Using Software Defined Networks for energy saving without compromising Quality of Service

Article · March 2020

CITATIONS
0

READS
31

3 authors, including:

Irena Seremet
University of Sarajevo
11 PUBLICATIONS **8** CITATIONS
SEE PROFILE

Sasa Mrdovic
University of Sarajevo
39 PUBLICATIONS **158** CITATIONS
SEE PROFILE

Using Software Defined Networks for energy saving without compromising Quality of Service

Irena Šeremet[1†], Saša Mrđović[2††] and Samir Čaušević[3†††]

Faculty of Traffic and Communication, Faculty of Electrical Engineering, Faculty of Traffic and Communication, Sarajevo, Bosnia and Herzegovina

Summary
The main goal of our paper is to show how to save energy in the network by turning off underutilized ports/links/modules/devices without compromising QoS. The idea is to use only the best path for transmitting packets and turn off other network components in order to save energy. If congestion on the best path occurs, the second-best path is powered on and traffic is load-balanced between two paths. If congestion ever occurs on these two paths, the third-best path is powered on and so on. The number of used paths depends on link utilization on paths. For finding optimal and backup paths we use a modified Lagrange relaxation-based aggregated cost (LARAC) algorithm. Software Defined Network controller constantly measures link utilization in the whole network. By turning off/on network components on the paths, the controller avoids congestion and packet drops, but at the same time saves energy.

Key words:
Software Defined Network, Energy Efficiency, Quality of Service

1. Introduction

Growth in the number of connected devices, processing, and storage means that the energy used to power the Internet is growing substantially [1]. The energy consumption is one of the key challenges for the ICT industry and it is only expected to grow in importance [2]. In 2010 there was a 6.5% increase in electricity consumption over 2009 [3]. The BP Statistical Review of World Energy [4] states that electricity generation in the world was 25,551 TWh in 2017, and energy consumption consumed by the ICT was at best 1,982 TWh per year, via expected development to 2,547 TWh per year, and in the worst-case scenario 3 422 TWh. The expected outcome of the Internet's electricity consumption is thus quite exactly 10%. As presented in Figure 1, the share of ICT global electricity usage by 2030 has been estimated in the study. The analysis in [5] shows that for the worst-case scenario, ICT could use 21% of global electricity in 2020, and 51% of global electricity in 2030. For 2030 only 8% of global electricity is the best case. On the other hand, Morley et al [1] states that, by 2030, "smarter" systems could save power consumption by 10 times.

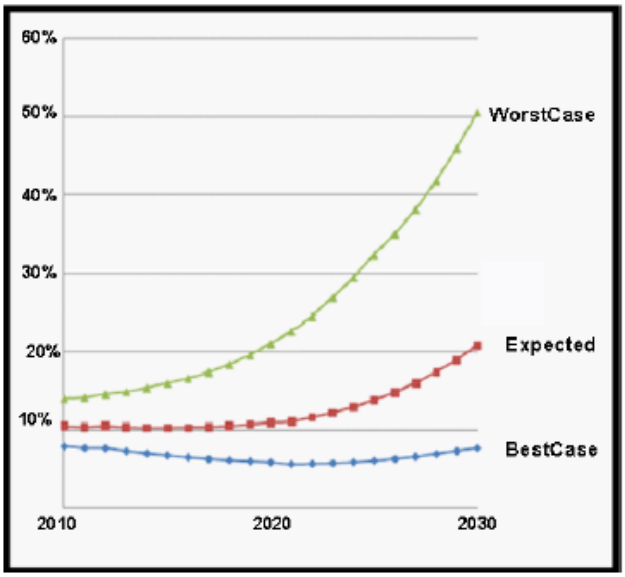

Fig. 1 ICT share in global energy consumption by 2030 [5]

ICT electricity usage is divided into

- end devices electricity usage
- network infrastructure electricity usage
- electricity usage in data centers
- electricity usage in the production of the above categories [5]

Study [2] suggests that communication networks are responsible for 3.5% of global electricity consumption and that 2.6% of that is directly attributable to consumer activity. Therefore, energy efficiency in the core and metro networks will also be of utmost importance for the sustainability of the future Internet. Undoubtedly, the amount of traffic in the metro and core networks will also increase significantly in the next years. The energy consumption issue in networks is global and affects all service providers and content providers. The challenge of reducing energy consumption has been recognized by the ICT sector. Generally, the energy efficiency of new releases of network equipment has improved by 10% to 20% year over year and continues to do so [2].

Manuscript received March 5, 2020
Manuscript revised March 20, 2020

New technologies that are coming to market are more energy-efficient than previous generations: for example LTE versus 2G or 3G, VDSL2, Network Function Virtualization (NFV), Software Defined Networks (SDN), etc. [2]

In this paper, a special focus is on using SDN as a new technology that can enable power savings in networks. SDN is capable of automatic traffic managing and turning off/on network components that are underutilized in order to decrease energy consumption. But even with SDN, determining which ports/links/devices to turn off and on without compromising QoS is quite challenging. In this paper, we present our approach of saving energy in SDN networks without compromising QoS.

In the second section, we explained briefly SDN technology, QoS and energy efficiency background. Third section presents related work. We analyzed energy consumption in the real network in section four. In the fifth section, we presented our approach of saving energy without compromising QoS, and we implemented that strategy in the network simulator in section six. We concluded our paper in section seven.

2. Energy Efficiency, SDN and QoS Background

Energy efficiency solutions can be hardware and software-based. Hardware-based solutions such as designing optimal energy-efficient topology, purchasing energy-efficient equipment or modeling energy-efficient TCAMs are more static solutions. Once implemented, energy consumption is fixed and cannot be dynamically optimized. On the other hand, software-based solutions are more dynamic. Software-based solutions are traffic-aware solutions and include Adaptive Link rate (ALR) and Energy Aware Routing (EAR) [6]. ALR is a method of reducing energy consumption by dynamically varying link data rates in response to the utilization of a link. EAR uses routing algorithms in which unused ports or underutilized links are turned off in order to increase energy savings. If all ports on the line-card are turned off, the whole line-card could be turned to standby mode. Also, if all line-cards are in standby mode, the whole router can be turned into the standby mode. Network components are often underutilized, which is the main motivation for developing and implementing traffic-aware solutions for saving energy. The undeniable fact is that monitoring, calculating, path computation and turning on/off of components should be dynamic and automatic. Software Defined Networks (SDN) seem to be the perfect solution for that. SDN is an architecture that decouples control and forwarding plane. In non-SDN networks, each device has its own control and data plane. The control plane contains all the intelligence of the router and is responsible for making routing decisions and exchanging protocol information with other network devices.

The data plane or the forwarding plane is responsible for packets forwarding from incoming interface to outgoing interface through the router. SDN architecture [7] consists of three layers: infrastructure, control and application. Each layer has its own role in the architecture. On the infrastructure layer are network devices and their connecting links. Routers on the infrastructure layer only have data plane and forwarding tables, which contain information about incoming and outgoing interfaces. Information found in forwarding tables were not obtained by network devices, but by network controller placed in the control layer. A centralized controller in the control layer has a global view of the whole network and provides network management according to forwarding policies. SDN controller is the core of the SDN, and it is placed between network devices and applications. SDN controller decides how network flows should be treated and gives instructions to network devices through protocols known as southbound protocols (SBI), and on the other hand, takes instructions from applications/users/engineers through protocols known as northbound protocols (NBI). In literature, the most mentioned southbound protocol is OpenFlow, but other industry forums such as IETF have also been developing similar concepts such as PCEP, BGP-LS, NETCONF/YANG/OpenConfig, SNMP and others. On the other hand, northbound application program interfaces (APIs) are usually SDN REST APIs used to communicate between the controller and application layer. Network engineers through APIs give instructions to a controller. A controller can be programmed to monitor links, power up or down interfaces, control security, Quality of Service (QoS) and so on. With SDN solution, network programmability, automation and rise of effectiveness is achieved.

The main goal of QoS is to achieve the best possible values of Key Performance Indicators (KPIs) such as bandwidth, latency, jitter and packet loss. QoS relies on several techniques that are applied to incoming traffic such as:

- classification and marking
- bandwidth allocation
- congestion management
- congestion avoidance

After incoming traffic arrives in network device the first step is to differentiate traffic into classes, such as voice traffic class, video traffic class and so on. After differentiating traffic into classes, every class is marked with a specific value. By examining that value, network devices can decide how to treat packets in every class. After classifying and marking traffic, bandwidth is allocated considering different network demands of different types of traffic. Network congestion occurs when the network device is receiving more traffic than it can handle. One way

(among the others) to avoid congestion is to implement a QoS-aware routing algorithm.
Guck et al [8] presented a high-quality overview of 26 QoS DCLC (different delay-constrained least cost) routing algorithms and compared their runtime and cost-efficiency. Analyzed DCLC algorithms are categorized in five main categories:

- elementary algorithms
- algorithms based on a priority queue
- algorithms based on Bellman - Ford
- algorithms making use of the Lagrange relaxation optimization technique
- algorithms making use of the knowledge of the least-cost (LC) and least-delay (LD) paths in the network

The best results have been achieved by using an algorithm that uses a Lagrange relaxation optimization technique named Lagrange relaxation based aggregated cost (LARAC). We also used an improved LARAC algorithm in our paper.

3. Related work

Many researchers have studied the issue of energy efficiency in different parts of the network, using different technologies, methods and algorithms on different levels.
Some papers are more focused on saving energy in data centers [9] [10] [11] [12] and other are more focused on saving energy in access and backbone networks [13] [14] [15].
Researches such as [16] [17] focused on software-based solutions, more precisely on traffic-based solutions of saving energy. In order to improve availability and reliability in networks, many network administrators install redundant links and devices. The real challenge is to determine algorithms that make idle links and devices in order to improve energy efficiency, but at the same time do not affect network performance, availability and reliability. Authors in [16] proposed switching off more edge devices in SDN networks by using a link based genetic algorithm (LBGA). Their results have shown energy savings up to more than 55 % during the hours when the edges are inactive. Similar idea of switching off edge devices had authors in [18]. Their first aim is to reduce number of active edges in the network. After that, a load balancing mechanism is carried out to reduce the variations in resource utilization among edge devices. In [19] authors presented a new edge weight optimization algorithm known as Grey Wolf Aware Load balancing and Energy saving(GLE). Authors in [17] considered a novel multidomain network service deployment framework by integrating SDN architecture and NFV technology.
When SDN was introduced, it became clear that SDN can be used as a tool for energy saving. Aseffa et al [14] [20] [21] [22] [23] [24] have several types of researches and survey papers on Energy Efficiency in Software Defined Networks.
In [14] [20] they have presented a nice classification of energy-efficient hardware and software-based methods and approaches in SDN. Hardware-based methods focused on TCAM compression, and software-based approaches are divided on (i) Traffic-Aware, (ii) End System Aware and (iii) Rule Placement. Each of these methods is nicely explained and substantiated with a large number of related papers. In [20] authors propose an IP formulation for traffic and energy-aware routing problems based on link utility information and evaluate the algorithms using real traces of different traffic volumes and network topologies. In [21] authors proposed a traffic-aware energy-efficient framework for SDN and heuristics algorithm that maintains the tradeoff between efficiency and performance. Later, in [22] authors continued researching trade-off between energy efficiency and network performance. They proposed a metric named Ratio for Energy Saving in SDN (RESDN) that quantifies energy efficiency based on link utility intervals. Zemmouri et al [25] presented four different computationally efficient algorithms namely (i) greedy first fit, (ii) greedy best ft, (iii) greedy worst fit and (iv) meta-heuristic genetic algorithm to solve the problem related to a distribution of traffic flows over pre-calculated paths which allow adapting the transmission rate of maximum links into lower states.
Our paper, similar to [13] [26] [27], is based on finding the best QoS-supported route and turning off the other links. In order to find the best QoS-supported route, the optimal QoS routing algorithm has to be implemented. Traditional routing algorithms (such as Shortest Path) and energy-efficient routing algorithms (such as Least Consumption) can only evaluate one routing parameter (i.e. Delay or energy consumption) at the time. Hence, authors in [27] used a Multi-Objective Evolutionary Algorithm (MOEA). An MOEA solves problems involving multiple connecting objectives using evolutionary mechanisms. But still, MOE algorithm uses so-called Pareto optimal solutions. A solution is Pareto optimal when none of the values can be improved without degrading some of the other values. In their simulation, three types of traffic are considered: IPTV, VoIP and Internet. For VoIP traffic, the objectives are to minimize both delay (objective one) and energy consumption (objective two); for IPTV traffic, the objectives are to minimize the blocking rate (objective one) and energy consumption (objective two). The objective of web-surfing traffic is to minimize energy consumption. This means that energy savings are obtained on the low priority traffic while QoS for high priority traffic is not degraded. Fernandez et al [26] also used the MOE algorithm that is based on the Strength Pareto Evolutionary Algorithm

2 (SPEA2) in order to enable the reduction of power consumption without degradation of the performance in SDN. Hongyu et al [13] proposed routing strategy which is especially aimed at QoS-guaranteed energy saving.
They used the BNESS (Backbone Network Energy Saving Strategy) algorithm that is integrated with OSPF and uses the Maximum Clique Problem (MCP) to search idle links during low traffic periods. Those idle links are then put into sleeping mode and energy saving is achieved. Similar to our paper, Liu et al in [28] use Lagrange relaxation based aggregated cost (LARAC) and K-Dijkstra combined algorithm to get the top K energy-minimum paths that satisfy the QoS in polynomial time. But unlike our paper, these algorithms are implemented in software-Defined multi-hop wireless networks (SDMWN).

4. Problem statement

Most of today's networks have poor energy efficiency. The main reason is running different network components at full capacity all the time regardless of different loads of traffic during the day. In this section, we analyze energy consumption in the real network. A part of the network topology of a real service provider in Bosnia and Herzegovina is shown in Figure 2.
Topology contains six Cisco C7606 - S routers connected with ten 10Gb links. Every router has a total of six modules. Two of six modules are processor cards, two modules are uplink cards and two modules are line cards for connecting end-devices such as DSLAMs, base stations, etc. Routers are connected between themselves through uplink cards. All ports on the uplink module have a maximum bandwidth of 10 Gb/s. To achieve redundancy on a module level, routers use separate uplink modules to connect to other routers that provide backup routes. For example, Router0 is connected to Router1 via module 2 (port TenGi2/1) and Router2 via module 3 (port TenGi3/1). That way, if module 2 on Router0 ever fails, Router 0 will have a backup route through module 3. Each port, module and device consume a certain amount of power. The amount of consumed power can be seen in the output of a command show power on the routers.
Topology information are presented in Table 1, and average measured energy consumption in the considered network is presented in Table 2.

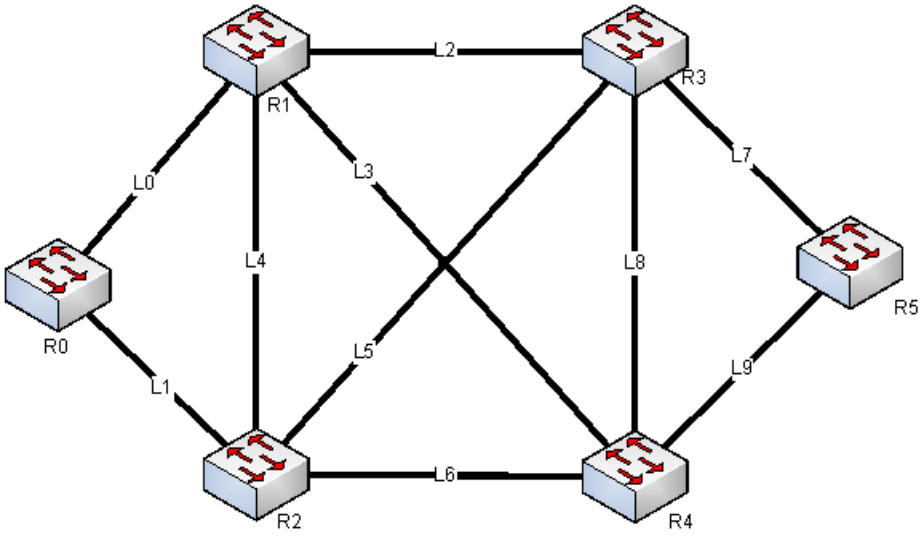

Fig. 2 Network topology of a real service provider

In order to prove network inefficiency, we analyzed the Router0 connection link to Router1 and Router2. On monitoring system, we recorded traffic level on links in 12 hours period. The highest level of traffic is between 21:00 and 23:00, and the lowest level of traffic is between 03:00 and 07:00. As an example, the graph of bandwidth usage on Router 0 link to Router1 (link L0) and Router 0 to Router 2 (link L1) is shown in Figures 3 and 4 respectively. In a period from 03:00 to 07:00, both links had the bandwidth usage of only 10% (1 Gb/s). During that time, both uplink modules consumed full energy. The solution of this energy-wasting problem could be better traffic management and putting unused ports and modules into sleep mode.
If total traffic of 2 Gb/s went through only one link, the other link could be put into sleep mode.
With this energy-saving practice, the savings could be significant. To apply the most efficient energy-saving solutions in networks, the view of the network should be global. That way, SDN controller would calculate the states on all links, find the best paths and put in the sleep mode other links. Also, since traffic level varies through the day, network devices should be programmed to constantly

measure states in the links and take appropriate actions. This approach can achieve up to 30% of energy saving [29]. Ideally, the whole process should be automated.

Table 1: Topology information

Device	Link	Port	Module
Router 0	R0 - R1	TenGi 2/1	Module 2
	R0 - R2	TenGi 3/1	Module 3
Router 1	R1 - R0	TenGi 2/1	Module 2
	R1 - R2	TenGi 3/1	Module 3
	R1 - R3	TenGi 2/2	Module 2
	R1 - R4	TenGi 3/2	Module 3
Router 2	R2 - R0	TenGi 2/1	Module 2
	R2 - R1	TenGi 3/1	Module 3
	R2 - R3	TenGi 3/2	Module 3
	R2 - R4	TenGi 2/2	Module 2
Router 3	R3 - R1	TenGi 2/1	Module 2
	R3 - R2	TenGi 3/1	Module 3
	R3 - R4	TenGi 3/2	Module 3
	R3 - R5	TenGi 2/2	Module 2
Router 4	R4 - R1	TenGi 3/1	Module 3
	R4 - R2	TenGi 2/1	Module 2
	R4 - R3	TenGi 3/2	Module 3
	R4 - R5	TenGi 2/2	Module 2
Router 5	R5 - R3	TenGi 2/1	Module 2
	R5 - R4	TenGi 3/1	Module 3

Table 2: Average measured energy consumption in the considered network

Modules	FAN	TCAM	Number of ports	Energy Consumption
76-ES+T-40G	180 W	~30 W	40	418 W
76-ES+T-2TG (UPLINK)	180 W	~30 W	2	300 W
76-ES+T-2TG (UPLINK)	180 W	~30 W	2	300 W
7600-ES+20G3C	180 W	~30 W	20	276 W
RSP720-3C-GE	180 W	~30 W	2	310 W
RSP720-3C-GE	180 W	~30 W	2	310 W
Total on all modules per device: 1 914 W				
One device in total: 2 124 W				
All devices in total: **12 744 W**				

This is not possible in traditional IP networks, but Software Defined Networks seems to be the perfect solution for that. SDN controllers are programmable machines that have a global view of the whole network, enable automation and insight into the link utilization.

A controller can determine the primary path in the network and turn off other underutilized links. If utilization on the primary path ever becomes too high and congestion occurs, SDN controller can again turn on other paths.

5. Energy saving without compromising QoS

5.1 The main idea

Consider the network topology that can be modeled as a directed, connected graph G= (R, L) where R represents nodes/routers R= (r_1, r_2 . . . r_n) connected with links L= (l_1, l_2 . . . l_n). Each link has characteristics such as cost (c_1, c_2 . . . c_n), delay (d_1, d_2 . . . d_n), bandwidth (b_1, b_2 . . . b_n), utilization (u_1, u_2 . . . u_n), and energy consumption (e_1, e_2 . . . e_n). Let assume that energy consumption on each link is identical: e_1=e_2=. . . =e_n. There is a finite number of possible paths/routes (P) from the source node (s) to the destination node (t). The first step is to find all possible paths that satisfy delay constraint and sort them from the best to the worst by cost and delay criteria. The best path ($P_{optimal}$) contains network components such as nodes $R_{optimal}$ and links $L_{optimal}$. SDN controller measures link utilization globally in the network. If link utilization on the $P_{optimal}$ is lower than the defined threshold, only the best path is used. All other network components are powered off in order to save energy. If congestion on the best path occurs, and link utilization on the $P_{optimal}$ becomes higher than the threshold, the second-best path ($P_{secondary}$) is powered on and traffic is load-balanced between two paths. If congestion occurs again, the third best path ($P_{tertiary}$) is powered on and so on. In order to achieve load-balancing of traffic due to congestion, paths should not contain links from other paths. In order to save energy without compromising QoS, we have put together a list
of requirements that must be satisfied:

- The optimal path should have the best cost path in the network and satisfy delay constraint at the same time
- The backup optimal path should be the second-best cost path in the network and satisfy delay constraint at the same time
- The backup optimal path should not contain the same links as the optimal path
- The N-optimal path should have the n-best cost, satisfy delay constraint at the same time and should not include links from other paths
- SDN controller constantly measures link utility on the optimal path

- If link utility on the optimal path is lower than 85%, only the optimal path is used; network components on other paths are powered off in order to save energy

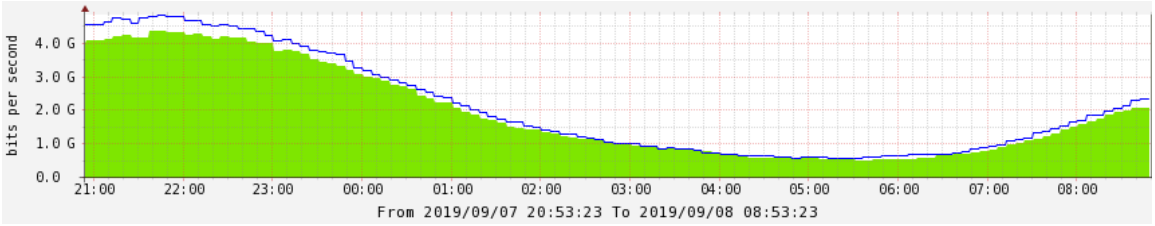

Fig. 3 Bandwidth usage on link L0 of Router 0

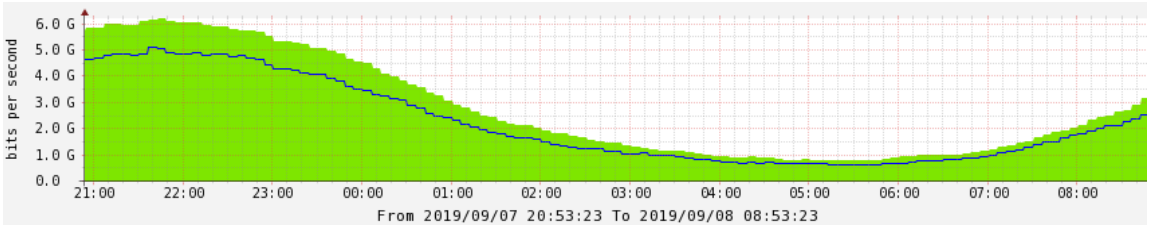

Fig. 4 Bandwidth usage on link L1 of Router 0

- If link utility on the optimal path is 85% or greater, SDN controller powers on network components on the backup path and load-balance traffic on two paths.
- If link utility on the backup path is 85% or greater, SDN controller powers on network components on the third-best path
- If link utility on the (n-1)th path is 85% or greater, SDN controller powers on network components on the n-th best path
- If link utility on the n-th best path becomes lower than 85% again, and the sum of traffic levels on the n-th best path and the (n-1)-th best path is lower than 85% of the maximum throughput of (n-1)-th best path, n-th best path should be turned off.

5.2 Problem formulation

The definition of the Delay Constrained Least Cost path problem (referred hereafter simply as DCLC) is the following [30]: Given a directed, connected graph G(R; L), a non-negative cost c(l) and a non-negative delay d(l) for each link l ∈L, a source node s, a destination node t, and a positive delay constraint Δdelay. The constrained minimization problem is presented as:

$$min_{p \in P'(s,t)} \sum_{e \in P} c(e) \qquad (1)$$

where P'(s; t) is the set of paths from s to t for which the end-to-end delay is bounded by Δdelay. Namely a p ∈ P(s; t) is in P'(s; t) if and only if

$$\sum_{e \in P} d(e) \leq \Delta_{delay} \qquad (2)$$

So routing of a single user is described formally as follows:

$$\min \{e(p) : p \in P(s, t) \text{ and } d(p)\} \leq \Delta_{\text{delay}} \qquad (3)$$

The DCLC problem is NP-hard [31], but there is a special solvable case in polynomial time where all link costs or all link delays are equal [32].

5.3 LARAC background

Lagrange Relaxation based Aggregated Cost algorithm (LARAC) is based on Lagrange relaxation [33]. The original LARAC algorithm is proposed by [34] and is based on the heuristic of minimizing $c_l = c + l\,d$ modified cost function. For known l, minimal path (p_1) can be calculated. In the first step of the LARAC algorithm, l is set to 0. The algorithm uses Dijkstra algorithm Dijkstra (s, t, c), Dijkstra (s, t, d) and Dijkstra (s, t, c_1) with respect to the multiplier l. Optimal solution is found if $l = 0$ and $d(p_1) \leq \Delta$delay. If $d(p_1) >$ Δdelay, the algorithm will store the path as the best path that does not satisfy the delay condition (p_c), and check

whether a solution can be found or not by using again Dijkstra in order to calculate the shortest path on delay d. If the obtained path does not satisfy the delay requirement, solution does not exist and the algorithm stops. Otherwise, the algorithm stores this path as the best path found until that moment (p_d).

Value of λ should be increased and values of p_c and p_d should be updated with other paths until an optimal λ is found. By calculating λ from (4) we will get aggregated cost c_λ. With this value of λ, we can find a new c_λ -minimal path r. By comparison, if $c_\lambda(r) = c_\lambda(p_c) = c_\lambda(p_d)$, we will obtain the optimal λ. Otherwise, we will set r as the new p_c or p_d depending on whether r is feasible or not. Improved LARAC algorithm stores former results of calculation and it can reuse them for different destinations.

$$\lambda = \frac{(p_c)-(p_d)}{(p_d)-(p_c)} \quad (4)$$

The LARAC algorithm [34] :

Procedure LARAC (s,t,c,d, Δdelay)
Pp: = Dijkstra (s,t,c)
If $d(p_c) \leq \Delta$delay then return p_c
P_d: = Dijkstra (s,t,d)
If $d(p_d) >\Delta$delay
Then return "There is no solution"
Repeat

$$\lambda = \frac{(p_c)-(p_d)}{(p_d)-(p_c)}$$

r:=Dijsktra (s,t, c_λ)
if $c_\lambda(r) = c_\lambda(p_c)$ then return p_d
else if $d(r) \leq \Delta$delay then p_d:= r
else p_c:= r
end repeat
end procedure

Dijkstra (s,t,c) returns a c-minimal path between the nodes s and t.

5.4 Our contribution

As mentioned before, the first step is to find all possible paths from source to destination that satisfy delay constraint and sort them from the best to the worst path by cost and delay criteria ($P_{optimal}$, $P_{secondBest}$, $P_{thirdBest}$, . . . , P_{nBest}).

In order to find the best path and all backup paths, we used the LARAC algorithm. The LARAC algorithm was explained in the previous section. We modified LARAC to store all possible paths from source (s) to destination (t) by nodes and links. Then, the LARAC algorithm was run several times. First time in order to find the optimal path. After finding the optimal path, we excluded that path (its belonging links) from a list of all possible paths. Then, we run the LARAC again in order to find the second-best path. Again, we excluded links from the second-best path and run LARAC again in order to find the third-best path. The same process was repeated until all possible paths were not considered. Algorithm stores sorted paths from the best path to the worst path. Then, the SDN controller measures link utilization on the optimal link. If link utilization is lower than 85%, only the optimal path is used and other paths are powered off. If utilization ever increases above 85%, network components on the backup path are powered on by a controller and traffic is load-balanced between the optimal and backup path.

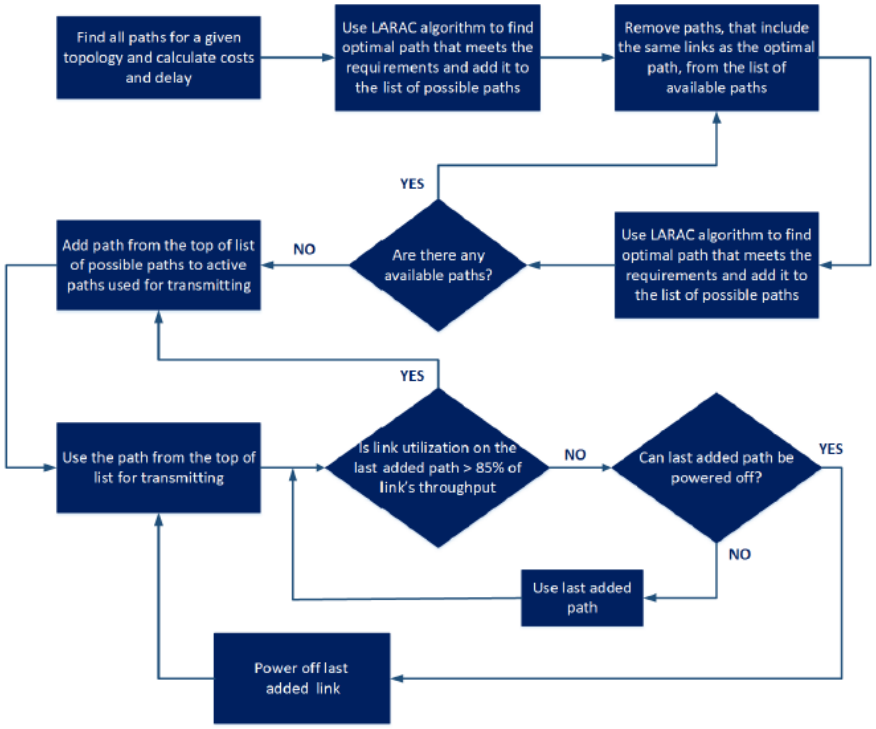

Fig. 5 Model of using a sufficient number of links in order to transmit packets without compromising QoS and reduce energy consumption in the network at the same time

If SDN controller ever detects congestion on the second-best path too, it powers on network components on the third-best path and so on. If the level of traffic decreases on the third-best path, SDN controller should check the sum of utilization on the second-best and third-best paths. If this sum is lower than 85% of the maximum throughput of the second-best path, the third-best path should be turned off. In case of a larger number of used paths, the controller should check the sum of utilizations on the two last added paths. Our model of finding and using the sufficient number of links in order to transmit packets without compromising QoS and reduce energy consumption in the network at the same time is presented in Figure 5.

6. Proof of concept

In this section, we simulated a previously considered network in Mininet. Topology, presented in Figure 6, consists of six routers connected with 10 links, controller, and six additional links for connecting the controller with each device. Each of the links has components (cost, delay). Our delay constraint was 15ms. Links in Mininet are configured to have maximal throughput of 10Gbps. Simulated topology with link costs and delays are shown in Fig. 6. In the first step, we found the optimal and the backup path. The paths were found using a script (Script1) that runs on the controller. Script1 first finds all possible links from R0 to R5 and finds the optimal path by running LARAC. LARAC calculated that with 15ms delay constraint, the optimal path is R0-R1-R4-R5.

Later in the Script1, optimal path is excluded from the list of all possible paths and LARAC runs again in order to define the backup path. The backup path by LARAC is R0-R2-R1-R3-R5. There are no other paths that satisfy requirements. Used paths are presented in Figure 7.

So, Script1 stores that $L_{(optimal)}$ are L0, L3, L9 (in the Figure 7 presented with green line), and $L_{(secondary)}$ are L1, L4, L2, L7 (in the Figure 7 presented with blue line). Other links stored as $L_{(idle)}$ (L5, L6, L8) are powered off.

Also, in Script1 are defined conditions that turn off or on $L_{(secondary)}$ depending on the used bandwidth on the links. In order to see which link is used for data transfer, we implemented the second script (Script2) on the controller that monitors and counts how many packages go through each port on the routers. Results are presented in the Figure 8. Using the IPERF tool, we sent traffic from host H1 to H2 with the bandwidth of 8 Gb/s (which is 80% of a link throughput). After we analyzed the number of tx packets (outgoing direction) on both ports on R0 (on Figure 8 datapath 0000000000000001) and the number of rx packets (inbound direction) on R1 (on Figure8 datapath 0000000000000002) and R2 (on Figure 8 datapath 0000000000000003), we noticed that the number of tx-bytes on R0's port 1 and rx-bytes on R1's port 1 constantly increase, while the number of rx-bytes on R2's port 2

remains the same. Hence, we concluded that R0 is using only L0 link. Then, in IPERF we increased sending traffic to 85% and analyzed the output of Script2 again. Now, the number of rx-bytes on R2's port 2 also started to increase. Hence, we concluded that R0 is using both links.

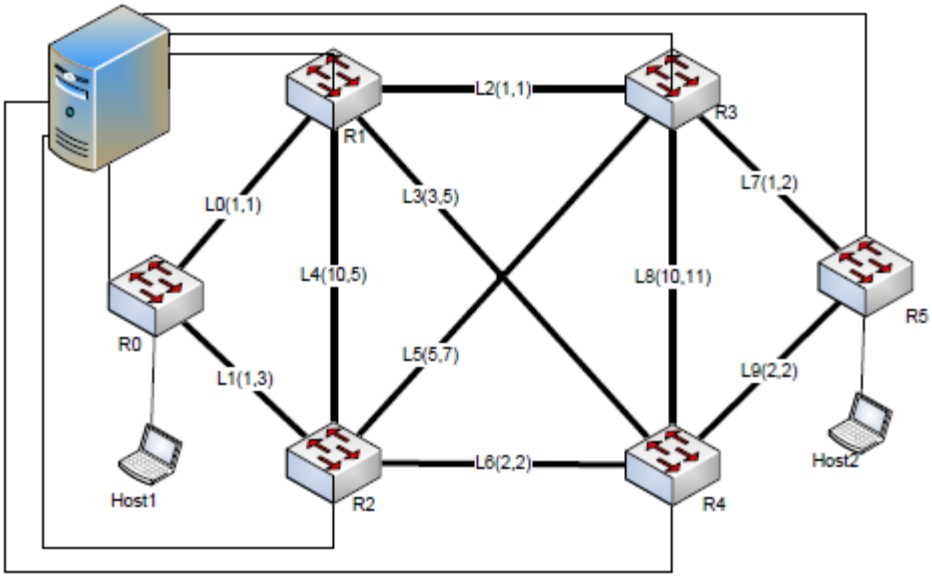

Fig. 6 Topology with link costs and delays simulated in Mininet

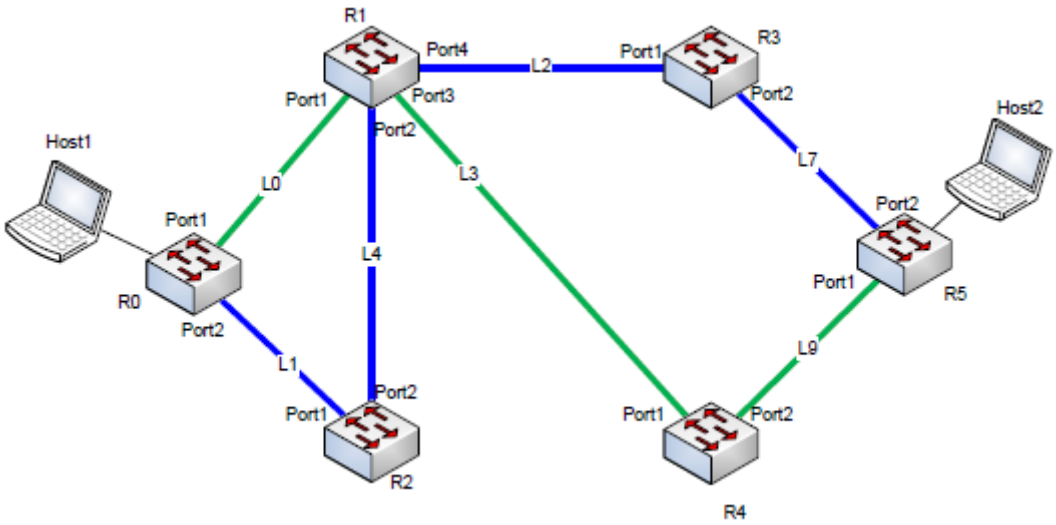

Fig. 7 Calculated paths by LARAC. Optimal path is highlighted in green, and the second-best path is highlighted in blue.

The number of receiver packets on R5's port 1 is the same as the number of sent packets on R0's port 1. Also, the number of received packets on R5's port 2 is the same as the number of sent packets on R0's port 2. This proves that there were no packet losses in communication. Using only optimal path in the beginning (when we sent 80% of maximal throughput) is the reason why the received number of packets on R5's port 1 is higher than received number of packets on R5's port 2. If this algorithm was implemented on the real network mentioned before, power savings in the best case would be 19%.

In the best case, Case I, the optimal-path link utility is lower than 85%, hence links L1, L2, L4, L5, L6, L7 and L8 could be turned off. Since Router0 only use Port1 (as part of a L0 link), Port2 on Router0 (part of link L1) could be turned off. Also, since Port2 is only configured port on Module3, the whole Module3 on Router0 could be turned off. With the same logic, Module3 could be turned off on the Router1,

Module 2 on the Router4 and Router5; and modules 2 and 3 could be turned off on Router2 and Router3.

```
EVENT ofp_event -> SimpleMonitor13 EventOFPPortStatsReply

Datapath         in-port  eth-dest           out-port  packets     bytes
---------------- -------- ------------------ -------- ----------- -----------
Datapath         port     rx-pkts    rx-bytes rx-error tx-pkts    tx-bytes    tx-error
---------------- -------- ---------- -------- -------- ----------- ----------- --------
0000000000000001 1        33         4489     0        79          9060        0
0000000000000001 2        26         3741     0        37          4707        0
datapath         port     rx-pkts    rx-bytes rx-error tx-pkts    tx-bytes    tx-error
---------------- -------- ---------- -------- -------- ----------- ----------- --------
0000000000000002 1        79         9060     0        33          4489        0
0000000000000002 2        37         4707     0        26          3741        0
0000000000000002 3        33         4489     0        79          9060        0
0000000000000002 4        26         3741     0        37          4707        0
datapath         port     rx-pkts    rx-bytes rx-error tx-pkts    tx-bytes    tx-error
---------------- -------- ---------- -------- -------- ----------- ----------- --------
0000000000000003 1        37         4707     0        26          3741        0
0000000000000003 2        26         3741     0        37          4707        0
datapath         port     rx-pkts    rx-bytes rx-error tx-pkts    tx-bytes    tx-error
---------------- -------- ---------- -------- -------- ----------- ----------- --------
0000000000000004 1        37         4707     0        26          3741        0
0000000000000004 2        26         3741     0        37          4707        0
datapath         port     rx-pkts    rx-bytes rx-error tx-pkts    tx-bytes    tx-error
---------------- -------- ---------- -------- -------- ----------- ----------- --------
0000000000000005 1        79         9060     0        33          4489        0
0000000000000005 2        33         4489     0        79          9060        0
datapath         port     rx-pkts    rx-bytes rx-error tx-pkts    tx-bytes    tx-error
---------------- -------- ---------- -------- -------- ----------- ----------- --------
0000000000000006 1        79         9060     0        33          4489        0
0000000000000006 2        37         4707     0        26          3741        0
```

Fig. 8 Statistics of sent and received packets on each port of every router

In Case I, energy savings would be 19%. In case of a greater optimal-path link utility (Case II), module 3 is turned off on R2, R3, R4 and R5. In the Case II, savings would be 8%. In Table 3 and Table 4 are presented information about power consumption in the network with powered off modules in Case I and Case II.

Table 3: Energy savings in Case I

Case I		
Device	Powered off modules	Energy Consumption per device
Router 0	Module 3	1824 W
Router 1	Module 3	1824 W
Router 2	Module 2 and 3	1524 W
Router 3	Module 3	1524 W
Router 4	Module 2	1824 W
Router 5	Module 2	1824 W
All devices in total: 10 344		
19% savings		

Table 4 Energy savings in Case II

Case II		
Device	Powered off modules	Energy Consumption per device
Router 0	/	2124 W
Router 1	/	2124 W
Router 2	Module 3	1824 W
Router 3	Module 3	1824 W
Router 4	Module 3	1824 W
Router 5	/	2124 W
All devices in total: 11 844		
8% savings		

7. Conclusion

This paper shows the use of Software Defined Network technology in order to save energy consumption without compromising Quality of Service. First, we analyzed the energy consumption of a real network. Results showed that six routers series C7606, connected with ten links, consume on average 12 744 W. In order to improve availability and reliability in networks, network administrators install redundant links and devices. The consequence of redundancy is underutilized and idle network components such as links, modules or devices.

Even though network components are idle, they still consume energy. In this paper, we implemented model with modified LARAC algorithm in order to turn off idle network components but without compromising QoS.
The idea is to find by cost and delay optimal and backup path. SDN controller checks network utility on optimal and backup paths. If the optimal path is less than 85% congested, the backup path also becomes idle. If traffic on the optimal path ever increases over 85% of link bandwidth, network components of the backup path are powered off and traffic load-balance between two paths. Other network components are powered off in order to save energy. That way, the controller guarantee QoS by preventing congestion and packets drops but at the same time saves energy. If this model was implemented in the real network, in the best case (where only the optimal path is used) energy savings would be 19%. If the optimal and backup paths were used together, energy savings would be 8%. As mentioned before, in this paper we analyzed a real IP network. Our future work will be expanded to analyzing energy consumption and saving in MPLS networks by using SDN technology.

References

[1] J. Morley, K. Widdicks, and M. Hazas, "Energy Research & Social Science Digitalisation , energy and data demand : The impact of Internet traffic on overall and peak electricity consumption," In: Energy Res. Soc. Sci., vol. 38, pp. 128–137, 2018.

[2] T. E. Klein and P. Richard, "ICT Energy Challenges , Impact and Solutions,", In: 19th International ICIN Conference-Innovations in Clouds, Internet and Networks pp. 281–286, 2016.

[3] OECD, "IEA: Electricity in a Climate-Constrained World," 2013.

[4] Dudley B., et al.:"BP Statistical Review of World Energy Statistical Review of World," In: BP Statistical Review, London, UK, accessed Aug, vol. 6, p. 2018 2019.

[5] A. Andrae, and T. Edler, "On Global Electricity Usage of Communication Technology: Trends to 2030,", In: Challenges, vol. 6(1) pp. 117–157, 2015.

[6] B. Rodrigues, A. Riekstin, G. C. Janu, and V. T. Nascimento, "GreenSDN : Bringing Energy Efficiency to an SDN Emulation Environment," IFIP/IEEE Int. Symp. Integr. Netw. Manag., vol. pp. 948-95, 2015.

[7] Brief O.S:"OpenFlow-enabled SDN and Network Functions Virtualization,", In: Open Network Foundation, vol 17, pp. 1-12, 2014.

[8] J. W. Guck, A. Van Bemten, M. Reisslein, and W. Kellerer, "Unicast QoS Routing Algorithms for SDN : A Comprehensive Survey and Performance Evaluation," In: IEEE Communications Surveys & Tutorials, vol. 20(1), pp. 388-415, 2017.

[9] G. Xu, B. Dai, B. Huang, J. Yang, and S. Wen, "Bandwidth-aware energy efficient flow scheduling with SDN in data center networks," In: Future. Generatoin. Computer. Systems, vol. 68, pp. 163-174., 2017.

[10] M. Dayarathna, Y. Wen, S. Member, and R. Fan, "Data Center Energy Consumption Modeling : A Survey," In: IEEE Communications Surveys & Tutorials, vol. 18(1), pp. 732-794,2015.

[11] K. U. N. Xie, X. Huang, S. Hao, and S. Member, "Improving Energy Efficiency via Elastic Multi-Controller SDN in Data Center Networks," vol. 4, 2016.

[12] N. Thazin, K. M. Nwe, and Y. Ishibashi, "Resource Allocation Scheme for SDN-Based Cloud Data Center Network," In: Seventeenth International Conference on Computer Applications (ICCA 2019), 2019.

[13] P. Hongyu, "QoS-guaranteed energy saving routing strategy using SDN central control for backbone networks," In: The Journal of China Universities of Posts and Telecommunications, vol. 22(5), pp. 92{100, 2015.

[14] B. G. Assefa, O. Ozkasap, "A Classification and Survey of Energy Efficient Methods in Software Defined Networking,", In: arXiv preprint arXiv:1807.08866, 2018

[15] F. Lamharras and O. Labouidya, "Comparative Analysis of Energy Saved Approaches in Software Defined Networks," In: International journal of computer science and network security, vol. 19 (1), pp.1-7, 2018

[16] K. Kurroliya, Kuldeep and Mohanty, Sagarika and Sahoo, Bibhudatta and Kanodia, "Minimizing Energy Consumption in Software Defined Networks," 2020.

[17] C. Zhang, X. Wang, A. Dong, Y. Zhao, Q. He, and M. Huang, "Energy efficient network service deployment across multiple SDN domains," In: Computer Communications, 2020.

[18] A. Alnoman and A. Anpalagan, "A SDN-assisted Energy Saving Scheme for Cooperative Edge Computing Networks," 2019 IEEE Global Communications Conference (GLOBECOM)., pp. 1–6, 2019.

[19] B. Kurroliya, Kuldeep and Mohanty, Sagarika and Kanodia, Khushboo and Sahoo, "Grey Wolf Aware Energy-saving and Load Balancing in Software Defined Networks Considering Real Time Traffic," 5th Int. Conf. Inven. Comput. Technol., 2020.

[20] B. G. Assefa and O. Ozkasap, "Link Utility and Traffic Aware Energy Saving in Software Defined Networks," In: 2017 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom), pp. 1-5, 2017.

[21] B. G. Assefa and O. Ozkasap, "Framework for Traffic Proportional Energy Efficiency in Software Defined Networks," , In: 2018 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom), pp. 1-5., pp. 2–7, 2018.

[22] B. G. Assefa and S. Member, "A Novel Metric and Method for Energy Efficient Routing in Software Defined Networks," In: IEEE Transactions on Network and Service Management, 2020, 2019.

[23] B. G. Assefa and Ö. Özkasap, "A Survey of Energy Efficiency in SDN: Software-based Methods and Optimization Models,", In: Journal of Network and Computer Applications, vol. 137, pp. 127-43, 2019.

[24] B. G. Assefa and S. Member, "IEEE Transactions on Network and Service Management RESDN : A Novel Metric and Method for Energy Efficient Routing in Software Defined Networks," In: IEEE Transactions on Network and Service Management, pp. 1–14, 2020.

[25] S. Zemmouri, S. Vakilinia, and M. Cheriet, "Let ' s Adapt to Network Change : Towards Energy Saving with Rate

Adaptation in SDN,", In: 2016 12th International Conference on Network and Service Management (CNSM), pp. 272–276, 2016.
[26] A. Fernández-fernández, S. Member, C. Cervelló-pastor, L. Ochoa-aday, and S. Member, "A Multi-Objective Routing Strategy for QoS and Energy Awareness in Software-Defined Networks," In: IEEE Communications Letters, vol. 21(11), pp. 2416–2419, 2017.
[27] J. Wang, X. Chen, C. Phillips, and Y. Yan, "Energy efficiency with QoS control in dynamic optical networks with SDN enabled integrated control plane,", In: Computer Networks, vol. 78, pp. 57-67, 2015.
[28] Z. Liu, G. Xu, P. Liu, X. Fu, and Y. Liu, "Energy-Efficient Multi-User Routing in a Software-Defined Multi-Hop Wireless Network,", In: Future Internet, vol. 11(6), p. 133, 2019.
[29] I. Šeremet, S. Hadžović, S. Mrdović, and S. Čaušević, "SDN as a Tool for Energy Saving," In: 27th Telecommunications Forum TELFOR 2019 , pp. 1–4, 2019.
[30] H. F. Salama, D. S. Reeves, and Y. Viniotis, "A Distributed Algorithm for Delay-Constrained Unicast Routing, In: Proceedings of INFOCOM'97, vol. 1, pp. 84-91.,IEEE, 1997."
[31] M. R. Garey and S. Francisco, "The NP-Completeness Column : An Ongoing Guide," , In: Journal of Algorithms, vol. 6(3), pp. 434-451, 1985.
[32] A. Puri and S. Tripakis, "Algorithms for Routing with Multiple Constraints, In: Proc. of AIPS, vol. 2. 2002..
[33] M. L. Fisher, "The Lagrangian relaxation method for solving integer programming problems," In: Management. Science. vol. 27(1), pp. 1-18, 1981.
[34] R. Z. Juttner A., Szviatovski B., Mecs I., "Lagrange relaxation based method for the QoS routing problem," In: Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No. 01CH37213), vol. 2, pp. 859-868. IEEE, 2001.

Irena Šeremet received the Bachelor and Master degrees in Transport and Communications from University of Sarajevo in 2013 and 2015, respectively. From December 2015, she has been working in BH Telecom, in Sector for IP/MPLS and Transmission Networks. Also, from 2016 she has been working as a teaching assistant at Faculty of Traffic and Communication, University of Sarajevo. Currently, Miss Šeremet is conducting research in Software Defined Networks, IP/MPLS networks, network security, and network programmability.

Sasa Mrdovic is associate professor at Faculty of Electrical Engineering, University of Sarajevo. He teaches computer networks and security courses. His main research interests include digital information security, digital forensics and next generation networks. Sasa published two security books and a number of papers in scientific journals and conference proceedings. He has been reviewing papers for various journals and conferences.

Samir Čaušević is professor at Faculty of Traffic and Communication, University of Sarajevo. From 2010 to 2019 he was a dean of Faculty of Traffic and Communication. He teaches system engineering and computer networks courses. His main research interests include environmental impact of networks and next generation networks. Samir has published several books in the field of system engineering and impact of networks on the environment. Also, he has published a number of papers in scientific journals and conference proceedings