OPENFLOW-BASED IMPLEMENTATION OF A GEARBOX-LIKE ROUTING ALGORITHM SELECTION IN RUNTIME

Mircea Valeriu ULINIC Andrei Bogdan RUS Virgil DOBROTA
Technical University of Cluj-Napoca, Communications Department, 400027 Cluj-Napoca, Romania,
Phone: +40-264-401226, Fax: +40-264-597083,
Emails: Mircea.Ulinic@yahoo.com, {Bogdan.Rus,Virgil.Dobrota}@com.utcluj.ro

Abstract:This paper aims to prove using an OpenFlow-based testbed that a real gearbox-like routing algorithm selection in runtime is feasible. The proper selection and interworking of flow forwarding schemes for Infrastructure and Service Providers may lead to enhanced performances. Limited resources and/or a partial topology are shared by the owner of the network with the service producer, based on a real-time evaluation of the status using the Floyd-Warshall algorithm for single path and the Ford-Fulkerson for multipath routing. At its level, the Service Provider decides the best forwarding scheme involving the Modified Dijkstra's algorithm and, whenever multipath is needed, the Ford-Fulkerson algorithm. However due to ownership, security issues, and the business models agreed nowadays, only the Infrastructure Providers are actually performing the routing according to own rules (for its users or in case of emergency) and on behalf of Service Providers, according to their rules. Other combinations of algorithms for the existing or new coming types of providers (e.g. for virtual service infrastructure) can be easily tested now in the proposed software-defined networking tool.

Keywords: *Floyd-Warshall algorithm, Ford-Fulkerson algorithm, gearbox-like routing algorithm selection, Modified Dijkstra's algorithm, OpenFlow*

I. INTRODUCTION

A lot of industry and research communities are fascinated nowadays by the huge potential of the software-defined networking and its applications for the Future Internet. OpenFlow-based technologies are a good example of how an innovative solution could be rapidly prototyped and deployed. This paper is focused on an adaptive routing solution in a testbed with Open vSwitches controlled by software. It allows to involve several routing algorithms and to properly switch from one to another, like in a gearbox.

Let us investigate first if similar approaches have been proposed. Thus in [1] an OpenFlow-based controller, called OpenQoS, takes dynamic routing decisions in a real network. Authors claim that a seamless video quality of service is provided based on a metric by minimizing the packets lost and the latency. The parameters were not measured, but calculated based on a function depending on the traffic. The delay represents the average of the observed values for a given protocol. The optimum path is chosen with LARAC (Lagrangian Relaxation Based Aggregated Cost), based on the Dijkstra's algorithm executed several times if the proposed solution is not satisfying the requirements. Anyway this approach is limited to video traffic only. In [3] regular measurements can be used for planning the resource allocation in clever manner. The feasibility and the benefits of OpenFlow-based implementation of routing and load balancing between data centers and thin client sites were demonstrated. Paper [4] proposes Q-Learning (Q-L) based opportunistic flow management, with a NOX controller. It is rather difficult to appreciate how deep will evolve this idea. A multi-site approach with SDN (Software-Defined Networking) used to automatically reconfigure the network in case of a one-zone failure is discussed in [5]. The tasks are taken over by the remaining functioning parts of the service provider's network. The logical topology is modified based on a day or night load.

Paper [6] covers the routing model in anycasting. The authors consider two routing strategies: based on the number of hops using the Floyd-Warshall algorithm, and based on the link load, measured as the number of packets sent in a time unit. They claim the second approach, based on load-aware anycasting is offering better performances (according to simulations in Mininet). Paper [7] encourages using several controllers in network (i.e. one for each eight OpenFlow-based switches). How scalable is this open-source technology is studied in [8]. The manufacturers (e.g. Cisco) are promoting commercial solutions where OpenFlow is just a particular case [9]. For instance Cisco ONE (Open Network Environment) is a more comprehensive tool, including controllers and agents, programmable interfaces, virtual overlays,

Manuscript received May 7, 2014;revised June 11, 2014

orchestration, automation, bidirectional interaction, real-time analytics etc.The infrastructureis based on Cisco ONE Platform Kit (onePK), Cisco Nexus 1000V Series Switches, Cisco ONE Controller, Cisco nLight technology multilayer control plane, Cisco Network Positioning System, Cisco Quantum Software Suite, as well as other network functions virtualizations.

In our paper we continue a theoretical idea of GRAS (Gearbox-like Routing Algorithms Selection) in runtime, proposed by us in [10] and partially implemented in [11]. We presume the same business model with Service Providers (SPs) and Infrastructure Providers (IPs). Summarizing their definitions, the first ones may not own the network, but they are in direct contact with the customers. The other ones hold the communications networks, but it is not mandatory to offer services to the end users.The IP applies single routing schemes based on the Floyd-Warshall algorithm and, whenever is needed, may switch to multipath approaches based on the Ford-Fulkerson algorithm. Partial up to the maximum resources and network topology configuration are communicated to each SP. The Service Provider involves a single path routing scheme based on the Modified-Dijkstra's algorithm, previously designed, simulated and evaluated in [12]. However the Ford-Fulkerson algorithm may be involved again (but at SP level) for multipath selections. Note that the real implementation using OpenFlow for all algorithms mentioned above, integrated within the GRAS solution, are our original contribution. We could accommodate also the concept of Virtual Service Infrastructure Providers (VSIPs), proposed in [2]. It means that virtualized infrastructures from several IPs are grouped in a virtual service infrastructure (called also fluid Internet). However this type of provider involves inter-domain routing schemes which are out of the scope of this work.

The rest of the paper is organized as follows. In the Section II a brief presentation of gearbox-like routing algorithm selection will be thoroughly described, while in Section III, we present the real implementation of GRAS using OpenFlow. The experimental results in Section IV are followed by the conclusions and future work.

II. BRIEF PRESENTATION OF THE GEARBOX-LIKE ROUTING ALGORITHMS SELECTION IN RUNTIME

We summarized herein the principles and the partial results previously obtained by us, discussed in details in [10],[11],[12].

Depending on the roles of the provider within a topology the recommended solutions are the following [10]:Floyd-Warshalland Modified Dijkstra's algorithms for single path routing and Ford-Fulkerson algorithm for multipath. Like in a gearbox (see *Figure 1*), we may switch from one algorithm to another.

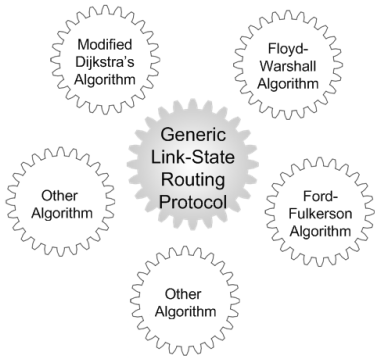


Figure 1. Adaptive routing system

*The Floyd-Warshall (FW) algorithm*aims to find the path with the lowest cost between any pair of nodes in the topology (i.e. peer-to-peer or wireless meshed networks). This is needed mainly by Infrastructure Providers.

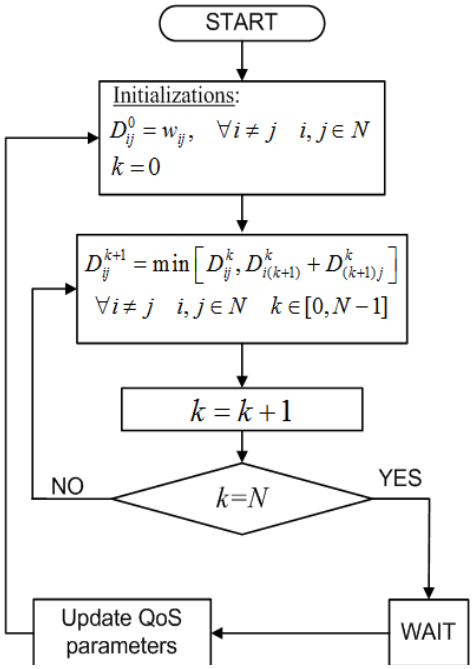


Figure2.The Floyd-Warshall algorithm [10]

In *Figure 2*the notation D_{ij}^krepresents the cost of the path between nodes *i* and *j* using a maximum of *k* intermediary nodes (i.e. nodes 1, 2,…k). *k=0*means that no intermediary nodes are involved (i.e. it is a direct link). The weight/cost of the arc between nodes *i* and *j* is

$$w_{ij} = \frac{F_{ij}}{C_{ij} - F_{ij}} + d_{ij} \cdot F_{ij} \qquad (1)$$

Equation (1) illustrates that in the implementation considered herein, the costs depends on the amount of flowF_{ij}, in bits per seconds, traversing the link *(i,j)*, the maximum theoretical transfer rate C_{ij} (i.e. capacity in bits per second) and the minimum latency d_{ij}, in seconds.The maximum number of iterations is the cardinality of set *N*.

*The Modified Dijkstra's (MD) algorithm*is used to find a path with the lowest cost to/from every other node in the topology if the source/destination node is fixed (e.g. sensor networks sending data to a single gateway or a provider offering a set of services to its subscribers). Mainly Service Providers could take benefits of this. For the Infrastructure Providers it is handier to calculate in a single iteration (using Floyd-Warshall) instead of repeated use of Modified Dijkstra's for each given source/destination.

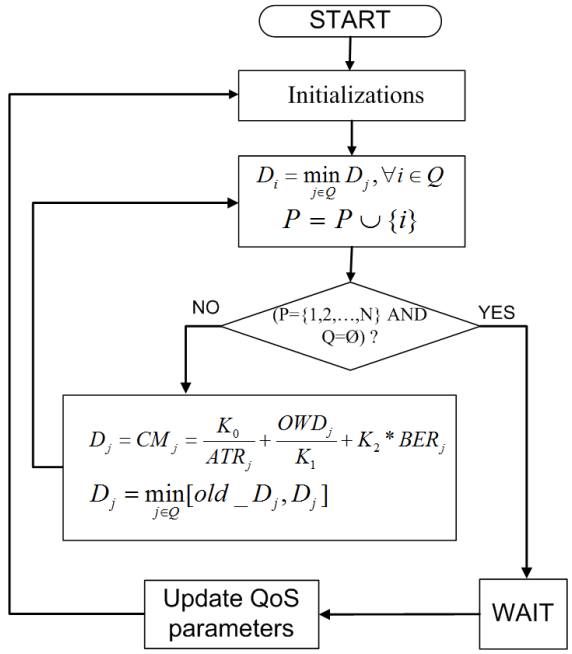


Figure3.The Modified Dijkstra's algorithm [10]

In *Figure 3P* is the set containing the nodes included in the spanning tree and *Q* is the set containing the rest of the nodes in the graph. Set *P={d}*, where *d* is the destination node and set *Q={1,2,…,N-1}*. Moreover, a formula that aggregates all the QoS parameters into a single one called Composite Metric CM_j will also be used. $CM_j=D_j$is the cost of the path between the node *j* and the destination. The AvailableTransfer RateATR_j,One-Way DelayOWD_j and Bit Error RateBER_j are calculated globally for all the concatenated links of a specific path having the starting point node *j* and the destination node *d*. However their composability is different compared to the classical approach.

$$ATR_j = \min(ATR_{ji}, ATR_{ik}, \dots ATR_{ld})$$
$$j \epsilon Q, i, k, l, d \epsilon P, i \neq k \neq \dots \neq l \neq d \quad (2)$$

$$OWD_j = OWD_{ji} + OWD_{ik} + \dots + OWD_{ld} \quad (3)$$

$$BER_j = 1 - P_j \quad (4)$$

$$P_j = (1 - BER_{ji})(1 - BER_{ik}) \dots (1 - BER_{ld}) \quad (5)$$

P_j is the probability of not having any erroneous bits on the entire path between nodes *j* and destination *d.* The cost D_j calculated for the entire path will aggregate into one composite metric CM_j all the traffic parameters.

$$D_j = CM_j = \frac{K_0}{ATR_j} + \frac{OWD_j}{K_1} + K_2 * BER_j \quad (6)$$

where: $K_0=10^9$[bps], $K_1=10^{-5}$[s] and $K_2=10^{12}$. The constants were chosen to have a minimum value of the metric (i.e. 1+1+1=3), for 1Gbps-link capacity, 10μs-OWD and 10^{-12} BER. Once the technologies evolve, the constants K_0, K_1 and K_2 could be updated. For instance if the 10 Gbps-link is the new reference, then $K_0=10^{10}$[bps]. Note that the Layer 1 technology, the type of media and the distance between neighboring nodes have a great impact that cannot be seized by ATR and/or OWD. This is the reason we introduced BER, but unfortunately itcannot be measured directly, being calculated or estimated (based on SNR values). In this paper we consider wired networks only and $K_2=0$.

TheFord-Fulkerson with Breadth-First Search (FF with BFS) algorithm finds the maximum transfer rate available between the given source and destination nodes.

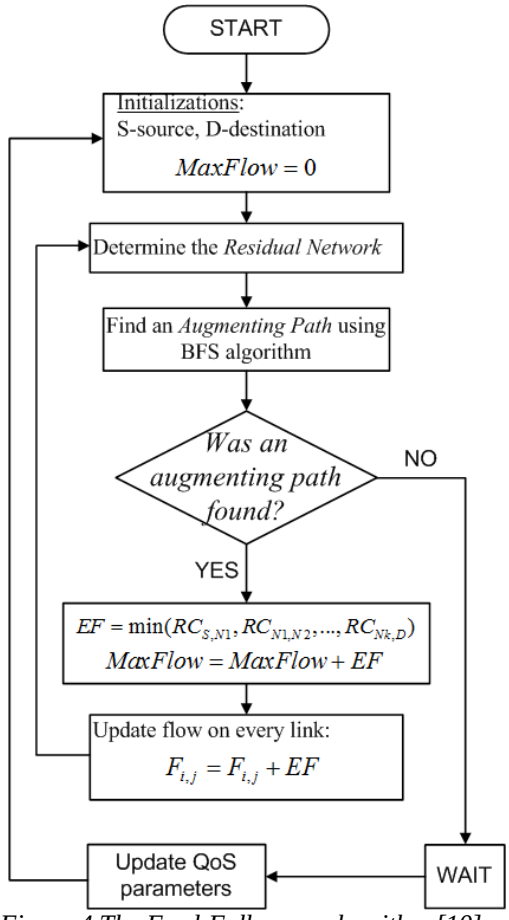


Figure 4.The Ford-Fulkerson algorithm [10]

We will use the acronym FF, but we suppose actually FF with BFS. The prior simulations in OMNET ++ proved that better performances for path selection are provided by the greedy algorithm instead of the round-robin one. In *Figure 4*Residual CapacityRC_{ij} of a link between nodes i and j is equal to the difference between the capacity $C_{ij}$of the corresponding link and the flow F_{ij}traversing it:

$$RC_{ij} = C_{ij} - F_{ij} \quad (7)$$

Residual Network is a network that has the same topology as the original one, but having the capacity of each link equal to its residual capacity. Augmenting Path is an alternating sequence of nodes (i.e. a path), having as starting point the designated source node S and ending with the destination node D. Moreover, no node is repeated and no link is saturated ($F_{ij} < C_{ij}$ or $RC_{ij} > 0$)for any i,j in set N. Excess Flow capacity of an augmenting path EF is equal to the minimum residual capacity of a link along an augmenting path found previously in the residual network, between the source and the destination:

$$EF = \min (RC_{S,N1}, RC_{N1,N2}, \dots, RC_{Nk,D} \quad (8)$$

The best way to summarize the gearbox-like routing algorithm selection in runtime is given by the following pseudo-codes:

```
procedureInfrastructureProvider(full_graph,
current_flow)
ifcurrent_flowϵ IP
then
best_path←FloydWarshall(full_graph)
ifbest_path.ATR<current_flow.transfer_rate
then
paths←FordFulkerson(full_graph,
current_flow.transfer_rate)
if size(paths) >1then
best_path←GreedyDistribution(paths,
current_flow)
endif
endif
else
partial_graph←FilterGraph(full_graph)
best_path←ServiceProvider(partial_graph,
current_flow)
endif
end procedure

procedureServiceProvider(partial_graph,
current_flow)
best_path←ModifiedDijkstra(partial_graph)
ifbest_path.ATR<current_flow.transfer_rate
then
paths←FordFulkerson(partial_graph,
current_flow.transfer_rate)
if size(paths) >1then
best_path←GreedyDistribution(paths,
current_flow)
endif
endif
end procedure
```

III. REAL IMPLEMENTATION USING OPENFLOW

Suppose a four-node testbed (SW1, SW2, SW3 and SW4) belonging to an Infrastructure Provider, with Open vSwitch1.3 installed on each PC under Linux. The traffic from IP and SP is generated by the SOURCE connected to SW1 as in *Figure 5*. Node DESTINATION connected to SW4 receives traffic from IP, whilst the virtual machine VM4 is needed to get the traffic sent by SP. All the nodes are directly connected through dedicated links to the centralized Beacon controller 1.0.2 (under Linux). Note that the Service Provider uses a light controller (written by us), needed to get through sockets the resources and the topology, as well as the routing requests. SP will send back its routing decisions to be applied by the Beacon controller in the real network. Note that IP could interwork with several SPs. Furthermore, the number of nodes could be easily expanded to a larger one due to the scalability of the OpenFlow architecture. Open vSwitch is configured to run in the forwarding mode: each time when a packet of a flow arrives in a switch, the software tries to identify the corresponding flow. If there is defined an action for the respective flow, the switch forwards the packet to the proper port, otherwise the software encapsulates the packet and sends it to the controller which returns an action to be applied to. The implementation issues were discussed in details in [11].

Some observations are summarized herein. Thus the Modified Dijkstra's algorithm proved to be clever enough to detect the background traffic on a given link (e.g. direct path 1-4), and to recommend paths (e.g. 1-2-4 or 1-3-4) with a better End-to-End Available Transfer RateEEATR.

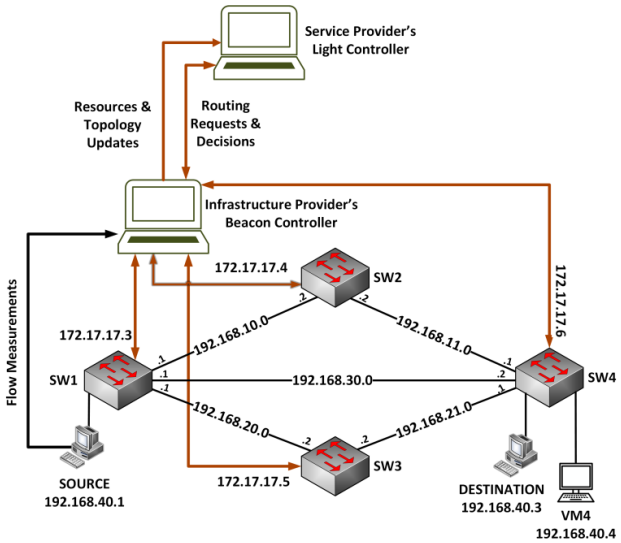


Figure5.Testbed for real implementation using OpenFlow

In comparison the Floyd-Warshall algorithm is summarizing the costs of all network segments of the path from 1 to 4, and may not detect background traffic. The conclusion was that MD is suitable to be used by SP (which nowadays needs more and more content-aware routing). However it is not efficient to apply it for

Infrastructure Provider, because it needs the minimum cost between any combination of two nodes (source and destination). Instead of running individual MD sessions for each node, using FW, it got all the results in a single run. The results are not actually used to perform routing, as their purpose is limited to the resource and topology allocation from IP to SP. We expanded the functionalities described in [11], by adding the following new features:

- Whenever the controller receives a routing request for a flow which does have neither the source, nor the destination in the list of the Infrastructure Provider machines, it forwards the request to the Service Provider. The communications between the IP and SP is made through two sockets (one for network updates, other to send routing requests IP to SP and to receive the routing decisions SP to IP. In any case, both IP and SP using the suitable algorithms, determine the entire path from the source to the destination. Only the IP is able to send commands to the switches, due to security reasons. Afterwards, the IP computes the output ports for every switch in the path and sends the corresponding commands. While the previous implementation in [11] was hop-by-hop based, the actual one is path-oriented (virtual path).

- The determination of the transfer rates for each flow is based on *conntrack* running on the SOURCE machine. It measures the throughput for all inbound and outbound flows and sends the statistics every one second directly to the controller.

- The route oscillation is reduced by a supplementary mechanism implemented herein. Thus if the current flow has the same source and destination as the previous flow, before applying the routing scheme to the virtual switches, the new route is compared to the previous one. In case it is different, the newest one will be taken into consideration only if the difference between the current EEATR and the previous one is greater than the sum of the throughputs of all flows through the respective route (see *Figure 6*).

$$EEATR_{current\ path} - EEATR_{previous\ path} > \sum_{(i,j) \in previous\ path} F_{ij} \quad (9)$$

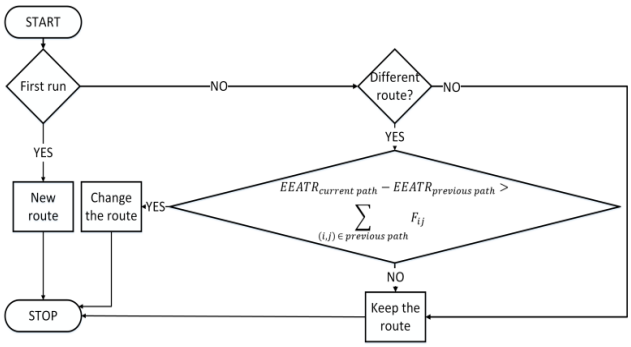


Figure 6.Reducing the route oscillation

IV. REAL IMPLEMENTATION USING OPENFLOW

We grouped the tests as in Table I. *Tests 1* and *2* referred to IP only, whilst *3* and *4* to SP only. *Tests 5-6*, involving both IP and SP, are actually implementing GRAS. Actually we have two instances of GRAS: one for the IP (*Test 2*) and another one for the SP (*Test 4*). Both are presented together in *Test 5-6*.

TABLE I: TESTING SCENARIOS

Test No.	Single path	Multi-path	Type of provider	Resources shared	Topology shared
1	FW	-	IP	No SP	No SP
2	FW	FF	IP	No SP	No SP
3	MD	-	SP	No IP	No IP
4	MD	FF	SP	No IP	No IP
5	MD	FF	SP	SP=: 100%IP	SP=: 100%IP
	FW	FF	IP		
6	MD	FF	SP	SP=: 100%IP	SP=: partial IP
	FW	FF	IP		

Test 1: Suppose six 30 Mbps-flows F_{1IP}, F_{2IP},F_{3IP}, F_{4IP}, F_{5IP} and F_{6IP}belonging to IP are sent from SOURCE to DESTINATION. $F_{1IP}$is started first. After 20 seconds $F_{2IP}$is issued, followed after another 20 seconds by F_{3IP} and so on. Using FW algorithm, the first flow is sent through the path 1-2-4. In order to avoid the routing oscillation all next flows will use the same path. After the third flow is injected into the network, the route 1-2-4 will be considered congested. According to [12], for a 100 Mbps link, the ATR in case of no background traffic is 82.051 Mbps (on top of MAC Sub-Layer) in the most pessimistic case, which is less than 90 Mbps needed for flows F_{1IP},F_{2IP}, and F_{3IP}. In this single-algorithm scenario, nothing can be done to improve the performances.

Test 2: This is similar to *Test 1* until the moment T3 (see *Figure 7*), when multipath with FF algorithm is activated in the GRAS approach for IP. According to Table II at moment T1 the best path was 1-2-4 with a cost 0.139202+0.068811=0.208013. This value is lower than the cost 0.190396+0.198407=0.388803 for path 1-3-4 and even lower than 0.239989 for the direct path 1-4.

Temporarily no congestion occurred compared to *Test 1*.

TABLE II: TEST 2 AT MOMENT T1

Link	ATR [bps]	OWD [s]	FW cost
1->2	82,049,400	0.000087	0.139202
2->4	82,049,400	0.000043	0.068811
1->4	82,049,400	0.000150	0.239989
1->3	82,049,400	0.000119	0.190396
3->4	82,049,400	0.000124	0.198407

Note that the Table III at moment T2 presents the costs of the paths before actually sending the flows. Thus, the path 1-2-4 is preserved, as 0.073941+0.096114=0.170055 is less than 0.354803 for 1-4 or 0.209593+0.108433=0.318026 for 1-3-4. However the third flow will be sent through the best link available 1-4 (according to greedy algorithm).

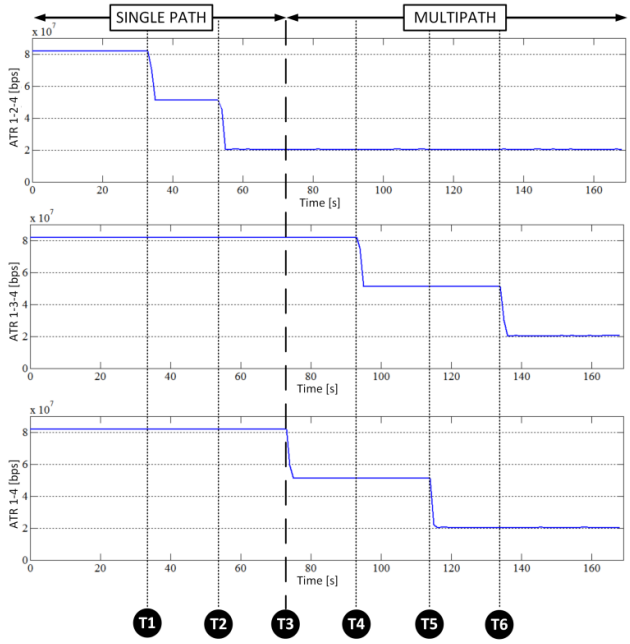


Figure 7.Test 2: ATR

TABLE III: TEST 2 AT MOMENT T2

Link	ATR [bps]	Total flows [bps]	OWD [s]	FW cost without flows
1->2	82,048,536	30,860,779	0.000030	0.073941
2->4	82,048,536	30,860,347	0.000039	0.096114
1->4	82,048,536	1,600	0.000144	0.354803
1->3	82,049,400	2,464	0.000131	0.209593
3->4	82,048,536	1,600	0.000044	0.108433

Table III provides also interesting practical results regarding the traffic for the unused links 1-3-4 and 1-4. Thus, 1,600 bps and respectively 2,464 bps refer to the OpenFlow signaling, plus the active probes every one second for measuring OWD, using the tool presented in [11],[12].

Test 3: Suppose six 30 Mbps-flows F_{1SP},F_{2SP},…F_{6SP} belonging to SP are sent from SOURCE to VirtualMachine4 (VM4), multipath routing being disabled (similar to *Test 1*, i.e. non-GRAS approach). All flows will be routed with the Modified Dijkstra's algorithm only. According to Table IV, F_{1SP} is routed through 1-4 (different than in *Test 1* for IP), whilst the next flows will follow the same path to reduce the routing oscillation.

TABLE IV: TEST 3 AND TEST 4 AT MOMENT T1

Link	Test 3		Test 4	
	ATR [bps]	OWD [s]	ATR [bps]	OWD [s]
1->2	82,049,400	0.000093	82,049,400	0.000086
2->4	82,049,400	0.000060	82,049,400	0.000052
1->4	82,036,102	0.000128	82,049,400	0.000134
1->3	82,049,400	0.000090	82,049,400	0.000092
3->4	82,049,400	0.000049	82,049,400	0.000050

Again congestion will occur at moment T3 and no improvements are possible.

Test 4:Similar to *Test 3*, six SP flows are sent from SOURCE to VM4, but the multipath routing is now enabled (i.e. GRAS for SP). See Table V for the routing decisions.

TABLE V: TEST 4 ROUTING DECISIONS

Path	T1	T2	T3	T4	T5	T6
1-2-4	-	-	F_{3SP}	F_{3SP}	F_{3SP}, F_{5SP}	F_{3SP}, F_{5SP}
1-3-4	-	-	-	F_{4SP}	F_{4SP}	F_{4SP}, F_{6SP}
1-4	F_{1SP}	F_{1SP}, F_{2SP}	F_{1SP}, F_{2SP}	F_{1SP}, F_{2SP}	F_{1SP}, F_{2SP}	F_{1SP}, F_{2SP}

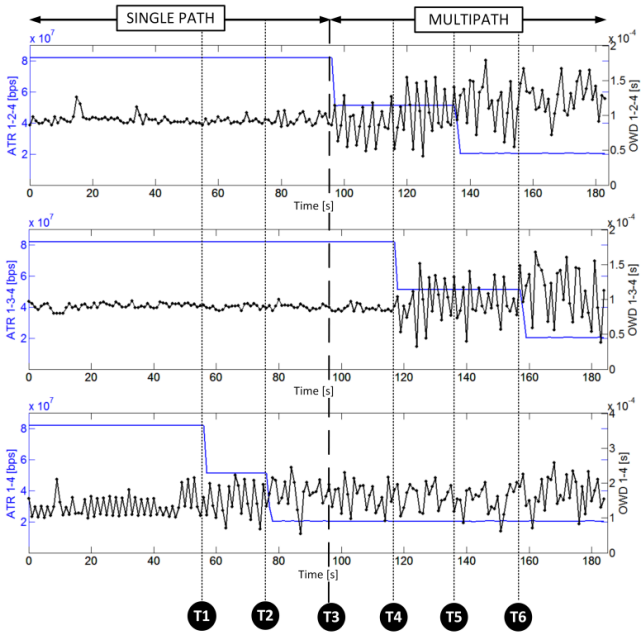


Figure 8.Test 4: ATR and OWD

Observe in *Figure 8* that in order to avoid congestion at moment T3, F_{3SP} is routed through the best path available: 1-2-4 (not significantly different than 1-3-4). Afterwards, 1-3-4 is the least used path, chosen for F_{4SP}. Similar, F_{5SP} uses 1-2-4 and F_{6SP} uses 1-3-4. Temporarily the congestion was avoided.

Test 5: F_{1IP}, F_{2IP} and F_{3IP} are sent from SOURCE to DESTINATION, whilst F_{1SP}, F_{2SP} and F_{3SP} are from SOURCE to VM4 (i.e. GRAS for both IP and SP). Suppose 100% of the resources and of the topology are shared by IP with SP. Each pair S-D and S-VM4 issued the flows concurrently. Because the network is unloaded, except negligible traffic for signaling and measurements, after the test started, SP and IP took different routing decisions, according to Table VI. The multipath routing was enabled earlier at moment T2. Both providers decided to use the path 1-2-4 at moment T3 to avoid temporarily the congestion.

TABLE VI: TEST 5 ROUTING DECISIONS

Path	T1	T2	T3
1-2-4	-	-	F_{3SP} , F_{3IP}
1-3-4	F_{1SP}	F_{1SP} , F_{2SP}	F_{1SP} , F_{2SP}
1-4	F_{1IP}	F_{1IP} , F_{2IP}	F_{1IP} , F_{2IP}

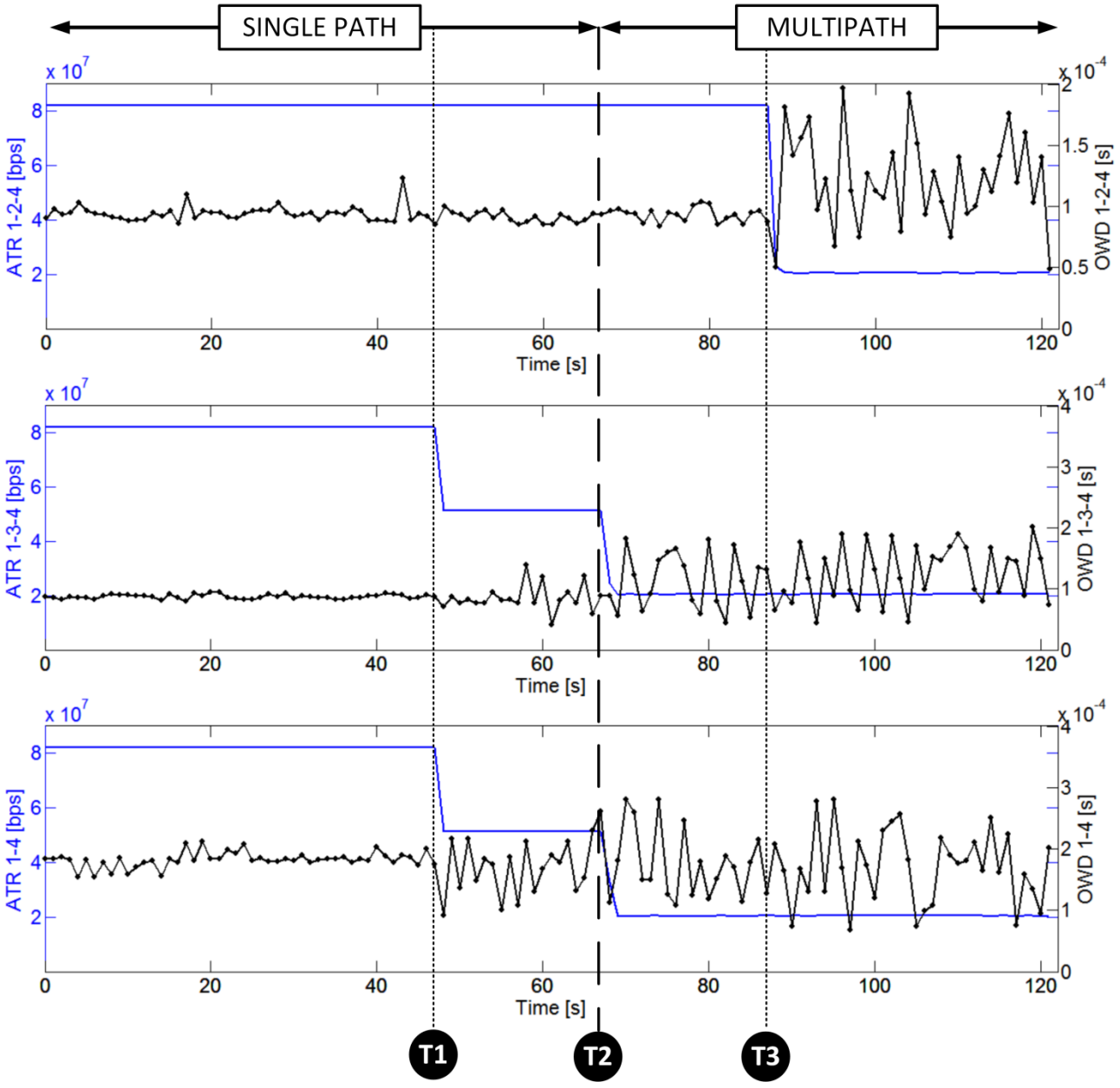


Figure 9.Test 5: ATR and OWD

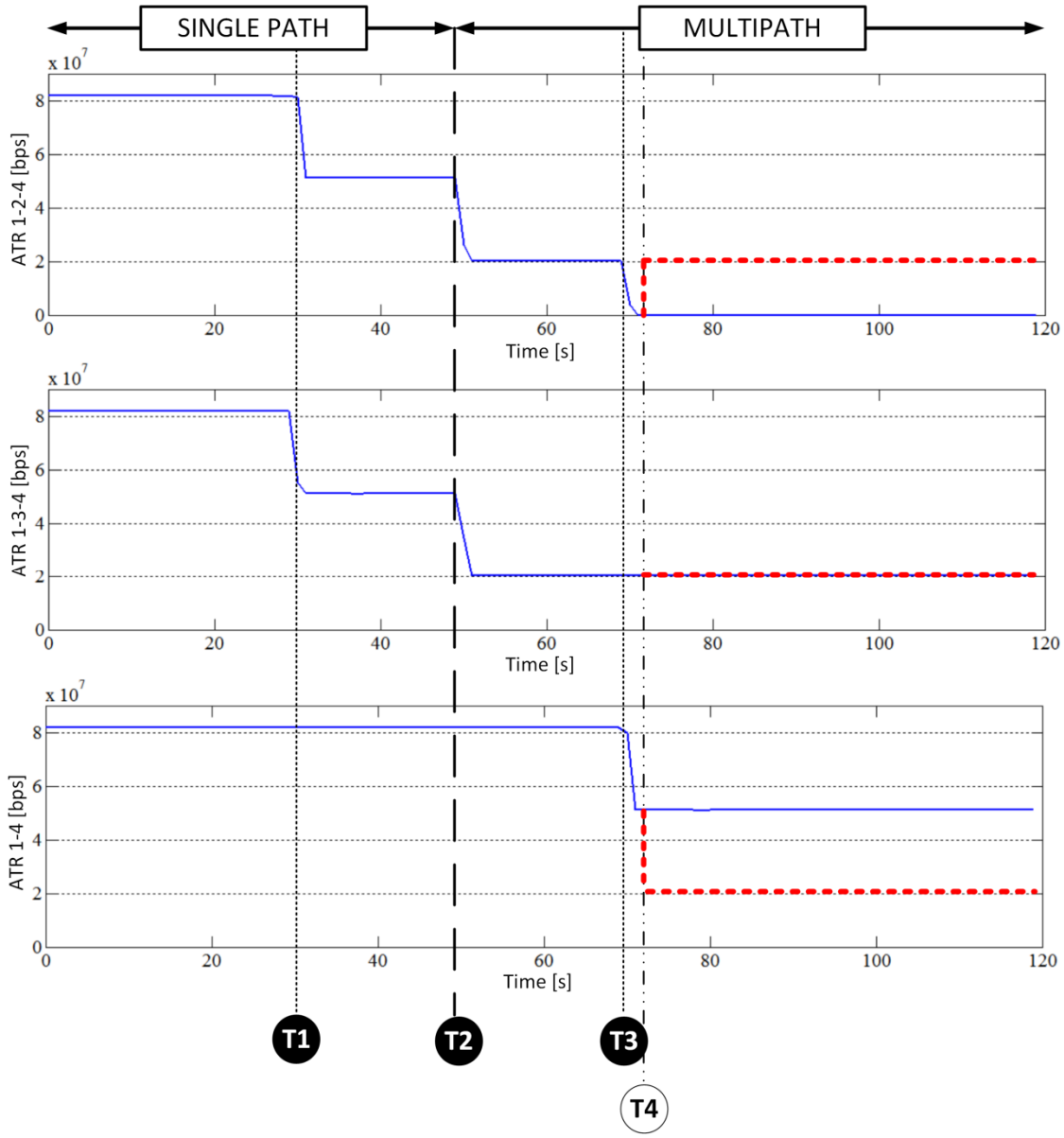


Figure 10. Test 6: ATR

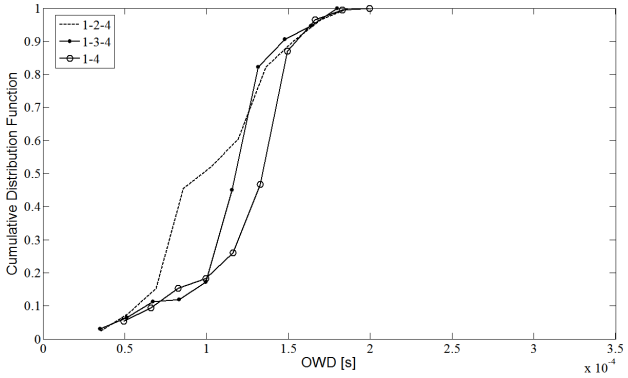
Test 6: This scenario also involves GRAS for both IP and SP, butIP does not share entirely the topology with SP (see Table I). We presumed that the direct path 1-4 is not available at all for SP. The flows can be distributed as in *Test 5*, except the moment when congestion occurs on a path guaranteed to the SP by SLA (Service Level Agreement). Actually we are trying to deteriorate the environment conditions for both providers to see how GRAS could help. As IP does not completely exploit the direct path 1-4, it could modify the current allocation of its own flows when it is needed (for instance in moment T4). The dotted line in *Figure 10*shows how IP can avoid congestion on path 1-2-4 by movingF_{2IP} from 1-3-4 to 1-4. On the other hand, SP at its turn should move $F_{3SP}$from congested path 1-2-4 to 1-3-4.

TABLE VII: TEST 6 ROUTING DECISIONS

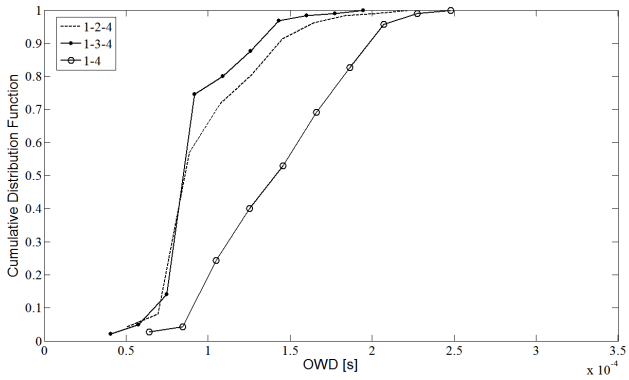
Path	T1	T2	T3	T4 with SLA
1-2-4	F_{1SP}	F_{1SP} , F_{2SP}	F_{1SP}, F_{2SP}, F_{3SP}	F_{1SP}, F_{2SP}
1-3-4	F_{1IP}	F_{1IP} , F_{2IP}	F_{1IP} , F_{2IP}	F_{1IP} , F_{3SP}
1-4	-	-	F_{3IP}	F_{2IP} , F_{3IP}

This was just an example of how GRAS is performing adaptive routing to avoid congestion and to improve the end-to-end performances compared to legacy solutions.

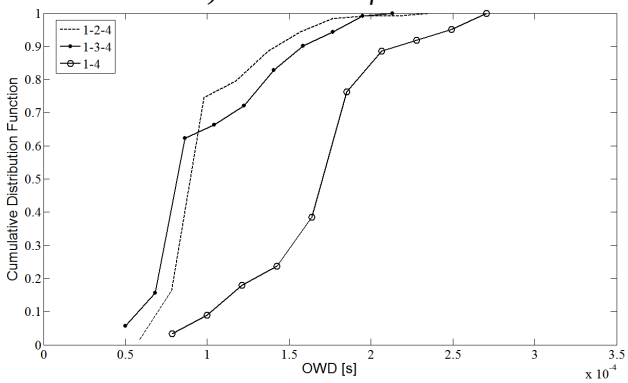
Let us have now an overall perspective and interpret the results in a correlated manner. The tests did not start simultaneously and did not last exactly the same amount of time. Thus the reference for each test wasthe moment T1.



a) Test 2: GRAS for IP



b) Test4: GRAS for SP

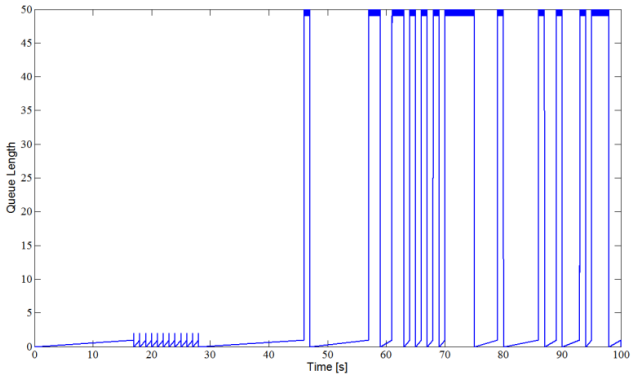


c) Test 5: GRAS for IP and SP

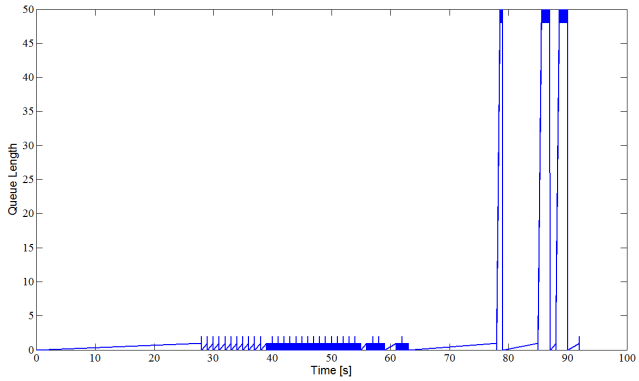
Figure 11. CDF of OWD for paths 1-2-4, 1-3-4 and direct path 1-4

The cumulative distribution function of one-way delay per path (not per packet!) depicted in *Figure 11a* shows that GRAS for IP has a more uniform use of resources. Due to FW's fairness the delays are grouped within the range $0.5\ldots1.5*10^{-4}s$. On the other hand GRAS for SP (*Figure 11b*) and GRAS for IP and SP (*Figure 11c*) has better performances for some paths (1-2-4, 1-3-4) and worst performances for 1-4. Due to the algorithm to reduce route oscillation, MD has less sensitivity for ATR variation, but still reacts to OWD variation. As a common observation, FF helped to postpone the congestion as much as possible.

We conducted prior simulations in OMNET++ for approximately the same testing scenarios as those discussed herein, but it is out of the scope of this paper to comment them in details. Having in mind that the real implementation using OpenFlow cannot reproduce exactly the simulation, we did not compare the absolute values of ATR and OWD. Thus, we proved the benefits of running an adaptive scheme instead of a single algorithm approach. The queue length versus time, as a key performance indicator, is depicted in *Figure 12*.



a) FW



b) GRAS

Figure 12. Queue length in OMNET++ simulations for Infrastructure Providers

Observe that GRAS is not a miracle solution, so it could face congestion too. With respect to the number of packets lost in 100-second simulation (see *Figure 12*), OMNET++ counted 93,717 drops for FW approach, and only 2,079 for GRAS.

V. CONCLUSIONS

This paper managed to prove that the concept of gearbox-like routing algorithm selection is fully implementable using OpenFlow. The Infrastructure Provider IP is actually performing the routing on behalf of Service Provider SP because some of the nodes may not belong to SP. Furthermore SP may not be authorized to perform

routing directly due to security reasons. SP decisions are taken based on limited resources and topology assigned by IP.

It is for further work to expand gearbox-like routing algorithm selection in runtime for Virtual Service Infrastructure Providers. It requests inter-domain routing schemes for the layer in-between IP and SP.

ACKNOWLEDGMENT

We acknowledge the valuable support obtained from Zsombor Muzsai (during his work with UCLabs), who previously implemented original OMNET++ simulations of Floyd-Warshall and Ford-Fulkerson. The results helped us to design the real implementation of GRAS using OpenFlow. Thanks go also to Gabriel Lazar for support in dealing with flow identification issues.

REFERENCES

[1] H.E. Egilmez, S. Civanlar, A.M. Tekalp, "An Optimization Framework for QoS-Enabled Adaptive Video Streaming Over OpenFlow Networks," *IEEE Trans. on Multimedia*, Vol.15, No.3, pp.710-715, April 2013

[2] S. Latre, J. Famaey, F. De Turck, and P. Demeester, "The Fluid Internet: Service-Centric Management of a Virtualized Internet", *IEEE Communications Magazine*, Vol.52, No.1, January 2014, pp. 140-148.

[3] P. Calyam et al., "Leveraging OpenFlow for resource placement of virtual desktop cloud applications," *2013 IFIP/IEEE International Symposium on Integrated Network Management*, pp.311-319, May 2013

[4] I.M. Seung, R. Kamal, H. ChoongSeon, L. Sungwon, "Towards opportunistic flow management in OpenFlow," *2013 IFIP/IEEE Int. Symposium on Integrated Network Management*, pp.696-699, May 2013

[5] C. De Cusatis et al., "Dynamic, software-defined service provider network infrastructure and cloud drivers for SDN adoption", *ICC 2013*, pp.235-239.

[6] C.D. Jingguo Ge, W. Yulei; E. Yuepeng, and Y. Junling, "Performance Analysis of Load-Aware Anycasting Based on OpenFlow," *12th IEEE TrustCom 2013*, pp.1868,1872, 16-18 July 2013

[7] R. Pries, M. Jarschel, and S. Goll, "On the usability of OpenFlow in data center environments," *ICC 2012*, pp.5533-5537, 10-15 June 2012

[8] S.H. Yeganeh, A. Tootoonchian, and Y.Ganjali, "On scalability of software-defined networking", *IEEE Communications Magazine*, Vol.51, No.2, pp.136-141, February 2013

[9] Cisco Open Networking Environment: Adaptable Framework for the Internet of Everything [Online], Available:http://www.cisco.com/go/one.

[10] A.B.Rus and V.Dobrota, "Case Study of a Gearbox-Like Routing Algorithm Selection in Runtime", *18th IEEE LANMAN 2011*, Chapel Hill, North Carolina, pp. 1-6, DOI: 10.1109/LANMAN.2011.6076938

[11] A.G.Furculita, M.V.Ulinic, A.B.Rus, and V.Dobrota, "Implementation Issues for Modified Dijkstra's and Floyd-Warshall Algorithms in OpenFlow", *12th RoEduNet International Conference*, Constanta, Romania, 2013, pp.50-55, DOI:10.1109/RoEduNet.2013.6714208

[12] A.B. Rus, V. Dobrota, A. Vedinas, G. Boanea, and M. Barabas, "Modified Dijkstra's algorithm with cross-layer QoS", *ACTA TECHNICA NAPOCENSIS, Electronics and Telecommunications*, vol. 51, no.3, 2010, pp. 75-80.

[13] Open Flow [Online], Available: http://www.openflow.org

[14] N. McKeown, et al., "OpenFlow: enabling innovation in campus networks", *ACM SIGCOMM Computer Communication Review*, 38 (2), April 2008, 69-74