INTERNATIONAL JOURNAL OF COMPUTERS COMMUNICATIONS & CONTROL
ISSN 1841-9836, 13(5), 824-836, October 2018.

Optimization of the Latency in Networks SDN

G.A. Keupondjo Satchou, N.G. Anoh, T. N'Takpé, S. Oumtanaga

Gilles Armel Satchou Keupondjo*
Laboratoire de Recherche en Informatique et Télécommunications (LARIT)
Institut National Polytechnique Félix Houphouët Boigny de Yamoussoukro, Côte d'Ivoire
*Corresponding author: armel.keupondjo@inphb.ci

Nogbou Georges Anoh
1. Laboratoire de Recherche en Informatique et Télécommunications (LARIT)
Institut National Polytechnique Félix Houphouët Boigny de Yamoussoukro, Côte d'Ivoire
2. Unité de Recherche et d'Expertise Numérique (UREN)
Université virtuelle, Côte d'Ivoire
georges.anoh@uvci.edu.ci

Tchimou N'Takpé
1. Laboratoire de Recherche en Informatique et Télécommunications (LARIT)
Institut National Polytechnique Félix Houphouët Boigny de Yamoussoukro, Côte d'Ivoire
2. Laboratoire de Mathématique et Informatique (LMI)
Université Nangui Abrogoua , Côte d'Ivoire
tchimou.ntakpe@gmail.com

Souleymane Oumtanaga
Laboratoire de Recherche en Informatique et Télécommunications (LARIT)
Institut National Polytechnique Félix Houphouët Boigny de Yamoussoukro, Côte d'Ivoire
oumtana@gmail.com

Abstract: Unlike traditional networks, software-defined networking (SDN) are characterized by a physical separation of the control and the transfer plan. Thus, a centralized controller communicates the functions of control plan to each device via the OpenFlow Protocol whenever he is asked or that it deems appropriate. This impact strongly the latency time which is important for new services or multimedia applications. In order to optimize the time of transmission in network data with SDN, the proactive approach based on the algorithm back - pressure is usually offered. However, we note that the proactive approach while reducing strongly this time, does not account settings such as the failure of a node part of the way to transfer or the breaking of a bond that greatly increases the latency time. In this document, we will propose a joint routing approach based on proactive and reactive routing. This in order to optimize the routing functions by simply placing the traffic where capacity allows, in order to avoid congestion of highly stressed parts of the network taking into account the failures and significantly reduce the time latency. Simulation results show that our proposal allows a considerable reduction of latency even when there are failures in the network.
Keywords: Software-defined Networking (SDN), routing, multi-path, latency.

1 Introduction

Conceptually, a router is characterized by a control plan, and a transfer plan. As a result, the routers are responsible for the supervision of the topology of the network and the transfer of packages using static, or dynamic routing protocols. What is often at the origin of a load of significant calculation. In networks SDN (Software-Defined Networking) this charge is the responsibility of one or several controllers [2, 9, 12, 14].

As a result, a centralized controller communicates control plane functions to each device via the OpenFlow protocol [17] whenever it is requested or deemed appropriate [4, 16]. This has a significant impact on latency, which is important for multimedia applications. In order to optimize the transmission delay in data networks with SDN, two routing approaches are generally proposed; such as the proactive approach and the reactive approach. However, with the reactive approach, resource utilization is optimal and this approach is resilient to outages; which is not the case for the proactive approach which greatly reduces the transmission delay. However, the proactive approach does not take into account parameters such as the failure of a node in the transfer path or the break of a link. But these failures greatly increase the latency. Despite the incremental deployment of new routing solutions adapted to modern uses of the Internet [5] facilitated by SDN networks. It is in this context that we propose in this article a mixed routing approach that will reduce latency in case of network failures. This makes it possible to optimize the routing functions by simply placing the traffic where the capacity allows it, in order to avoid the congestions of some parts of the network that are heavily loaded, taking into account faults and considerably reducing the latency.

2 Related works

The algorithm back - pressure [8, 11, 13] is a well-known optimal flow algorithm. It refers to an algorithm to dynamically route traffic over a network to multiple bonds using gradients of congestion.
Its implementation requires that each node maintains a separate queue for each stream on the network, and a single queue is served at a time. As traffic in the data network is usually very large, maintaining the data structure of the queues at each node becomes complex. In [1], the authors propose an approach allowing each node on the network to maintain a single queue that implements the FIFO (First In First Out) approach for each outbound links, instead of keeping a separate queue each stream on the network. They also propose an algorithm that force back - pressure, to use the minimum amount of network resources while maintaining maximum throughput. However, for low flow rates arrived, delays will be much higher. That's why in [15], the authors propose to combine the algorithm of routing of shortest path to back - pressure, in order to maintain several queues at each intermediate node for each stream and ensure that packets of a stream will reach the respective after crossing more than destination n nodes. This allows to optimize the delay of package among the variants of back - pressure presented. But it overload due to queues of more and larger nodes. In [10] , the authors propose a variant of the algorithm of back - pressure based on the LIFO (Last In First Out) approach for minimization of the time. But the main limitation of this method is that some early packages get trapped in the LIFO buffer indefinitely.
With the emergence of SDN and its use of more and more in networks of data in [6], the authors rely on the centralized management of the network with SDN and offer improved routing back - pressure. Associated with techniques of shortest paths to the destination routing to optimize transmission times and delays results in looping of packets on the network. This proposed method shows a significant gain in performance in terms of reduction of delays in packets, length of queue average, average length of jump while retaining ownership of optimality of the routing algorithm flow back - pressure base.
However, suppose that a node reboots into the network. It has more rules of transfer and so cannot make a decision of transfer if a flow happens at that moment as long as the controller will again provide transfer rules. This can significantly impact the time of transmission. And also one of the problems in data with SDN network is control of the size of the table of rules of transfers.

The question that arises is how to optimize latency in the face of the problem of power outages on the network.

3 The SDN paradigm

In networks SDN, one or several controllers have supported the calculation of roads, thereby routers are reduced to simple devices of transmission. Figure 5, shows the routing of a package between a source and a destination in SDN network with Openflow Protocol.
The controller sends a transfer rule to the router A (1) so that when a package arrives at this router (2), with an IP destination which is included in the transfer rule network IP address, the router will know immediately by what interface transfer the package (proactive approach). Router B, having no management rule concerning the package reached his level, will contact the controller (4) to find out the attitude compared to the package. The controller sends a transfer rule (5) concerning the package and the router is then able to transfer the package to the next node of the network (6) (reactive method).

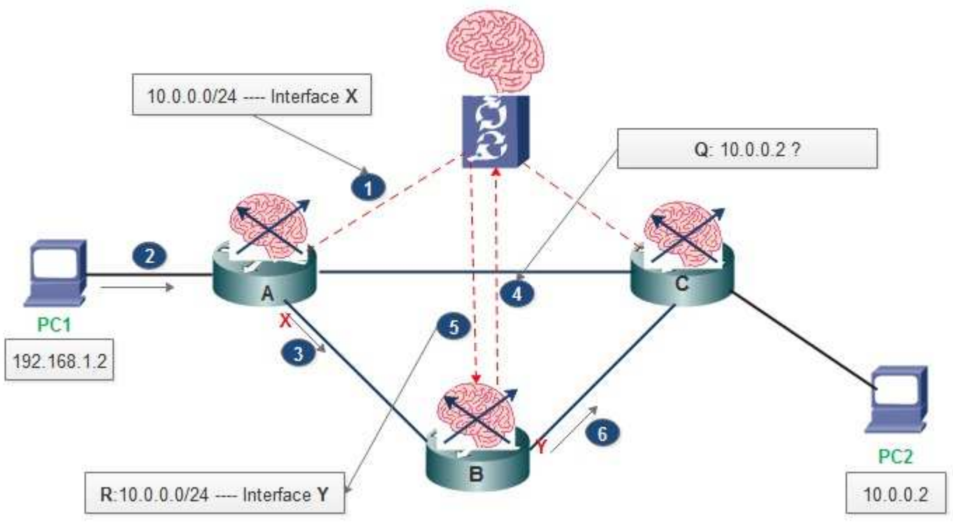

Figure 1: Diagram of a simple SDN network

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) [3]. The challenge here is to adapt well to the dynamic character of the topology (case of breakdowns) and the demand.

3.1 Analysis of reactive and proactive methods

For the realization of this work we have adopted a methodology following the goal. It is, to make an assessment of the transfer time of the stream as well as load produced for this transfer of flow from the source to the destination, while considering the different routing methods.

Reactive method

In this approach, each time the router or the node receiving a package, reports the event to the controller and receives in return rules in an Exchange, in order to decide on the transfer to the appropriate following router.
An example on an architecture represented by figure 2 below.
When the node (A), receives a package from PC1 to PC2 (1), it passes the request to the

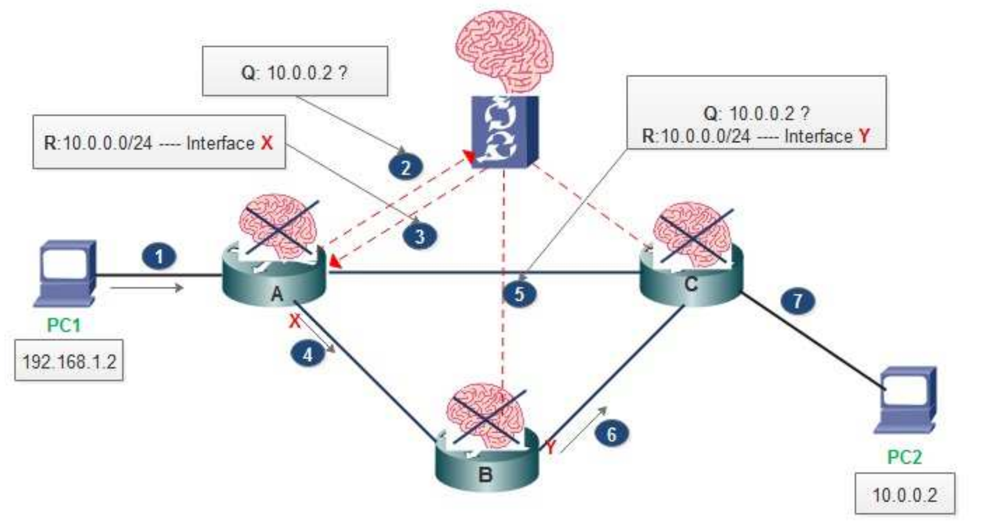

Figure 2: Descriptive diagram approach reactive

controller (2). The controller relies on the current topology of the network and communicates the transfer rule to the node (a) that runs (3).
Operations (1), (2), (3) are thus repeated in each node until the package arrives at its destination (PC2).
We evaluate the deadline for transmission and all the messages exchanged during the transfer of the flow from the source to the destination.
During the application of the rule of transfer to the controller:

- It takes a time $T = t_{jc} + t_{cj}$ for (1) and (2) on different nodes requesting the controller. For n nodes in the path from the source s to destination d, the transfer time is defined as follows:

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

- c represent 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 verification time of the transfer rules in order to decide on the transfer

- All of the messages of the path with the source s and the destination d is define by :

$$\alpha_{ch(s,d)} = \sum_{i=1}^{n}(\alpha_{ic} + \alpha_{ci}) + \sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_{ij} \tag{2}$$

- c represent the controller
- α_{ic} the number of messages exchanged between a node i and the controller c
- α_{ci} the number of messages exchanged between the controller c and a node i
- α_{ij} the number of messages exchanged between a node i and a node j

Proactive method

Proactive behavior consists of the transmission of rules by the controller until the router receives the associated packages. Several solutions have been developed based on the algorithm of back - pressure suitable for the data networks very sensitive to latency and especially the QoS [7].

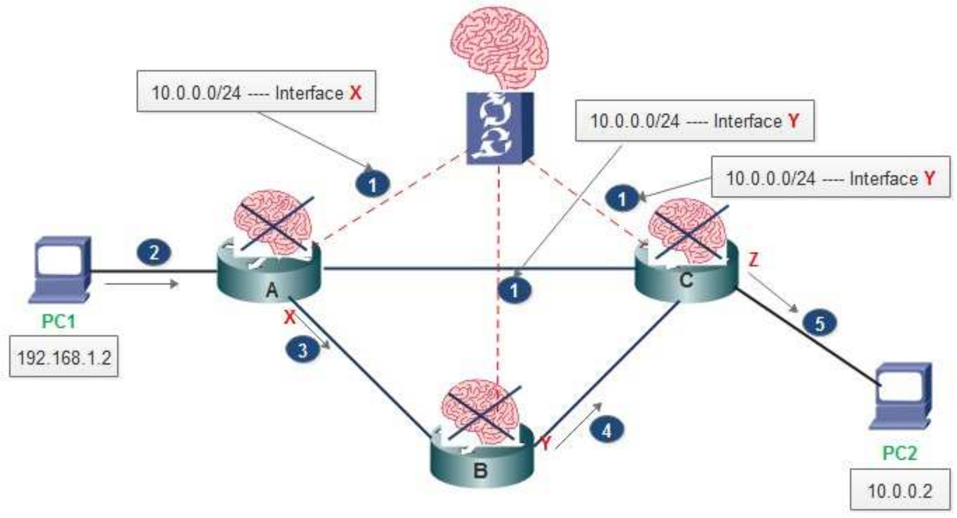
10.0.0.0/24 ---- Interface X
10.0.0.0/24 ---- Interface Y
10.0.0.0/24 ---- Interface Y
1
1
1
1
2
3
4
5
A
B
C
X
Y
Z
PC1
192.168.1.2
PC2
10.0.0.2

Figure 3: Descriptive diagram proactive approach

Figure 3 representing a network architecture to proactive behavior. When the node (A) of the figure 3 receives a packet from PC1 to PC2 destination (1), he runs the appropriate for this type of package and this transfer rule in its table. Operation (1) is thus repeated until the package arrives at its destination. When checking the rules of transfer from the source to the destination:

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

$$T(s,d) = \sum_{j=1}^{n} t_j \tag{3}$$

- Messages sent on the path to source s and destination d are defined by :

$$\alpha_{ch(s,d)} = \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_{ij} \tag{4}$$

where $\alpha_{ij} = \{^{ksilelien(i,j) \in ch_{(s,d)}}_{0,sinon}$

Experimental evaluation of the proactive and reactive approaches

In order to value these different approaches, we emulated the topology (figure 4) WAN, on a physical machine. The latter is characterized by an Intel Core i5 processor 4 cores running at 2.53 Ghz and an equal to 8 GB RAM. The machine that simulates the virtual network uses Mininet in its version 3.2, which allows to create the topology considered SDN. We will evaluate on the architecture figure 4, the impact that can have these two approaches on latency. We ignore possible failures in the network.

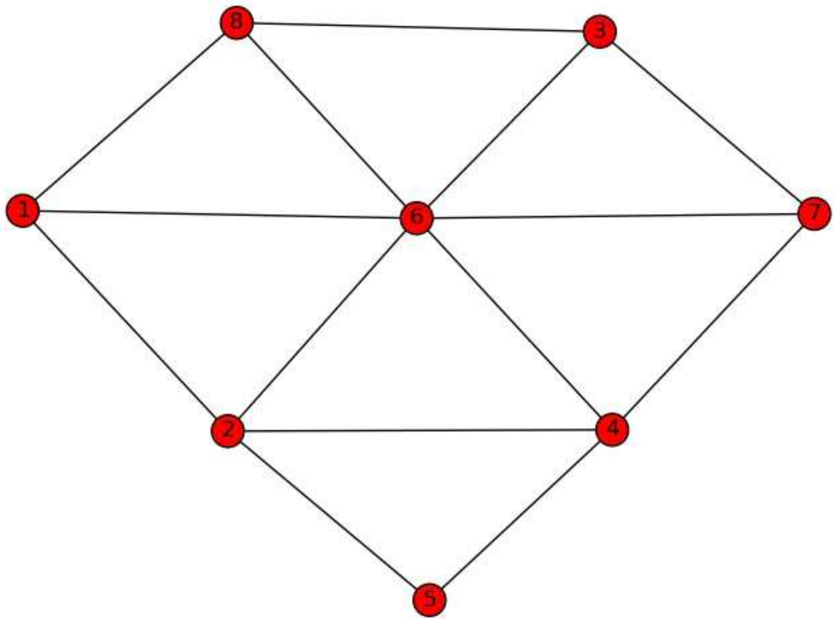

Figure 4: Topology of simulation

Our tests are carried out with the controller Ryu under its 3.2 version. It is based on python and provided APIs to communicate between applications and the network infrastructure. In addition, the protocol used for communication between the controller and the switches is OpenFlow1.3, as well as the Python3 programming language. In addition, we use Ping and Cbench tools to measure network performance metrics.
The simulation followed an iterative process and the simulation process is repeated ten (10) times. Thus we have the results of figure 5.

We see that the result of the proactive approach remains very lower than the reactive approach to each iteration. We note as well that the proactive approach allows to optimize the latency

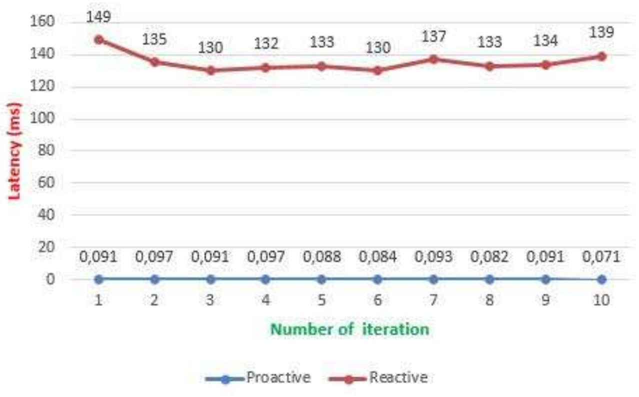

Figure 5: Latency analysis

compared to the reactive approach assuming that no failure occurs in the network. We suppose issues occurred at the level of some network nodes (the nodes have restarted) and we test the scalability of the proactive approach.

Assessment of the latency of the proactive approach with failure

When a node restart or when a break of link on a path, we see that it takes a time μ which varies according to the time of retransmission of the transfer rules of each node, which restarts on the path between the source s and destination d. We suppose that μ is the time-out period new rules of transfer on the part of the controller. Suppose that the controller sends the transfer rules to the nodes each q seconds. So when he is a loss on a transmission path, it becomes obvious that the timeout is less than or equal to the time taken by the controller before sending the new transfer rules. We have :

$$\mu = q - t_e \tag{5}$$

Where t_e is the elapsed time since the controller has transferred the transfer rules. It is obvious that $t_e \leq q$ then $\mu = q \leq t_e < q$.
Suppose a node j decides to contact the controller for new transfer rules. 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 time-out for the node j contact the controller c, t_c is the time set by the controller to process the request of the node and t_{cj} the time taken to transmit the transfer rule to the node. Still using the Simulator Mininet, Ryu under OpenFlow1.3, as well as programming Python3 language controller. We have obtained the results in figure 6 and figure 7.

These results show that when there is a failure in the network, the time the latency of the proactive method increases considerably (figures 6 and 7). Because it is not resilient to failures.

If $\mu > t_p$ in the strictly proactive case, one is obliged to wait for a time μ. However in this case, it is interesting to ask the transfer rules in order to receive them within t_p (more quickly). Where hybrid our approach, that we present in the following section.

4 Proposed approach

To realize the potential of programmable networks, we propose in this paper another variant of the algorithm of back - pressure with League of Nations based on a mixed method. Taking into account the set of messages exchanged between the source and destination during the transfer of

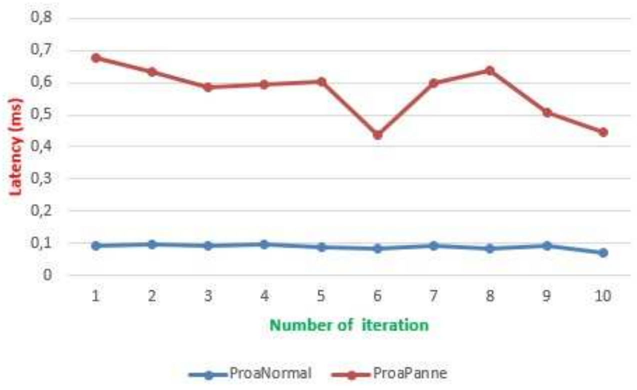

Figure 6: Latency with failure analysis

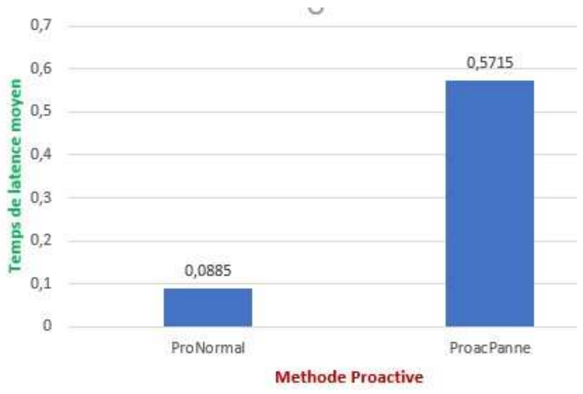

Figure 7: Average latency time with failure analysis

the stream for a better evaluation of the period. This in order to optimize the routing functions by simply placing the traffic here where capacity allows, to avoid congestion in some parts of the network strongly requested and also control failures in the network. The operation of the joint method is shown in figure 8.

4.1 Analysis of approach

When a node receives a packet for a given destination, it checks if there is a transfer for this destination rule since his transfer table and runs it so yes (node 8). Otherwise a message (PACKET_IN) (node 2) is sent to the controller and the controller determines the path satisfiable, then sends the transfer rules (PACKET_OUT) to all nodes in the path selected to avoid situations where the transfer rules are obsolete. If a node on a path restarts, and he contacts the controller as node 2 of the descriptive scheme, it will take time then the set R the path nodes that restart will use a time defined by :

$$\sum_{j \in R} t_{jc} + t_{cj} + t_c \tag{7}$$

To get the new transfer rules. If a set node restart but do not contact the controller, they will use a time defined by:

$$\sum_{j \in R} \mu_j \tag{8}$$

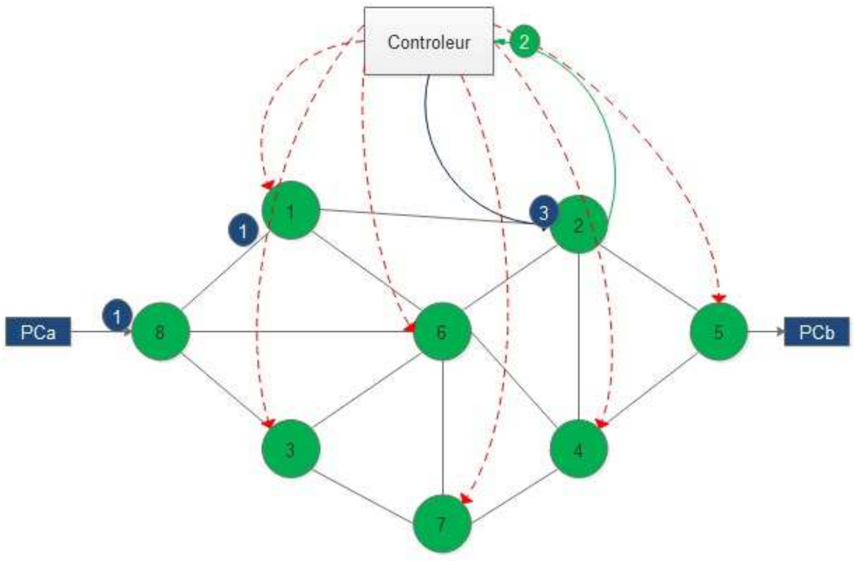

Figure 8: Descriptive diagram approach joint

to get the transfer rules. In these analyses, the transmission time of the information on a path to a source s to destination d is defined by :

$$T(ch(s,d)) = \sum_{j \in R}(t_{jc} + t_{cj} + t_c) + \sum_{j \in R} \mu_j + \sum_{j \in ch(s,d)} (t_j + t'_j) \tag{9}$$

where t_j is the transfer time of the node j, t'_j is the waiting time in the queue of the node j and μ_j is the timeout of the new rules of transfer of the node j with :

$$R = \{j \in ch(s,d) : \mu_j > t_{jc} + t_c + t_{cj}\} \tag{10}$$

$$R' = \{j \in ch(s,d) : \mu_j < t_{jc} + t_c + t_{cj}\} \tag{11}$$

The number of messages in the network nodes in the path for the transmission of information between the source to the destination is defined by :

$$\alpha_{ch(s,d)} = \sum_{i=1}(\alpha_{ic} + \alpha_{ci}) + \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_{ij} \tag{12}$$

$\alpha_{ic} = \{^{1 if the node i \in ch_{(s,d)}}_{0, else}$
$\alpha_{ci} = \{^{1 if the node i \in ch_{(s,d)}}_{0, else}$
$\alpha_{ij} = \{^{k if the link (i,j) \in ch_{(s,d)}}_{0, else}$
k with all the messages destined for the node j

4.2 Proposed algorithm

The algorithm that we offer can be described in the following :
We repeat the actions 1 to 3 to the destination.

Algorithm 1 : ReaPro-CM

Input: Network topology
Output: Transfer rules

```
Begin
  Step 1 : Processing of a node that receives a data transfer to the destination
  If a packet arrives at a node $j$ then
      If $j$ is not the recipient node then
          If $j$ has a transfer rule so
              j applies this rule to transfer
            Else
              If $\mu > t_p$ then
                  j contact the controller. Go to step 2.
                  If j is waiting for the new transfer rule
              Endif
          Endif
        Else
          $j$ receive the packet
      Endif
  Endif
  Step 2 : Controller Processing
  Determine the assessments of the different links on the network
  Determine the optimal paths using the dijsktra algorithm of the $k$ shortest paths at time $t$
  Communicate the transfer rule to the node that initiated the request as well as the nodes
  as a part of the path selected to reduce the tables of transfers of other nodes
  Go to step 3
  Step 3 : Processing of a node that receives a transfer rule
  The node running the transfer rule and send the package to the next to the destination node
End
```

5 Simulation and results

In order to evaluate our algorithm, we emulated the topology (figure 9) WAN, on a physical machine. The latter is characterized by an Intel Core i5 processor 4 cores running at 2.53 Ghz and an equal to 8 GB RAM. The machine that simulates the virtual network uses Mininet in version 2, a widely used tool in the SDN community and it can simulate topologies classic like bus, ring, hierarchical, or customized, that reflect the exact configuration a topology of company. In order to isolate the operating system experiences, we simulate virtual machines (VMs) through a named VirtualBox virtualization application. Reserve at least two VMs, one for the emulated network and one for the controller. The performance of the proposed algorithm is presented in figure 8.

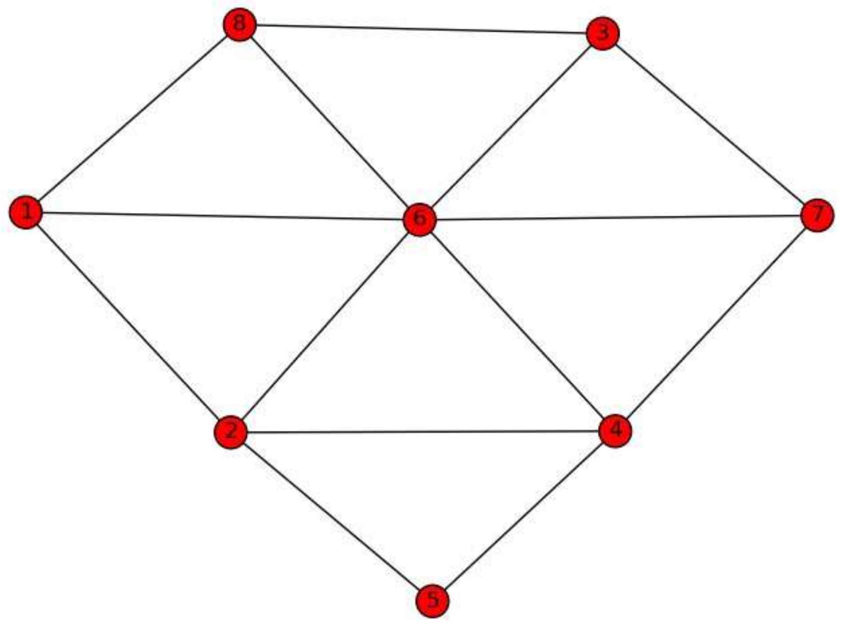

Figure 9: Topology of simulation

The simulation followed an iterative process. The simulation process is repeated ten (10) times and the results are presented in figures 10 and 11. Figure 8 shows the latency time of the routing Back-pressure trouble free approach is lower compared to the routing back - pressure with failure. Indeed, a failure of a node or link requires wait for a node controller of new transfer rules. However, the controller sends the transfer rules according to a period. As a result, a node that is a failure of a link or falling down, must wait for a period of time less than or equal to the delay of the controller transfer rules.

To minimize the latency time found with the results of figure 5, we proposed an alternative approach that improves significantly the lag time when a failure occurs in the network as shown in figure 5. The simulation results show that our approach (ReaPro-CM) latency time, remains much lower than that of Back-pressure with failure (Voting-BP with breakdown). Indeed, when a failure occurs in the network (link broken) during transmission of the stream, our approach provides a backup path allowing to move this flow without however wait until the controller for

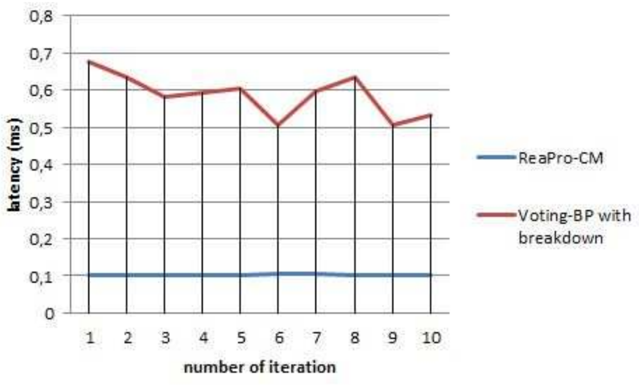

Figure 10: Latency with failure and rescue way

new transfer rules. When a failure occurs at the level of a node (a node reboot), our approach evaluates the time remaining until the node receives new transfer rules. If this time is very high, the node asks the controller new rules of transfer immediately. These principles implemented in our approach to routing can significantly reduce latency when there is a failure in the network (figure 11).

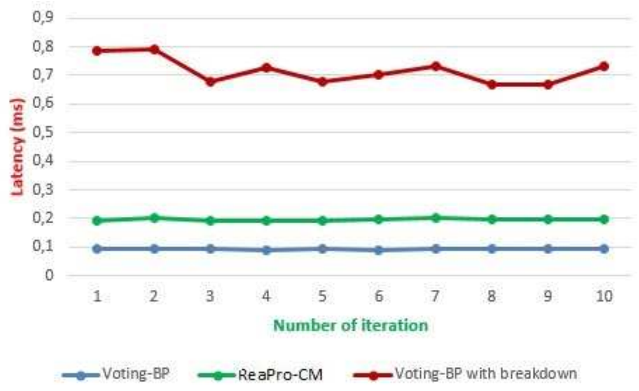

Figure 11: Comparison approach mixed with failure and proactive approaches

6 Conclusion

This article offers an approach of mixed multi-path routing in networks of data with SDN to resolve failures coming into the network. These failures have a significant impact on latency where interest to propose a joint routing based on the proactive and reactive approaches. Mathematical analysis and simulation results show both the algorithm that we offer allows to optimize the transmission time in networks of data with League if all the messages exchanged during the transfer of information are taken into account. Because the main problem in networks with SDN SDN routing tables cannot contain only a very limited number of rules.

Bibliography

[1] Bui, L.; Srikant, R.; Stolyar, A. (2009); Novel architectures and algorithms for delay reduction in back-pressure scheduling and routing, *INFOCOM 2009, IEEE*, 2936–2940, 2009.

[2] Erickson, D. (2013); The beacon openflow controller, *Proc. of the second ACM SIGCOMM workshop on Hot topics in software defined networking*, 13–18, 2013.

[3] Gojmerac, I.; Reichl, P.; Jansen, L. (2008); Towards low-complexity internet traffic engineering: the adaptive multi-path algorithm, *Computer Networks*, 52(15), 2894–2907, 2008.

[4] Goransson, P.; Chuck B. (2014); *Software Defined Networks: A Comprehensive Approach*, British Labrary Cataloguing-in-Publication Data, 2014.

[5] Jacobson, V.; Smetters, D.K.; Thornton, J.D.; Plass, M.F; Briggs, N.H.; Braynard, R.L.(2009); Networking named content, *Proc. of the 5th Intl. Conf. on Emerging networking experiments and technologies*, 1–12, 2009.

[6] Kulkarni, S.S.; Badarla, V.(2014); On multipath routing algorithm for software defined networks, *Advanced Networks and Telecommuncations Systems (ANTS), 2014 IEEE Intl. Conf. on*, 1–6, 2014.

[7] Luca, R.L.; Ciotirnae, P.; Popescu, F. (2016); Influence of the QoS Measures for VoIP Traffic in a Congested Network, *Intl. J. of Computers Communications* & *Control*, 11(3), 405-413, 2016.

[8] Maglaras, L.A.; Katsaros, D. (2011); Layered backpressure scheduling for delay reduction in ad hoc networks, *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2011 IEEE International Symposium on*, 1–9, 2011.

[9] Maldonado-Lopez, F.A. ; Calle, E.; Donoso, Y. (2016); Checking Multi-domain Policies in SDN, *Intl. J. of Computers Communications* & *Control*, 11(3), 428-440, 2016.

[10] Moeller, S.; Sridharan, A.; Krishnamachari, B.; Gnawali, O.(2010); Routing without routes: The backpressure collection protocol, *Proc. of the 9th ACM/IEEE Intl. Conf. on Information Processing in Sensor Networks*, 279–290, 2010.

[11] Nithin, M.; Ao, T.; Dahai, X. (2013); Optimal link-state hop-by-hop routing, *Network Protocols (ICNP), 2013 21st IEEE Intl. Conf. on*, 1–10, 2013.

[12] Pinheiro, B.; Cerqueira, E.; Abelem, A. (2016); NVP: A Network Virtualization Proxy for Software Defined Networking, *Intl. J. of Computers Communications* & *Control*, 11(5), 697-708, 2016.

[13] Tassiulas, L.; Ephremides, A. (1992); Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks, *IEEE transactions on automatic control*, 37(12), 1936–1948, 1992.

[14] Tootoonchian, A.; Gorbunov, S.; Ganjali, Y. (2015); On Controller Performance in Software-Defined Networks, *IEEE/Mediterranean Electrotechnical Conference*, 1–6, 2015.

[15] Ying, L.; Shakkottai, S.; Reddy, A.; Liu, S. (2011); On combining shortest-path and backpressure routing over multihop wireless networks, *IEEE/ACM Transactions on Networking (TON)*, 19(3), 841–854, 2011.

[16] Foodlight [Online]. http://openflowhub.org, Accessed: March 10, 2016.

[17] OpenFlow Switch Specification [Online]. https://www.opennetworking.org/software-defined-standards/specifications/, Accessed: June 25, 2016.