Journal of Engineering Science and Technology
Vol. 14, No. 5 (2019) 3062 - 3072
© School of Engineering, Taylor's University

HYBRID ROUTING WITH LATENCY OPTIMIZATION IN SDN NETWORKS

GILLES A. S. KEUPONDJO[1,2,3,*], NOGBOU G. ANOH[3], JOEL C. ADEPO[3], SOULEYMANE OUMTANAGA[2,3]

[1]Ecole Doctorale Polytechnique de l'INP-HB UMRI 78: EEA, Côte d'Ivoire
[2]Institut National Polytechnique Félix Houphouët Boigny de Yamoussoukro, Côte d'Ivoire
[3]Laboratoire de Recherche en Informatique et Télécommunications, Côte d'Ivoire
*Corresponding Author: armel.keupondjo@inphb.ci

Abstract

SDN networks opt for a centralized approach, in which, the controller is the element that defines the overall network management policy and communicates it to the OpenFlow switches via the OpenFlow protocol whenever it is requested by the latter or that he deems it expedient. This strongly impacts the latency and packet loss, which are of paramount importance for multimedia applications. In order to optimize the transmission time in the data networks with SDN, two approaches are generally proposed; reactive approach and proactive approach. However, we note that the reactive approach, although it allows to better evaluate the quality of the optimum paths, increases the latency time. While the proactive approach greatly reduces this time, it does not consider the parameters such as the failure of a node that is part of the transfer path. We will propose in this document a mixed approach based on the algorithm backpressure, which is proven as a routing algorithm and optimal flow planning. This to optimize the routing functions by simply placing traffic where the capacity allows it, to avoid the overloads of certain parts of the network strongly requested. The results of our simulations show that our proposal allows a considerable reduction in latency and better controls the transfer tables of the different nodes.

Keywords: Latency, Multi-path, Routing, SDN.

1. Introduction

In conventional networks, routers determine using distributed routing protocols on which, output interface to direct packets for a given destination. In SDN networks, one or more controllers support the calculation of roads and routers are reduced to simple dispositive of transmission. Figure 1 shows the routing of a packet between a source and a destination in SDN network with the Openflow protocol [1-3].

The controller sends a transfer rule to Router A (1) so that when a packet arrives at this router (2), with a destination IP address that is included in the network IP address of the transfer rule, the router will immediately know, which interface to transfer The package (proactive method) (3). Router B, having no management rules concerning the package reached its level will contact the controller (4) In order to know the attitude to be adopted in relation to the package, the controller lowers a transfer rule (5) about the package and the router is then able to transfer the packet to the next node of the network (6) (reactive method).

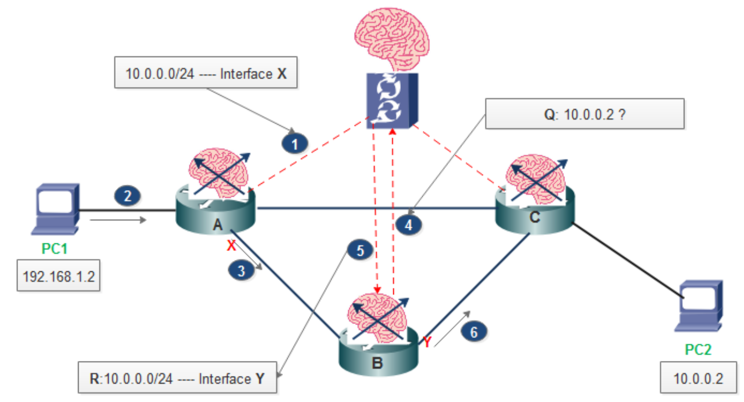

Fig. 1. Schema of a simple SDN network.

The collaboration between network equipment is based on the exchange of messages. Doreid [4] and Havet et al. [5] mentioned that these messages might be upstream warnings requiring, for example, a reduction in flow, as introduced. It can also be imagined to be messages allowing the admission of new feeds [6], or disseminating information related to the state of the router.

However, the main purpose of traffic engineering is to avoid the congestion of some heavily-solicited 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 failures) and demand. Because the current bottleneck in communications networks like, data networks and cloud computing is due to the increasingly important traffic that can affect the quality of service.

To optimize the transmission time in the data networks with SDN, Proactive routing is very often used in relation to reactive routing.

Proactive behaviour is the transmission of rules by the controller before the router receives the associated packages. Several Solutions have been developed and are based on the backpressure algorithm adapted for networks data that is very sensitive to latency and especially to QoS [7].

2. Related Work

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 traffic in the data network is generally very large, maintaining the data structure of the queues at each node becomes complex. Gojmerac et al. [8] 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 offer an algorithm that forces back-pressure to use the minimum amount of network resources while maintaining maximum throughput. However, for low arrival rates, the delays will be much higher. That is why Vissicchio et al. [9] propose to combine the shorter path to backpressure routing algorithm, in order to maintain multiple queues at each intermediate node for each stream and ensure that packets of a stream will reach the respective destination after crossing at most n knots. This makes it possible to optimize the package delay among the back-pressure variants presented. But it overloads the nodes due to the increasingly important queues of expectations. Moeller et al. [10] propose a variant of the backpressure algorithm based on LIFO, for the minimization of deadlines. But the main limitation of this method is that some first packets find themselves trapped in the LIFO buffer indefinitely.

With the emergence of SDN and its increasing use in data networks, Dynerowicz and Griffin [11] rely on centralized management of the network with SDN and propose an improvement in back-pressure routing. Coupled with *K* routing techniques shorter paths to the destination to optimize transmission delays as well as delays results of packet looping on the network. This proposed method shows a significant gain in performance in terms of packet delay reduction, average queue length, average jump length while maintaining the flow optimality property of the routing algorithm Basic back-pressure.

2.1. Proactive method analysis

2.1.1. Methodological approach

For the realization of this work, we have adopted a methodology according to the objective to be achieved. This is an assessment of the flow transfer time as well as the load generated for this flow transfer from the source to the destination while considering the different routing methods.

2.1.2. Observation

In a proactive approach, the controller is responsible for transmitting the transfer rules to the switches before they receive a stream to be transmitted, as shown in Fig. 2 by the messages (1) sent by the controller.

When the node (A) receives a packet (2) from PC1 to PC2, it executes the appropriate transfer rule for the destination concerned if this rule is present in its transfer table.

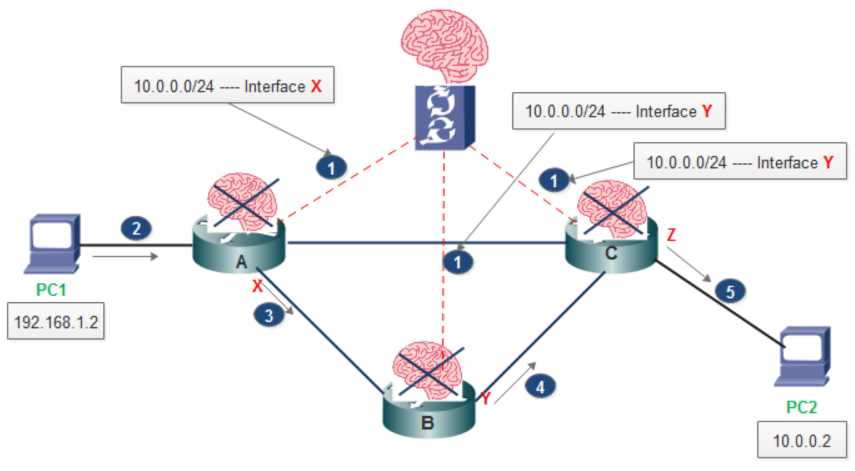
10.0.0.0/24 ---- Interface X
10.0.0.0/24 ---- Interface Y
10.0.0.0/24 ---- Interface Y
1
1
1
2
3
4
5
A
B
C
X
X
Y
Z
PC1
192.168.1.2
PC2
10.0.0.2

Fig. 2. Descriptive schema proactive approach.

The operation (2) is repeated on each node starting from the path until the stream arrives at its destination.

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

There is a time t_j, which varies depending on the size of the transfer table of each node and J belonging to the path between source *s* and the destination *d*.

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

$$where\ t_j = \begin{cases} 1\ if\ \ j\ is\ a\ \ proactive\ node \\ 0\ else \end{cases}$$

Messages exchanged on the path with the source *s* and destination *d* is defined by:

$$\alpha_{ch(s,d)} = \sum_{i}^{n}\sum_{j}^{n}\alpha_{ij} \tag{2}$$

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

and *k* is the number of messages for node *j*.

2.2. Experimental evaluation of the proactive approach

In order to swallow this approach, we have emulated the WAN topology (Fig. 3), on a physical machine. The latter is characterized by an Intel Core I5 4 processor running at 2.53 GHz and a RAM equal to 8 GB. The machine that simulates the virtual network uses Mininet in its version 3.2, which allows creating the SDN topology considered.

We will evaluate the architecture of Fig. 3 (which will be executed in Mininet, so that each node can connect to the controller), the impact that the proactive approach can have on latency. We do not consider any failures in the network**.**

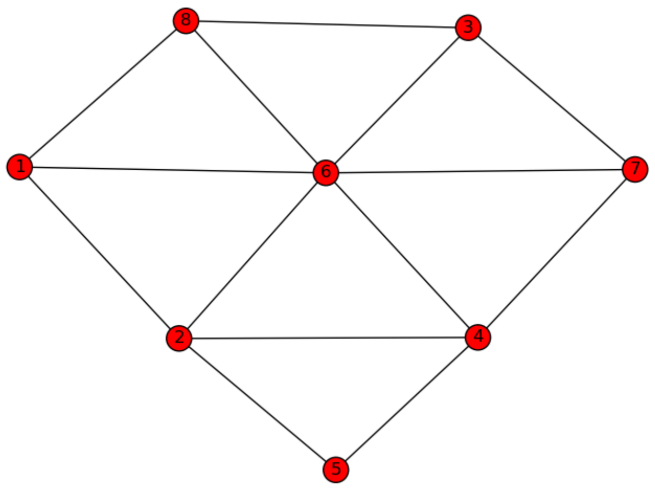

Fig. 3. Simulation topology.

Our tests are made with the Ryu controller under its version 3.2. It is based on Python and Provides APIs for communicating between applications and network infrastructure. In addition, the protocol used for communication between the controller and switches is OpenFlow 1.3, as well as the Python3 programming language. In addition, we use the Ping and Cbench tools to measure the network performance metrics.

The simulation followed an iterative process and the simulation process is repeated ten (10) times. In order to obtain statistically significant measurements for each execution, we start the controller, then we start the topology under Mininet so that each node can connect to the controller. As soon as the different nodes establish a connection with the controller, we eject a flow of traffic into the network. And we raise the latencies.

Thus, we were able to obtain the results of Fig. 4.

These results show that when there is a failure in the network, the time the latency of the proactive method increases considerably (BProUP Figs. 4 and 5) because it is not resilient to failures.

The proactive approach is not resilient to failures (BproUP Figs. 4 and 5). Indeed, we note that, at each stage of the simulation, the latency of the proactive approach with failure is always higher than that of the normal proactive approach.

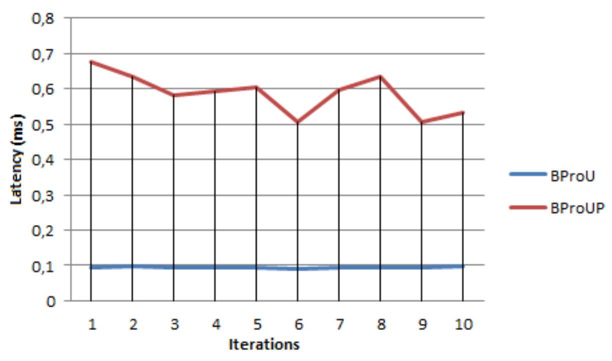

Fig. 4. Latency analysis with failure.

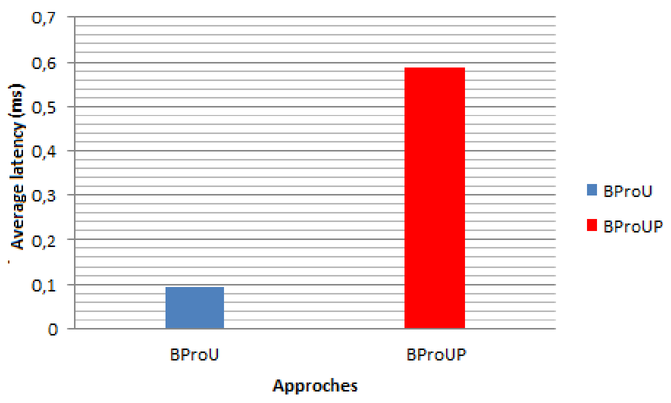

Fig. 5. Latency analysis way with fault.

Because the node affected by the failure can make a transfer decision if the controller does not re-transmit the transfer rules.

When a node in the path restarts:

There is a time *t'*$_j$, which varies depending on the time to retransmit the transfer rules of each node and J belonging to the path between *S* and *D*. Suppose that μ is the time to wait for new transfer rules on the part of the controller. Suppose the controller sends the transfer rules to the proactive nodes each *q* seconds. Thus, a proactive node that restarts has a timeout less than or equal to the delay put by the controller before sending the new transfer rules as follows:

$0 < \mu \leq q$ and so $1 < 1 + \mu \leq 1 + q$. As a result:

$$t'_j = \begin{cases} 1 + \mu \leq 1 + q \ \ with\ \ j\ proactive \\ 0\ else \end{cases} \tag{3}$$

Hence, it is certain that the transmission delay after restarting a proactive node is important compared to the case where the transmission is done without restarting nodes.

Thus, our hybrid approach, in which, we present in the next section.

3. Proposed approach

To realize the potential of programmable networks, we propose in this document another variant of the backpressure algorithm with SDN based on a mixed-method. By taking account of all the messages exchanged between the source and the destination when transferring the stream for a better evaluation of the delay. This in order to optimize the routing functions by simply placing traffic where the capacity allows it, in order to avoid the overloads of certain parts of the network strongly requested.

The operation of the mixed method works is shown in Fig. 6.

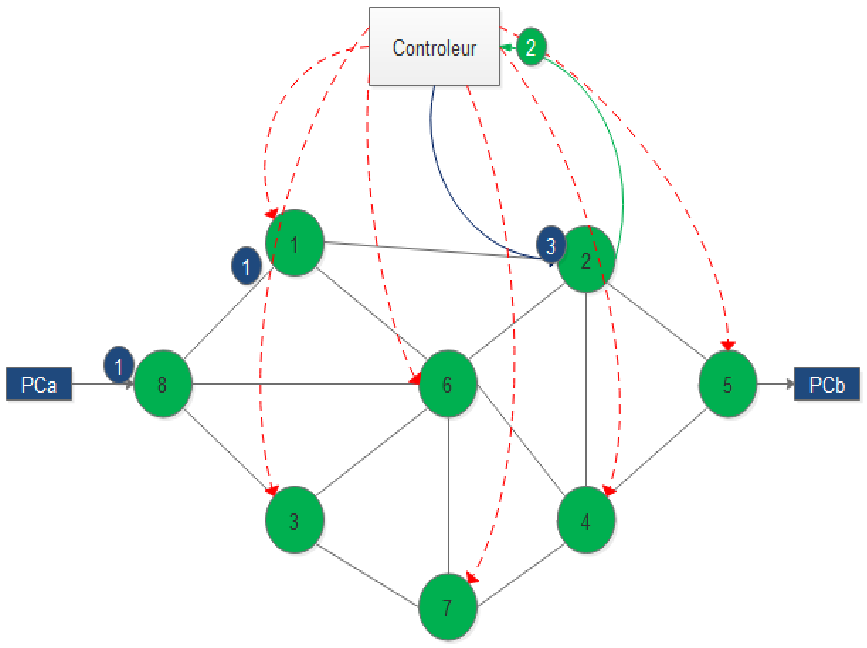

Fig. 6. Descriptive drawing mixed approach.

When a node receives a packet (1) for a given destination, it checks to see if it has a transfer rule for this destination from its transfer table and executes it if YES (Node 8). Otherwise, a message (PACKET_IN) (2) is sent to the controller and the controller determines the satisfactory path and then sends the transfer rules (PACKET_OUT) (3) to all nodes in the selected path to avoid situations where the transfer rules are obsolete.

In the case of hybrid routing, the transmission time takes into account the time taken by the nodes with proactive behaviour and that of the nodes with responsive behaviour.

The time for transmitting information from the source *s* to the destination *d* is given by the Eq. (4).

Suppose that between *s* and *d*, there are *n* nodes:

$$T(s,d) = \sum_{j=1}^{|R|}\left(t_{jc} + t_{cj}\right) + \sum_{j=1}^{|P|} t_j \tag{4}$$

where: *R* is the set of reactive nodes, *P* is the set of proactive nodes, and *C* is the controller

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

$$t_j = \begin{cases} 1 \; if \;\; j \; is \; proactive \\ 0 \; else \end{cases}$$

In order to better evaluate the transmission delay, we analyse the number of messages emitted between the numbers of posts emitted in the network because indeed, a large number of messages emitted in the network can degrade the transmission delay.

To evaluate the number of messages we consider the messages exchanged between the controller and the switches and the messages transmitted from one switch to another.

The messages issued during the transmission of the source *s* to the destination *d* are determined by the Eq. (5).

$$\alpha_{ch(s,d)} = \sum_{i=1}^{n}(\alpha_{ic} + \alpha_{ci}) + \sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_{ij} \tag{5}$$

where *ch*(*s*, *d*) represents the path between the source *s* and the destination *d*:

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

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

$k \; the \; number \; of \;\; messages \;\; for \;\; node \;\; j$

4. Proposed Algorithm

When a stream is received, the node verifies that there is a transfer rule for it in its flow table and executes it. Otherwise, the node contacts the controller to get the action to hold for that stream. The controller observes the topology at the time of the request, defines the conduct to be held, and communicates it to the requested node as well as to the different nodes from the transmission path.

We note here that nodes have responsive and proactive behaviour. Hybrid behaviour for a node is to execute a transfer rule that is already present in its flow table (proactive behaviour). If the rule is not present in the transfer table, then the node contacts the controller to get it (responsive behaviour). The algorithm of the proposed approach is presented in Table 1.

Table 1. Algorithm proposed.

	ALGORITHM: ReaPro-CU
	Input : Network topology. **Output :** The transfer rules.
	Processing a node that receives a data to be transferred to the destination
1	- If a packet arrives at a *j* node, the node checks to see if there is a transfer rule for its destination and applies it. Else, he contact the controller. - End if. - Go to step 2, if *j* is not the destination else STOP.
	Controller processing
2	- Determine the valuations of the various links on the network. - Determine the optimal path using back-pressure at the moment *t*. - Communicate the transfer rule to the node that initiated the query as well as the nodes 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.

*We repeat the actions from 1 to 3 to the destination.

5. Simulation and Results

The tools used for our simulations remain the same as those mentioned in section 2.2. Our main objective here is to show that our hybrid approach gives better results in terms of transmission delay than those of the proactive approach in case of failure on the flow transfer path. We consider as the main factor, the delay taking into account all the messages of solicitation of the controller to decide the conduct to be held for an unknown packet and the set of messages exchanged between two ends during the transmission. Because it is important to minimize the exchange of rule, solicitation messages in the network because they act much in the controller-side request-processing rate.

The results of the simulation show that our solution (ReaPro-CU Figs. 7 and 8) approach, each time offers better latency, compared to the proactive approach (ProPan Figs. 7 and 8). Nevertheless, we note that some points have almost the same results. Hence, when there is a failure on a transmission path, it becomes apparent that the timeout is less than or equal to the delay put by the controller before sending the new transfer rules. Therefore, if the transmission time is equal to the timeout, it makes sense for this relevant node to wait for the transfer rule.

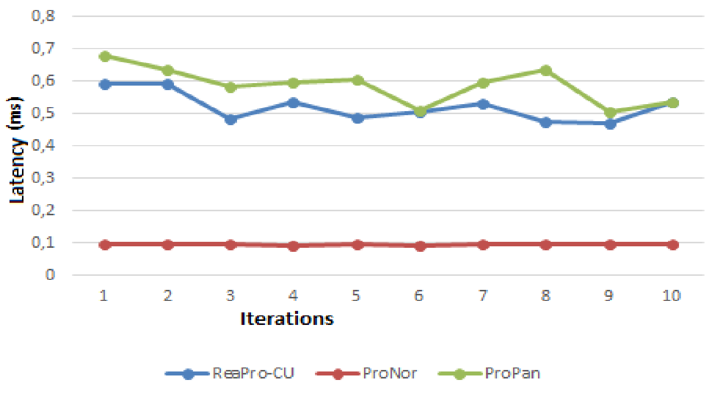

Fig. 7. Latency with failure and mixed solution.

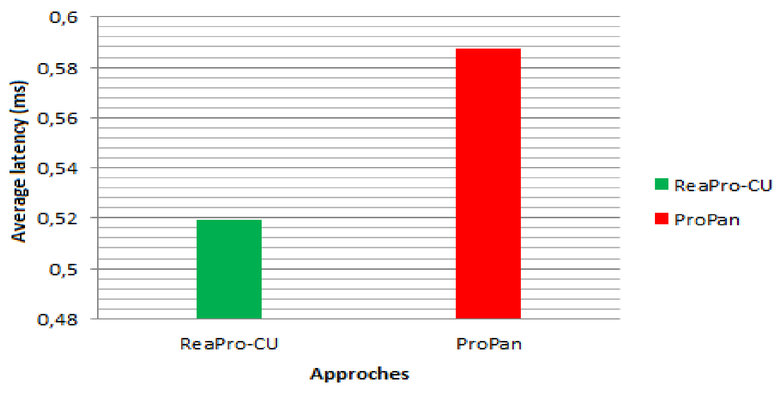

Fig. 8. Average latency with fault and mixed solution.

6. Discussion

Suppose a node *j* decides to contact the controller for new transfer rules. The time to get the rule t_p is estimated by:

$$t_p = t_{jc} + t_{cj} + t_c \tag{6}$$

where t_{jc} is the delay for the node *j* to contact the controller *c*, t_c is the time taken by the controller to process the request of the node and t_{cj} the time taken to transmit the transfer rule to the node.

If $\mu > t_p$ in the strictly proactive case, we have to wait a while μ. However, in this case, it is interesting to request the transfer rules to receive them within a period t_p.

7. Conclusion

This paper proposes a variant of the SDN-based data network backpressure algorithm based on a hybrid approach. The results of the mathematical analysis and the simulation show that the algorithm (ReaPro-CU) that we propose improves 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 of networks with SDN is that the SDN routing tables can only contain a very limited number of rules.

Because, to reduce the latency observed with the BproUP approach, we have proposed another approach (ReaPro-CU) that significantly improves the latency when a failure occurs in the network. The results of the simulation show us that the latency of our approach (ReaPro-CU), remains much lower than that of Back-pressure with failure (BProUP).

Indeed, when a failure occurs in the network (restart of a node) during the transmission of the flow, our approach (ReaPro-CU) proposes that the concerned node contacts the controller instead of waiting for the moment when the controller gives him the new rules. In addition, when the node contacts the controller, the controller communicates the transfer rules to all the nodes coming from the destination path, therefore, they cannot contact it if they do not have the rule transfer for the destination. These principles implemented in our routing approach can significantly reduce the latency when there is a failure in the network and also controls the size of the node transfer tables on the network.

References

1. OpenFlow. (2013). Beacon. Retrieved February 4, 2016, from https:// openflow.stanford.edu./display/Beacon/Home.
2. Fernandez, M.P. (2013). Comparing OpenFlow controller paradigms scalability. Reactive and proactive. *Proceedings of the IEEE 27th International Conference on Advanced Information Networking and Applications (AINA)*. Barcelona, Spain, 1550-4456.
3. Project Floodlight. (2010). Project floodlight. Retrieved March 10, 2016, from http://openflowhub.org/.
4. Doreid, A. (2012). *Plan de connaissance pour les réseaux sémantiques : Application au contrôle d'admission*. Ph.D. Thesis. University Claude Bernard Lyon 1, Lyon, France.
5. Havet, F.; Huin, N.; Moulierac, J., and Phan, K. (2015). Routage vert et compression de règles SDN. *ALGOTEL - 17émes Rencontres* Francophones sur les Aspects Algorithmiques des Télécommunications. Beaune, France, 1-4.
6. Badarla, V.; Subramanian, V.; and Leith, D.J. (2009). Low-delay dynamic routing using fountain codes. *IEEE Communications Letters*, 13(7), 552-554.
7. Kulkarni; S.S.; and Badarla, V. (2014). On multipath routing algorithm for software defined networks. *Proceedings of the International Conference on Advanced Networks and Telecommunications Systems*. New Delhi, India, 1109-1111.
8. Gojmerac, I.; Reichl, P.; and Jansen, L. (2008). Towards low-complexity internet traffic engineering: The adaptive multi-path algorithm. *Computer Networks*, 52(15), 2894-2907.
9. Vissicchio, S.; Vanbever, L.; and Bonaventure, O. (2014). Opportunities and research challenges of hybrid software defined networks. *ACM SIGCOMM Computer Communication Review*, 44(2), 70-75.
10. Moeller, S.; Sridharan, A.; Krishnamachari, B.; and Gnawali, O. (2010). Routing without routes: The backpressure collection protocol. *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*. New York, United States of America, 279-290.
11. Dynerowicz, S.; and Griffin, T.G. (2013). On the forwarding paths produced by internet routing algorithms. *Proceedings of the 21st IEEE International Conference on Network Protocols (ICNP)*. Goettingen, Germany, 1-10.