Let's Adapt to Network Change: Towards Energy Saving with Rate Adaptation in SDN

Samy Zemmouri, Shahin Vakilinia, Mohamed Cheriet
École de Technologie Supérieure (ÉTS)
Synchromedia Lab
Email: {samy.zemmouri, Shahin Vakilinia, Mohamed.Cheriet}@synchromedia.ca

***Abstract*—The exponential growth of network users and their communication demands have led to a tangible increment of energy consumption in network infrastructures. A new networking paradigm called Software-Defined Networking (SDN) recently emerged which simplifies network management by offering programmability of network devices. SDN assists to lower link data rates via rate-adaptation technique which reduces power consumption of the network. The main idea behind this paper is to find a distribution of traffic flows over pre-calculated paths which allow adapting the transmission rate of maximum links into lower states. We first formulate the problem as a Mixed Integer Linear Program (MILP) problem. We then present four different computationally efficient algorithms namely greedy first fit, greedy best fit, greedy worst fit and a meta-heuristic genetic algorithm to solve the problem for a realistic network topology. Simulation results show that the genetic algorithm consistently outperforms the three greedy algorithms.**

I. Introduction

Information and Communication Technologies (ICT) is becoming a key contributor to global warming and environmental pollution emissions [1], [2]. Consequently, green networking and energy saving mechanisms attract a growing attention from the networking industry and research community. In this context, many studies are unanimous in indicating that hardware technologies are expected to improve energy efficiency over next decade.

The main problem tackled in this paper is the reduction of energy consumption in the network using SDN capabilities. Indeed, the facility brought by SDN technology to control network devices and adapt their transmission rate to the current traffic load motivated us to propose a new power management strategy. The main contributions of this paper are summarized as follows: 1) Power consumption at network level is optimized by taking advantage of SDN capabilities, 2) Rerouting of flows in an unsplittable way through several preestablished paths is considered with adjusting the rate of links in an SDN-enabled infrastructure, 3) Flow allocation problem with an energy minimization objective is formulated as an MILP problem. 4) Extensive simulations are carried out to assess the performance of the proposed algorithms. It is shown that genetic algorithm can save up to 60% of energy consumption by incorporating rate adaptation into network devices. Notwithstanding that researchers have proposed many approaches for power saving in traditional networks; there has been little effort on reducing the energy consumption in SDN-enabled network infrastructures. Generally, for both legacy and SDN networks, there are two well-known strategies targeting network energy efficiency namely switch-off and rate adaptation [5]. The First approach attempts to reduce the set of active network resources by redistributing the traffic into a sub-network in a power-aware manner so that the rest of the network sleep and consequently energy is saved [8]–[16]. The latter one aims to achieve proportional energy consumption by adapting links data rates. It is shown in [5] that power consumption can be reduced considerably by lowering link data rates. However, the proposed algorithm in [5] is based on the switch-off strategy. Several problems may arise when completely turning off routers and switches such as route oscillation and tardy network convergence. During the convergence phase, the network can experience performance degradation due to temporarily unfeasible paths, that is, loops. Moreover, time to switch *off/on* device can increase waiting time for optimizing all network flows and may lead to reducing the responsiveness of new flows. To avoid these problems, we choose a different methodology and opt for the so-called rate-adaptation strategy. The idea of adapting link data rate to the traffic load is not new and has been investigated formerly [5]. [6] addressed the splittable flow allocation problem for both single and multiple communication sessions to optimize power saving. However, splitting streams are typically undesirable due to adverse reordering effects in the transport layer. In this paper, we exploit the rate-adaptation strategy similar to [5] and [6]; However, we consider an unsplittable flow allocation problem to minimize the energy consumption.

The rest of the paper is organized as follows; We briefly describe the problem and formulate it in Section II. The proposed algorithms are described in Section III. We evaluate and discuss the performance of the proposed approaches in Section IV and finally, we conclude the paper in Section V.

II. Problem Description And Formulation

In the following we consider SDN networks where the controller can collect information about flows carried by each path between each Origin and Destination nodes (O-D). Monitored traffic load on each link is the input of flow allocation algorithm which aims to optimize energy consumption of the network. Based on the result of proposed flow allocation algorithm, the controller redistributes flows among such given multipath [7] and re-adapt links rate to their new traffic load while preserving connectivity and

978-3-901882-85-2

satisfying links capacity constraints. Note that the main target is not to propose a new routing algorithm but to bring forth a complementary flow allocation algorithm which can be handled by an SDN controller and can operate jointly with routing protocols. Practically, SDN controller can compute any standard routing protocol to find a set of candidate paths between each (O-D) and thereupon the proposed flow allocation algorithms can be executed to distribute traffic load over these paths.

MILP formulation: In order to present Mixed Integer Linear Program (MILP) formulation for the unsplittable flow allocation with rate adaptation problem, we define $\mathcal{L}$ as the set of edge pairs (physical links) in the network. A path in the network is the sequence of links. $\mathcal{P}$ denotes the set of paths in the network. For every pair of nodes in the network, say node a, b , we consider different number of pre-calculated paths from node a to b, $\{p_{ab}(1), p_{ab}(2)..p_{ab}(\mathcal{N})\}$ where $p_{ab(n)} = p_{ba(n)}$. We also assume that there are always some existing flows between all source and destination nodes. $\mathcal{K}$ commodities are assumed in the network where for commodity k, d_k denotes its demand traffic. Given a set of $\mathcal{P}_k$ candidate paths for k^{th} traffic demand where $\{p_k \in \mathcal{P}_k\}$ denotes an appropriate path, We seek f_{p_k}, the amount of traffic routed through each given path p_k. This traffic setting should be done in a manner that total power consumption of network is minimized, and traffic demand of all commodities are satisfied. We consider the power consumption of link ℓ as discrete step increasing function $\mathcal{W}(f_\ell)$ of its traffic load f_ℓ. Link ℓ can work in one of $\mathcal{S}_\ell$ states . Each link at state s has a fixed capacity c_s with corresponding fixed power consumption w_s such that $w_{s-1} < w_s \Leftrightarrow c_{s-1} < c_s$. So, energy consumption of link ℓ, represented by $\mathcal{W}(f_\ell)$ can be obtained by,

$$\mathcal{W}(f_\ell) = \begin{cases} \omega_1 & if \;\; 0 < f_\ell < c_1 \\ \omega_2 & c_1 < f_\ell < c_2 \\ \cdots & \\ \omega_s & c_{s-1} < f_\ell < c_s \\ \cdots & \\ \omega_{\mathcal{S}_\ell} & c_{s_\ell - 1} < f_\ell < c_{s_\ell} \end{cases} \quad (1)$$

Equation 1 is not a linear conversion and can not be modeled with MILP. So, it is required to add extra complementary binary variables namely y_ℓ^s such that,

$$y_\ell^s = \begin{cases} 1 & \textit{if link } \ell \textit{ works in state } s \\ 0 & otherwise \end{cases} \quad (2)$$

Then, the total energy consumption of the network is equal to: $\sum_{\forall \ell \in \mathcal{L}} \sum_{\forall s \in \mathcal{S}_\ell} \omega_s y_\ell^s$. Moreover, to make sure that the path of the traffic demands are unsplittable, other complementary binary variables namely x_{p_k}s should be defined as,

$$x_{p_k} = \begin{cases} 1 & if \;\; f_{p_k} > 0 \\ 0 & f_{p_k} = 0 \end{cases} \quad (3)$$

Then, the optimization problem is given by,

$$\min \sum_{\forall \ell \in \mathcal{L}} \sum_{\forall s \in \mathcal{S}_\ell} \omega_s y_\ell^s$$

$$\text{subject to} \quad \frac{f_\ell - c_{s-1}}{c_{s_\ell}} \leq y_\ell^s \leq 1 + \frac{c_s - f_\ell}{c_{s_\ell}} \quad (C.1)$$

$$\sum_{\forall p \in \mathcal{P}_k} f_{p_k} = d_k \qquad (C.2)$$

$$f_{p_k} - x_{p_k} \geq 0 \qquad (C.3); \;\; \theta x_{p_k} - f_{p_k} \geq 0 \qquad (C.4)$$

$$\sum_{\forall p \in \mathcal{P}_k} x_{p_k} = 1 \qquad (C.5)$$

$$f_\ell = \sum_{k=1}^{K} \sum_{\forall p, \ell \in \mathcal{P}_k} f_{p_k} \quad (C.6); \;\; f_\ell \leq c_{\mathcal{S}_\ell} \quad (C.7)$$

In the above optimization, objective is the sum of energy consumption of all network links. (C.1) which stand $\forall s \in \mathcal{S}_\ell, \ell \in \mathcal{L}$, is applied to extract the complementary binary variables y_e^ss out of the traffic rates variables of ℓs represented by $f_\ell s$ in a linear format. Group (C.2) Constraints, which stands $\forall k \in \mathcal{K}$, ensure the traffic demands are satisfied. (C.3) and (C.4), which stand $\forall p \in \mathcal{P}_k, \forall k \in \mathcal{K}$, are applied to extract the complementary binary variables x_{p_k}s out of f_{p_k}s (traffic rate of commodity k in path p variables). In fact, they are applied to linearize Eq. 3. Constraint group (C.5), which stand $\forall p \in \mathcal{P}_k, \forall k \in \mathcal{K}$, ensures that the path attributed to k^{th} demand traffic is unsplittable. Note that θ is assumed as an arbitrary integer much larger than the capacity of links. Group (C.6) Constraints evoked f_ℓs variables out of f_{p_k}s. Finally, group (C.8) ensures the capacity constraint of links are satisfied. (C.6) and (C.7) group Constraints stand $\forall \ell \in \mathcal{L}$. The aforementioned problem is NP-hard and can not be solved in a short period. Hence, in next Section, few low complexity algorithms are proposed to solve this optimization problem.

III. Proposed Algorithms

Finding flow routes in a general network while not exceeding the capacity constraint of any links is called multi-commodity flow problem, which is known to be NP-Hard for integer or unsplittable flows [17]. In this approach, demand is represented by a set of flows so that each flow supports a fixed sub-demand and have to be routed over one of the pre-calculated paths which consequently is an NP-Hard problem as well [18]. Thus, due to the large scale of the network and time constraints for SDN controller to react instantly to the traffic changes and reroute the flows, solving the exact optimization problem is not applicable. Therefore, given a current network flow configuration, following four computationally efficient algorithms are suggested to address the problem above in large scale SDN-enabled networks; Greedy : First fit, Best fit, Worst fit algorithms and Meta-heuristic: GA

A. Greedy Algorithms: The proposed three greedy algorithm are described in the following pseudo-code:

1: List nodePairs; //list of all (O-D)
2: List numFlowPerPairs; //Number of flows between (O-D)
3: List S= sortNodePairs(nodePairs,numFlowPerPairs);

```
4:    For n in S
5:       List X=kPaths(n); //return list of pre-calculated paths
6:       List C=pathsCost(X);
7:       List Y=sortPaths(X,C);
8:       For i in Y
9:          List Z=flowsInPath(i); //get all flows from path i
10:         List W=sortFlows(Z);
11:         For j in W
12:             greedyStrategy(strategy,j);
13:         end for
14:      end for
15:   end for
```

Explanation : **Step(3)**: Sort (O-D)s in descending order regarding their number of flows. **Step(6)**: $pathsCost()$; Sort flows in descending order regarding their current load. **Step(7)**: $sortPaths()$; Sort (O-D)s in descending order using parameter C which is computed at **step(6)**. **Step(10):** Compute the cost for all the path in X by summing of states of all links in the path divided by the number of links of the path. **Step(12)**: If variable $firstfit$ strategy, we try to move the flows one by one in the first path of the sorted list Y, which satisfy the capacity and cost constraints(energy consumption) and update the path cost of both old and new path. In $Bestfit$ case we try to move one by one the flows in the path of the sorted list Y, which provide the best costs and satisfy capacity constraint. Finally, in $Worstfit$ case, we try to move the flows one by one in the path of the sorted list Y, which provides the worst energy cost compared to others paths in Y and satisfies the capacity and cost constraints.

B. Metaheuristic Algorithm: The fourth proposed strategy is based on GA [20] which is a suitable method to work the optimization problems out. GA algorithm uses three operators on its population [21], which described in the following pseudo-code of the proposed adapted GA:

```
1: List nodePairs;
2: List numFlowPerPaire;
3: Double porbMutation=0.02;
4: List S= sortNodePairs (nodePairs,numFlowPerPaire);
5:    For n in S
6:       List X=kPaths(n);
7:       List Y=kFlows(n);
8:       List init=initPopulation (X,Y);
9:       List Selection=fitnessFunct(init);
10:      Integer numCrossover= numFlow(Selection)*10;
11:      Integer i=1;
12:      while(i< numCrossover)
13:         Integer path1=random(0,X.size());
14:         Integer path2=random(0,X.size()); //path1!= path2
15:         Integer pos1=random(0,nFlowperPath(path1).size());
16:         Integer pos2=random(0,nFlowperPath(path2).size());
17:         List crossover=Crossover(path1,path2,pos1,pos1);
18:         Selection.add(crossover);
19:         if (rand>probMutation)
20:            Double rand=random(0,1);
21:            Integer path3=random(0,X.size());
22:            Integer path4=random(0,X.size());
23:            List mutation=Mutation(path3,path4,rand);
24:            Selection.add(mutation);
25:         end if
26:         i=i+1;
27:      end while
28:      Selection=fitnessFunct(Selection);
29:   end for
```

Description : **Step(8)**: Population is initiated, each path receive the flows of others paths. Then the new path cost for each new path added to the Initial Population is calculated. Variable $init$ contain the pre-calculated paths and their respective flows for each (O-D). In addition, the flows of the others paths are assigned to each path and the cost of each new path are recalculated before adding it to $init$ list. Thus $init.size() = X.size()^2$. **Step(9,28)**: We assess for each new path if it satisfies the capacity constraint. If not, we remove the path from the initial Population. We also remove the path with higher cost. **Step(9,28)**: selection function Contains the new population(paths with higher cost and path in $init$ not satisfying capacity constraint for each of their links are removed. **Step(17)**: For each two selected path, we perform crossover between their flows at the selected position. Then we add the newly generated path to the population ($Selection()$). **Step(17)**: For each two selected path, we move the flow selected randomly from one path to the other. We assess the satisfaction of the capacity constraint.

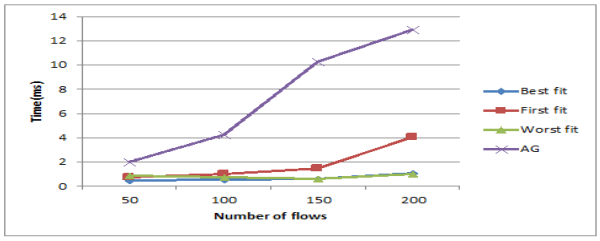

Fig. 1. Computation time of First fit, Best fit, Worst fit and GA (mutation prob = 0.02 , number of crossover = 10 *number of flows).

IV. Tests and Simulation Results

In this Section, several tests are conducted to evaluate the proposed algorithms in different scenarios: *Test 1:* The first test aims to compare our four proposed algorithms between each other. For this purpose, we limited our first test to a pair of nodes. We generate 10 paths with a random length between this pair of nodes (O-D). Next, we simulate a different number of flows equally distributed over the paths. In the case of GA, the test is repeated for various Crossover values respectively {1,3,6,10} *some flows with fixed mutation probability=0.02. The probability of crossover is fixed to 1. Tab.1 gives the corresponding capacity thresholds and power consumption value in [6]. We repeated the simulation 10000 times, and we collect the mean and the maximum energy saving percentage

at the end. In Fig.1, it can be seen the difference in term of computation time between the four strategies. We notice in this figure that all proposed algorithms provide relatively small computation time in order of (ms). Analyses of Tab.I and Tab.II shows that the GA outperform in term of energy saving percentage which is around 60% for mean value in 10000 iterations and around 70% for the maximum saving rate in the same number of iterations. By increasing the number of generated flows, GA keeps saving the significant amount of power compared to other three greedy algorithms. Particularly, for the case where some crossovers are equal to 10×number of flows (between the pairs of nodes). It is shown in this test that GA provide better performance in polynomial time compared to the others greedy algorithms in term of energy saving. Thus the following criteria will be based only on GA.*Test 2:* to assess the performance of GA in realistic topology, we consider two well-known network topologies: NSF network with 14 nodes and 20 links and the Abilene network with 10 nodes and 13 links. For both topologies, we generate 2 disjoints paths between each pair of nodes. Next, 11 pairs of nodes are chosen randomly, and 10 flows are generated over their paths. Each path has a random load value between [0,50] Mbps. *Test 3:* In this test, simulation is conducted similar to test 2 but instead of generating 2 Disjoints paths, 2 Shortest paths are generated and computed by Yen's algorithm that we implemented between each pair of nodes. Each path carries 10 flows with traffic load value randomly generated between [0,100] Mbps. *Test 4:* In this test, 5 shortest paths are generated. Each path carries 15 flows with traffic load value randomly generated between [0,10] Mbps.

TABLE I
MEAN , MAX ENERGY SAVING OF FIRST, BEST AND WORST FIT(%)

	First fit		Best fit		Worst fit	
nb of flows	mean	max	mean	max	mean	max
50	13.46	74.14	18.64	74.14	21.96	75.68
100	40.26	76.25	53.48	68.41	53.22	72.11
150	22.01	58.62	16.23	68.87	29.97	76.12
200	1.81	37.4	0.99	36.6	1.77	42.5

TABLE II
MEAN , MAX ENERGY SAVING(%) OF GA :MUTATION=0.02

	Genetic Algorithm							
	Crossover/mutation probability=0,02							
	1*nb flow/0,02		3*nb flow/0,02		6*nb flow/0,02		10*nb flow0,02	
nbr of flow	mean	max	mean	max	mean	max	mean	max
50	44.4	74.9	47.15	76	47.91	76.38	48.56	77
100	59.54	78.05	61.63	79.34	62.4	77.52	62.67	79.54
150	59.87	77.09	62.04	72.69	62.69	77.68	63.16	72.66
200	54.1	69.69	57.52	73.13	58.58	72.28	59.09	71.61

Result and discussion: Results for tests 2, 3 and 4 are shown in Tab.III. An interesting observation is that our proposed strategy based on GA performs better in the NSF network than in Abilene Network. It is depicted that our solution achieves average power saving {35.31%,21.72%,26.81%} for 3 scenarios respectively: 2 Disjoints paths, 2 Shortest paths, 5 Shortest paths. It also achieves maximum power saving {66.93%,43.51%,62.54%} for the 3 scenarios respectively: 2 Disjoints paths, 2 and 5 Shortest paths. On the other hand for the Abilene network, it gets average power saving {18.63%,20.87%,11.01%} for the same scenarios with maximum power saving estimated to {54.45%,50.61%,38.89%}. The power saving difference between NSF and Abilene networks is mainly because NSF network is denser with less shared links among the pairs. Regarding the computation time, it is noticed that computation time increases proportionally with the growth of preestablished paths between (O-D)s which can be justified by the fact that the proposed algorithm proceeds iteratively between the pairs of nodes.

Assessment : In term of energy saving, our work give a better energy saving percentage for the case of single commodity flow allocation compared to [6]. Indeed, it is shown in Table. 3 that we can save between 48% and 59% depending on the number of flows per demand. However, only 26% were saved for the best case in [6]. Besides, for a large number of flows this work outperform in time computation too, in comparison with 2 LP solver (LP GAMS, LP GLPK) [11] as it is show in Fig.2.

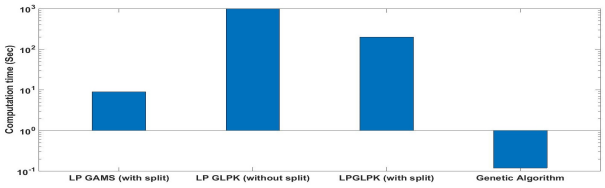

Fig. 2. Computation time for 800 flows.

TABLE III
MEAN , MAX ENERGY SAVING(%) AND COMPUTATION TIME (MS)

	GA		
	crossover= 10*nb flows / mutation prob= 0,02		
	mean (%)	max(%)	time (ms)
2-disj-paths-Abilene	18.63	54.45	4.5
2-disj-paths-NSF	35.31	66.93	3.8
2-sh-paths-Abilene	20.87	50.61	4.3
2-sh-paths-NSF	21.72	43.51	3.9
5-sh-paths-Abilene	11.01	38.89	10.3
5-sh-paths-NSF	26.81	62.54	12.8

V. CONCLUSION

In this paper, we propose an energy-efficient scalable mechanism by exploiting the SDN capability to provide the global view of the network and to adapt the links data rates regarding their carried traffic demand. To do so, an MILP formulation of the flow placement with energy minimization objective at the link level is provided. Next, fours algorithms are proposed and assessed to solve the problem. Simulations results indicate that GA based method outperforms the greedy algorithms. Indeed, it saves up to 63% between one O-D and between 11% to 35% on the entire network which carries more than 800 flows, while still preserving time complexity which does not exceed 13ms.

REFERENCES

[1] M.R.Celenlioglu, S.B. Goger and H.A. Mantar,"An SDN-based energy-aware routing model for intra-domain networks", in Proceeding of 22nd International IEEE Conference on Telecommunications and Computer Networks (SoftCOM), pp. 61-66, September 2014.

[2] L. Chiaraviglio, M. Mellia and F. Neri, "Minimizing ISP network energy cost: formulation and solutions", IEEE/ACM Transactions on Networking (TON), vol. 20, no. 2, pp.463-476, 2012.

[3] M. Zhang, C.Yi, B. Liu, and B. Zhang, "GreenTE: Power-aware traffic engineering", in Proceeding of 18th IEEE International Conference on Network Protocols (ICNP), pp. 21-30, 2010.

[4] P.Mahadevan, P.Sharma, S. Banerjee, and P. Ranganathan, "A power benchmarking framework for network devices",. In proceeding of Springer NETWORKING conference, pp. 795-808, 2009

[5] S.Nedevschi, L. Popa, G.Iannaccone, S. Ratnasamy and D. Wetherall, "Reducing Network Energy Consumption via Sleeping and Rate-Adaptation", In NSDI, vol. 8, pp. 323-336), 2008.

[6] J. Tang, B. Mumey, Y. Xing, and A.Johnson,"On exploiting flow allocation with rate adaptation for green networking", in Proceedings of IEEE INFOCOM, pp. 1683-1691, 2012.

[7] Kumar, Alok, Sushant Jain, Uday Naik, Anand Raghuraman, Nikhil Kasinadhuni, Enrique Cauich Zermeno, C. Stephen Gunn et al. "BwE: Flexible, hierarchical bandwidth allocation for WAN distributed computing." In ACM SIGCOMM Computer Communication Review, vol. 45, no. 4, pp. 1-14. ACM, 2015.

[8] M. Gupta, S. Singh, "Greening of the Internet", in Proceedings of ACM conference on Applications, technologies, architectures, and protocols for computer communications, pp. 19-26, 2003.

[9] L. Chiaraviglio, M. Mellia and F. Neri, "Reducing power consumption in backbone networks", in Proceeding of IEEE International Conference on Communications (ICC), pp. 1-6, 2009.

[10] Bianzino, A.P., Chiaraviglio, L. and Mellia, M., 2011, September. "Grida: A green distributed algorithm for backbone networks" i proceeding of IEEE Conference on Green Communications (GreenCom), pp. 113-119, 2011.

[11] B.Heller, S. Seetharaman, P.Mahadevan, Y.Yiakoumis, P.Sharma, S.Banerjee and N. McKeown, "Elastic Tree: Saving Energy in Data Center Networks", In NSDI, vol. 10, pp. 249-264, 2010.

[12] Sh,Vakilinia, D. Qiu, and MM. Ali. "Optimal multi-dimensional dynamic resource allocation in mobile cloud computing." EURASIP Journal on Wireless Communications and Networking,no. 1, pp.1-14, 2014.

[13] S. Vakilinia S, MM Ali, D. Qiu "Modeling of the resource allocation in cloud computing centers", Computer Networks. 14;91: pp. 453-70, 2015.

[14] S. Vakilinia, M Alvandi, M Khalili Shoja, and I Vakilinia. "Cross-Layered Secure and QoS Aware Design of VOIP over Wireless Ad-Hoc Networks." International Journal of Business Data Communications and Networking (IJBDCN) 9, no. 4, pp. 23-45, 2013.

[15] S. Vakilinia, M Alvandi, M Khalili Shoja, and I. Vakilinia. "Multi-path multi-channel protocol design for secure qos-aware VOIP in wireless ad-hoc networks." In IEEE Wireless and Mobile Networking Conference (WMNC), 6th Joint IFIP, pp. 1-6, 2013.

[16] X. Wang, , Y. Yao, K. Lu, and Q. Cao, , "Carpo: Correlation-aware power optimization in data center networks", in Proceedings of IEEE INFOCOM, pp. 1125-1133, 2012.

[17] S. Even, A. Itai and A. Shamir, "On the complexity of time table and multi-commodity flow problems", in Proceeding of IEEE 16th Annual Symposium on Foundations of Computer Science, pp. 184-193, 1975.

[18] S.Brandt, K.T. Foerster, and R. Wattenhofer, "On Consistent Migration of Flows in SDNs", in Proceeding of IEEE INFOCOM, pp. 41-50, 2016.

[19] CPLEX, IBM ILOG.,"V12. 1: Users manual for CPLEX.", International Business Machines Corporation, 2009.

[20] K. Deb, "Multi-objective genetic algorithms: Problem difficulties and construction of test problems", Evolutionary computation, vol. 7 no. 3, pp.205-230. 1999.

[21] C.Sharma, S. Sabharwal, and R. Sibal, "A survey on software testing techniques using genetic algorithm", International Journal of Computer Science Issues, Vol. 10, no 1, pp. 1141-1154, 2014.