach strongly reduces this time, but does not consider parameters such as the failure of a node on the transfer path. We will propose in this document a mixed approach based on the back-pressure algorithm which is proven as an optimal routing algorithm and rate planning. The aim is to optimize the routing functions by simply placing the traffic where the capacity allows, to avoid congestions of some parts of the network strongly solicited.

Key words: latency, proactive routing, SDN, hybrid algorithm

1. INTRODUCTION

In classical networks, routers determine on which output interface to direct packets for a given destination, using distributed routing protocols. In SDNs, one or more controllers support the calculation of routes and routers are reduced to simple transmission devices. Figure 1 shows the routing of a packet between a source and a destination in an SDN with the Openflow protocol.

The controller sends a forwarding rule to the router A (1) so that when a packet arrives at that router (2), with a destination IP address that is included in the network IP address of the forwarding rule, the router will immediately know which interface to transfer the packet (proactive method). Router B, having no management rule concerning the package reached at its level, will contact the controller (4) to know the attitude to adopt with respect to the package. The controller sends down a transfer rule (5) concerning the packet and the router is then able to transfer the packet to the next node of the network (6) (reactive method).

TREE 2018, March 21-22, Abidjan, Côte d'Ivoire

DOI 10.4108/eai.21-3-2018.2277821

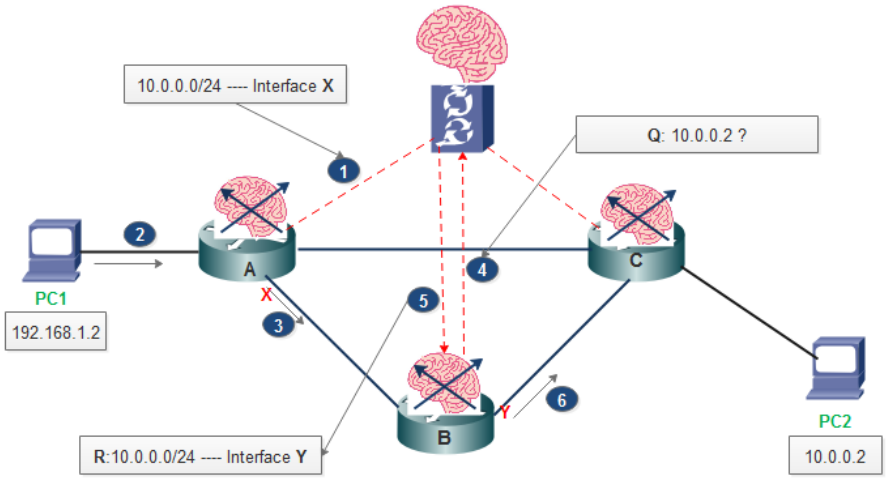

Fig. 1: Diagram of a simple SDN network

The collaboration between network equipment's is therefore based on the exchange of messages. These messages can be upstream warnings asking by example a reduction of the flow, as introduced in [16] [17]. We can also imagine that it is messages authorizing the admission of new flows [18], or broadcasting information related to the state of the router.

However, the main goal of traffic engineering is to avoid congestion of some heavily used parts of the network by controlling and optimizing routing functions (placing traffic where capacity permits). The challenge here is to adapt well to the dynamic character of the topology (case of breakdowns) and the demand. Because the current bottleneck in communications networks like, data networks and cloud computing are due to traffic increasingly important, which can affect the quality of service.

To optimize the transmission delay in data networks with SDN, two approaches are generally proposed.

2. RELATED WORK

2.1. Proactive approach

Proactive behavior is the transmission of rules by the controller before the router receives the associated packets. Several solutions have been developed based on the back-pressure algorithm adapted for data networks that are very sensitive to latency and especially QoS [15] [16].

The back-pressure algorithm is a well-known optimal flow algorithm. Its implementation requires each node to maintain a separate queue for each stream on the network, and only one queue is served at a time. Since the traffic in the data network is generally very large, maintaining the data structure of the queues at each node becomes complex. In [12], the authors propose an approach that allows each node on the network to maintain a single FIFO queue for each outgoing link, instead of keeping a separate queue for each stream on the network. They also propose an algorithm that forces back-pressure, to use the minimal amount of network resources while maintaining maximum throughput. However, for low arrival rates, the delays will be much higher. Therefore in [13], the authors propose to combine the routing algorithm from short path to back-pressure, in order to maintain several queues at each intermediate node for each flow and to ensure that the packets of one stream will reach the respective destination after crossing at most n nodes. This optimizes the packet delay among the presented back-pressure variants. But it overloads the nodes due to queues increasingly important. In [21], the authors propose a variant of the LIFO-based back-pressure algorithm for minimizing delays. But the main limitation of this method is that some first packets get trapped in the LIFO buffer indefinitely.

With the emergence of SDN and its increasing use in data networks, in [14], the authors rely on centralized network management with SDN and propose an improvement in back-pressure routing. Coupled with K's shortest route routing techniques to the destination to optimize transmission delays as well as packet delay delays on the network. This proposed

method shows a significant performance gain in terms of reducing packet delays, average queue length, average hop length while retaining the bit rate optimality property of the back-end routing algorithm basic pressure.

2.2 Reactive approach

In this approach, the router that receives a packet reports the event to the controller and receives back rules during an exchange, to decide on the transfer to the next appropriate router. Sun Uk Baek et al [13] propose to develop Openflow extensions on the controller side. The mechanism is composed of four modules. The topology module identifies physical connectivity at the OpenFlow switch links. The supervisory module monitors the usage and error rate of each device and link and collects and provides basic information for calculating the path. The path calculation module gives the priority of several paths in the information collected from the topology module and the supervision module. And the path installation module is used to apply priority flow table information to the OpenFlow switch.

Thus, in the search for solution based on the Openflow protocol extension, Channegowda et al., Have defined an approach based on an OpenFlow protocol extension on the Openflow switch side. They offer a pure OpenFlow extended approach, which includes an OpenFlow agent on each network element. The developed agent uses an internal API to communicate with the network elements, and extends the protocol to communicate with the developed OF controller.

3. ANALYSIS OF PROACTIVE AND REACTIVE METHODS

3.1. Methodological approach

For the realization of this work we adopted a methodology according to the define goal. This involves an evaluation of the transfer delay of the stream as well as the load generated for this flow transfer from the source to the destination, while considering the various routing methods.

a) Reactive method

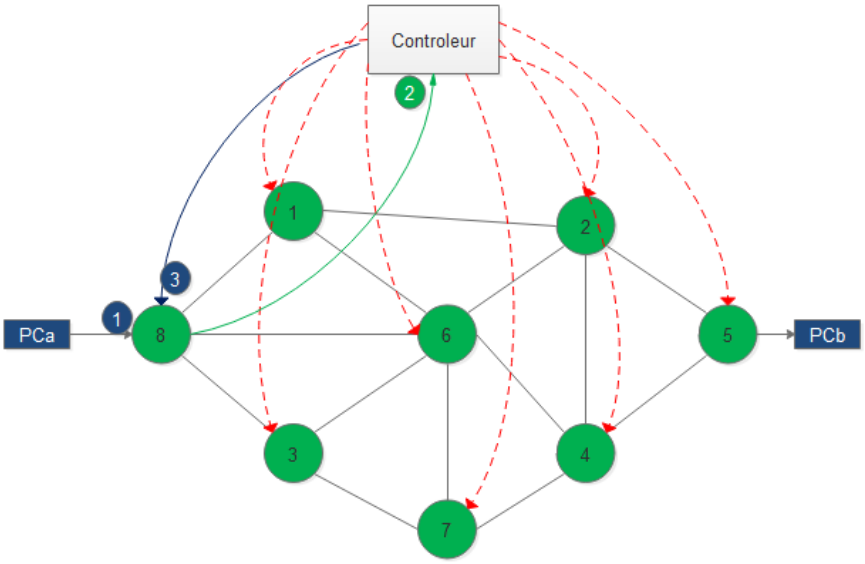

Fig. 2 Descriptive scheme of reactive approach

Observation: When the node (8) of the diagram of Figure 2 above, receives a packet from PCa for PCb (1), it transmits the request to the controller (2). The controller relies on the current topology of the network and communicates the transfer rule to the node (a) that executes it (3).

Operations (1), (2), (3) are thus repeated at each node until the packet arrives at destination (PCb).

Analysis: When requesting the transfer rule to the controller:

- Time elapses T = $t_{jc} + t_{cj}$ for (1) and (2) on different nodes requesting the controller. For n nodes of the path between source s and destination d, the transfer time is defined as follows:

$$T(s,d) = \sum_{j=1}^{n} \left(t_{jc} + t_{cj} + t_j \right) \qquad (1.1)$$

*c represents the controller

*t_{jc} represents the transmission delay of node j to the controller

*t_{cj} represents the transmission delay of the controller at node j

*t_j is the time to check the transfer rules in order to decide on the transfer

- The set of messages of the path having for source s and the destination d is defined by:

$$\alpha_{ch(s,d)} = \sum_{i=1}^{n} (\alpha_{ic} + \alpha_{ci}) + \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_{ij} \qquad (1.2)$$

*α_{ic} the number of messages exchanged between a node i and the controller c

*α_{ci} the number of messages exchanged between controller c and node i

*α_{ij} the number of messages exchanged between a node i and the node j

$ch(s,d)$: a path between the source s and the destination d

$\alpha_{ch(s,d)}$:number of messages exchanged between source and destination

$$\alpha_{ic} = \begin{cases} 1 \; if \;\; i \in ch(s,d) \\ 0 \; else \end{cases} \quad \text{and} \quad \alpha_{ci} = \begin{cases} 1 \; if \;\; i \in ch(s,d) \\ 0 \; else \end{cases} \quad \text{and}$$

$$\alpha_{ij} = \begin{cases} k \; if \;\; (i,j) \in ch(s,d) \\ 0 \; else \end{cases}$$

k *is* number of messages intended for j

b) Proactive method

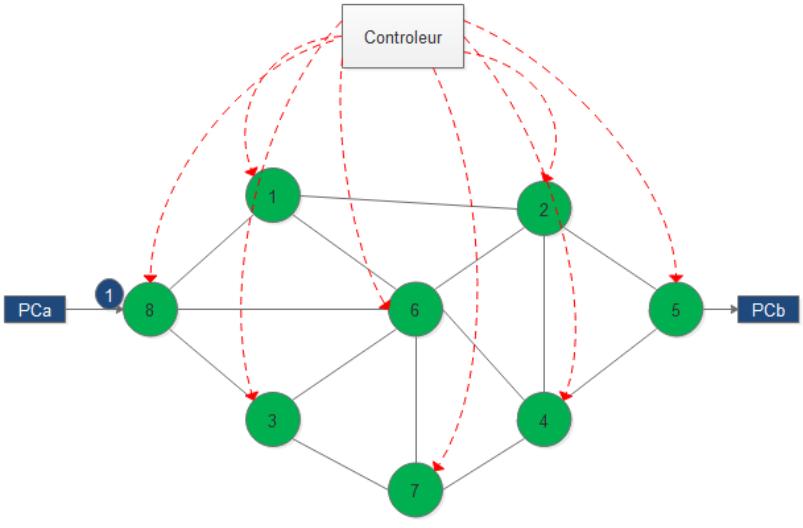

Fig. 3: Descriptive diagram proactive approach

Observation: When the node (8) of FIG. 3 receives a packet from the PCa destined for PCb (1), it executes the transfer rule appropriate for this type of packet and present in its table.

The operation (1) is thus repeated until the packet arrive at its destination.

Analysis: When checking the transfer rules from source to destination:

- There elapses a time t_j which varies according to the size of the transfer table of each node and j belonging to the path between s and **d**.

$$T(s,d) = \sum_{j=1}^{n} t_j \qquad (2.1)$$

- - The messages exchanged on the path with source s and destination d is defined by:

$$\alpha_{ch(s,d)} = \sum_{i}^{n} \sum_{j}^{n} \alpha_{ij} \qquad (2.2)$$

$$où\ \alpha_{ij} = \begin{cases} k \text{ if the link } (i,j) \in ch(s,d) \\ 0\, else \end{cases}$$

k is the number of messages intended for the node j

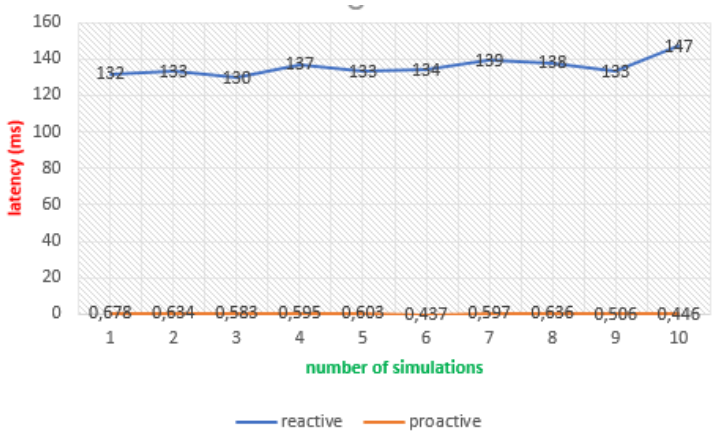

Fig. 4: Latency evaluation of reactive and proactive approaches

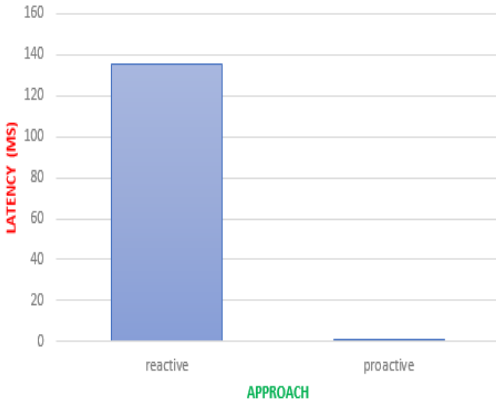

Fig. 5: evaluation of the average latency of reactive and proactive approaches

Considering all the messages exchanged during the transfer of the stream between the source and the destination; the results of the mathematical analysis and the simulation show; The proactive approach makes it possible to improve the transmission delay contrary to the reactive approach. However, suppose a proactive node restarts. It no longer has transfer rules and therefore cannot make a transfer decision if information arrives at that moment until the controller sends the transfer rules again. This can therefore have a significant impact on the transmission delay.

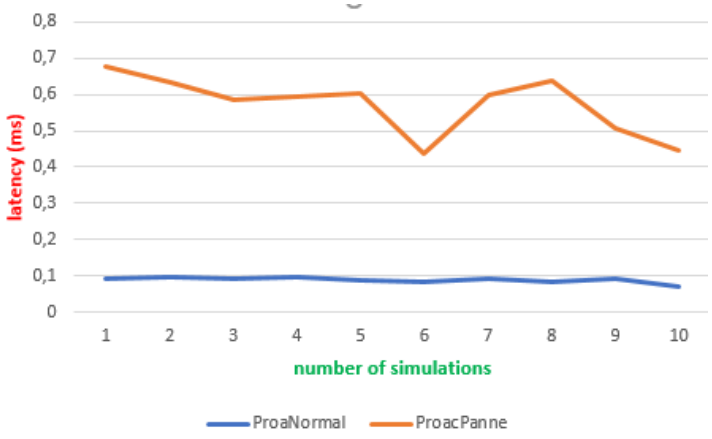

Fig. 6: latency evaluation proactive approach with incident (orange curve)

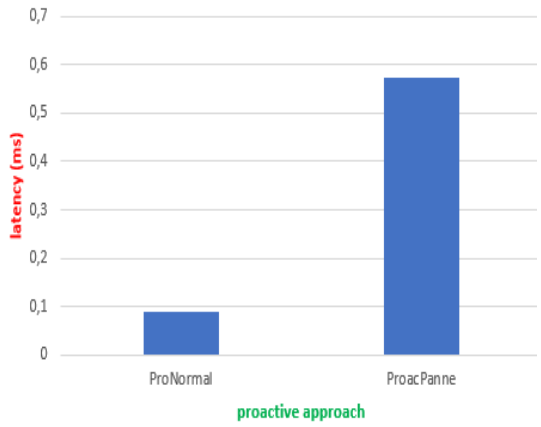

Fig. 7: average latency assessment proactive approach with incident

The proactive approach is not resilient to breakdowns. Indeed, we observe that, at each simulation, the latency of the proactive approach with failure always remains higher than that of the normal proactive approach. Because the node affected by the failure can make a transfer.

Analysis 2: When a node of the path restarts we observe the following:

- A time elapses t'_j which varies according to the retransmission delay of the transfer rules of each node and j belonging to the path between s and d. suppose that μ is the delay of new transfer rules from the controller. Suppose the controller sends the forwarding rules to the proactive nodes every q second. So, a proactive node that restarts, has a waiting time less than or equal to the delay set by the controller before sending the new transfer rules as follows decision if the controller does not transmit the transfer rules again:

  $0 < \mu \leq q$ and so $1 < 1+\mu \leq 1+q$. Thereby,

$$t'_j = \begin{cases} 1+\mu \leq 1+q \ \textit{with} \ \ j \ \textit{proactif} \\ 0 \ \textit{else} \end{cases}$$

And so, it is certain that the delay of transmission after the restart of a proactive node is important compared to the case where the transmission is done without reboot of nodes.

4. PROPOSED APPROACH

To realize the potential of programmable networks, we propose in this paper another variant of the back-pressure algorithm with SDN based on a mixed method. By considering all the messages exchanged between the source and the destination during the transfer of the stream in order to better estimate the delay. This is to optimize the routing functions by simply placing the traffic where the capacity allows, to avoid congestions of some parts of the network strongly solicited.

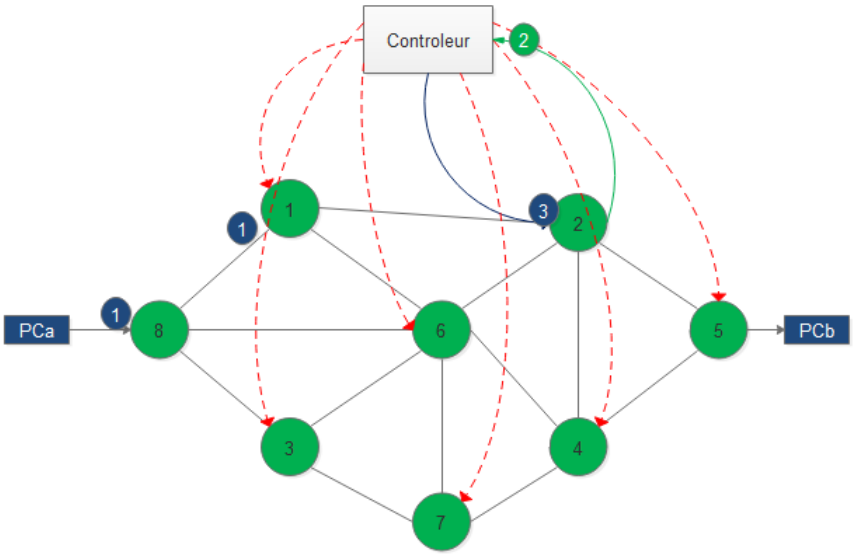

Fig. 8: physical topology with hybrid nodes

Observation: The operation of the mixed method is shown in Figure 4 above.

When a node receives a packet for a given destination, it checks whether it has a forwarding rule for that destination in its forwarding table and executes it if so (node 8). Otherwise a message (PACKET_IN) (node 2) is sent to the controller and the controller determines the satisfactory path then sends the transfer rules (PACKET_OUT) to all the nodes of the selected path to avoid situations where the transfer rules are obsolete.

Analysis: We will have here the sum of the analysis of the reactive method and that of the proactive method.

- Time of transmission of the information from the source to the destination.
Suppose that between s and d, there are n nodes:

$$T(s,d)=\sum_{j=1}^{|R|}\left(t_{jc}+t_{cj}\right)+\sum_{j=1}^{|P|}t_j \qquad (3.1)$$

*R is the set of reactive nodes

*P is the set of proactive nodes

$$t_{jc}=\begin{cases}1\ if\ \ j \text{ is reactive}\\ 0\ else\end{cases} \qquad t_{cj}=\begin{cases}1\ if\ \ j \text{ is reactive}\\ 0\ else\end{cases} \qquad t_j=\begin{cases}1\ if\ j \text{ is proactive}\\ 0\ else\end{cases}$$

- Messages during transmission from source to destination

$$\alpha_{ch(s,d)}=\sum_{i=1}^{n}(\alpha_{ic}+\alpha_{ci})+\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_{ij} \qquad (3.2)$$

$$\alpha_{ic}=\begin{cases}1\ if\ \ the\ node\ \ i\in ch(s,d)\\ 0\ else\end{cases};\ \alpha_{ci}=\begin{cases}1\ if\ the node\ i\in ch(s,d)\\ 0\ else\end{cases};$$

$$\alpha_{ij}=\begin{cases}k\ \ \text{if the link}\ (i,j)\in ch(s,d)\\ 0\ else\end{cases}$$

k the size of the messages for the node j

4.1 PROPOSED ALGORITHM

The algorithm we propose is described in the following points:

ALGORITHM: ***ReaPro***	
	Input: Network topology
	Output: Transfer rules
	---- Processing a node that receives data to be transferred to the destination -----
1	- *If a packet arrives at a node j, the node checks if there is a transfer rule for its destination and applies it. Otherwise, he contacts the controller.* - *End if* - *Go to step 2 if I am not the destination otherwise STOP.*
	---- Controller processing -----
2	- *Determine the valuations of the various links on the network;* - *Determine the optimal path using the Dijkstra algorithm at time t;* - *Communicate the transfer rule to the node that initiated the request as well as the nodes starting from the selected path to reduce the transfer tables of the other nodes.* - *Go to step 3*
	---- Processing a node that receives a transfer rule -----
3	- *The node executes the transfer rule and sends the packet to the next node to the destination*

The actions from 1 to 3 are repeated until the destination.

5. CONCLUSION

This article proposes a variant of the data-based back-pressure algorithm with SDN based on a mixed approach. The results of the mathematical analysis and the simulation show at the same time that the algorithm we propose will improve the performance of the data network with SDN if all the messages exchanged during the transfer of the information are taken into account. Because the main problem in networks with SDN is that the SDN routing tables can contain only a very limited number of rules.

REFERENCES

[1] Beacon. https://openflow.stanford.edu/display/Beacon/Home.

[2] Controller performance comparisons. http://www.openflow.org/wk/index.php/ Controller

Performance Comparisons.

[4] Devolved Control of ATM Networks. http://www.cl.cam.ac.uk/research/srg/netos/old-projects/dcan/#pub.

[5] Floodlight, an open sdn controller. http://floodlight.openflowhub.org/.

[6] C.Hopps, Analysis of an Equal-Cost Multi-Path Algorithm, RFC 2992, 2000.

[7] Timur Friedman Brice Augustin and Renata Teixeira. Measuring multipath routing in the internet. In IEEE/ACM Transactions on Networking, vol. 19, issue 3, pp. 830-840, June 2011.

[8] Nithin Michael, Ao Tang, and Dahai Xu. Optimal link-state hop-by-hop routing. In ICNP, pages 1–10, 2013.

[9] Pascal Mérindol, Routage multichemins par interface d'entrée. PhD thesis, Université Louis Pasteur, Strasbourg, 2008.

[10] Rodolfo Alvizu, all**,** Differential delay constrained multipath routing for SDN and optical network, Janvier 2015.

[11] Sun Uk Baek, Hyun Ho Shin, Dong-Ryoel Shin, Consideration of Multi-path Routing over SDN-enabled Network**,** 2016.

[12] L. Bui, R. Srikant, and A. Stolyar, "Novel architectures and algorithms for delay reduction in back-pressure scheduling and routing," in INFOCOM 2009, IEEE, 2009, pp. 2936–2940.

[13] L. Ying, S. Shakkottai, A. Reddy, and S. Liu, "On combining shortestpath and back-pressure routing over multihop wireless networks," IEEE/ACM Transactions on Networking (TON), vol. 19, no. 3, pp. 841–854, 2011.

[14-12] Kulkarni and Venkataramana Badarla, On Multipath Routing Algorithm for Software Defined Networks, *Siddharth S,* 2014.

[15] Stefano Vissicchio, Laurent Vanbever, and Olivier Bonaventure. Opportunities and research challenges of hybrid software defined networks. ACM Computer Communication Review (Editorial Zone), 44(2), April 2014.

[16] Seweryn Dynerowicz and Timothy G. Griffin. On the forwarding paths produced by internet routing algorithms. In ICNP, pages 1–10, 2013.

[17] I.Gojmerac, P.Reichl, L.Jansen, Towards Low-complexity Internet Traffic Engineering: The Adaptive Multi-Path Algorithm, Journal of Computer Networks, Vol. 52, No. 15, pp. 2894–2907, December 2008.

[18] Doreid Ammar, Plan de connaissance pour les réseaux sémantiques : application au contrôle d'admission, PhD thesis, Université de Lyon, December 2012.

[19] Frédéric Havet, Nicolas Huin, Joanna Moulierac, Khoa Phan. Routage vert et compression de règles SDN. ALGOTEL 2015 – 17émes Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications, Jun 2015, Beaune, France.

[20] V. Badarla, V. G. Subramanian, and D. J. Leith, "Low-delay dynamic routing using fountain codes," IEEE Communications Letters, vol. 13, no. 7, pp. 552–554, 2009.

[21] S. Moeller, A. Sridharan, B. Krishnamachari, and O. Gnawali, "Routing without routes: The backpressure collection protocol," in Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks, 2010, pp. 279–290.