INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2020, VOL. 66, NO. 1, PP. 77–83
Manuscript received November 15, 2019; revised January, 2020. DOI: 10.24425/ijet.2019.130268

Routing policy validation for the integrated system supporting routing in Software Defined Networks (SDNRoute)

Piotr Jaglarz, Grzegorz Rzym, Piotr Jurkiewicz, Piotr Boryło, Piotr Chołda

***Abstract*—In the article, a validation module, being a component of an integrated system supporting routing in software defined networks (SDNRoute), is proposed and thoroughly examined. The module allows for the verification of the results provided by the optimization module before these results are deployed in the production network. Routing policies are validated for their impact on the network quality parameters and against the threat of overloading (congestion).**

***Keywords*—congestion, optimization, quality of service (QoS), software defined network (SDN), validation**

I. Introduction

THE rapid advancement of broadband networks has created challenges in the design of modern network mechanisms and architectures. A non-flexible architecture and complex distributed management are making it increasingly difficult for traditional techniques to cope with growing network requirements. Software defined networks (SDNs) are perceived as a solution to these problems. Their basic advantage over traditional techniques is the centralized management architecture and the separation of the network control plane from the data plane [1]. In such networks, the central controller is responsible for managing data switching while network devices are only carrying out its commands (unlike in the traditional IP architecture, they do not make their own decisions regarding packet routing). Such the mode of operation simplifies the control over the network and allows flexible reconfiguration. Additionally, it allows to choose any routing algorithm and ensures quality of service (QoS).

The validation module proposed in this article is a component of the SDNRoute system [2], which goal is to support routing decisions in SDNs. The system is developed as a part of the LIDER project financed by the National Center for Research and Development. The system's architecture is presented in Fig. 1.

The SDNRoute system prepares routing policies for the next time window in advance. The operation takes into account the data gathered from many different sources. These encompass the following: (a) information on the current state of the network acquired from the network controller, (b) historical data, and (c) additional information provided by network applications. Then, the pieces of data are used to predict the traffic matrix in the upcoming time window.

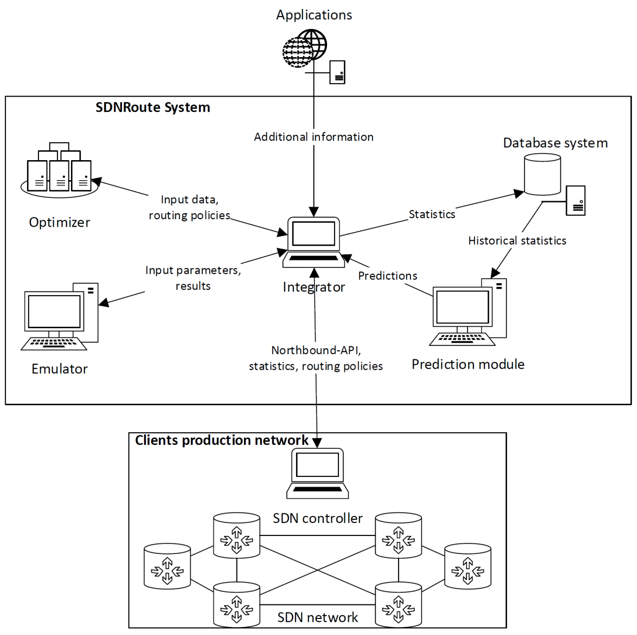


Fig. 1: Architecture of the SDNRoute system

The optimization module is responsible for preparing routing policies that will minimize the risk of network congestion and ensure effective usage of available resources. The optimization is based on the predicted traffic matrix. The SDNRoute system also allows to reduce power consumption of a network infrastructure.

Before being deployed in the production environment, the effectiveness of routing policies is verified in the validation module, which is a part of the Emulator (shown in Fig. 1). Its main task is to select the most adequate statistical measures calculated on the basis of the predicted traffic matrix. Assuming the parameters of the traffic matrix, the optimization module propose the most suitable routing policies. Subsequently, these policies are deployed in the production network by a central SDN controller. In the article, we focus on the selection of the most adequate parameters used to feed the traffic matrix, in order to obtain the most useful results.

The remaining part of the paper is organized as follows. State-of-the-art is presented in the next section. Section III

This work was funded by the Polish National Centre for Research and Development under "SDNRoute: integrated system supporting routing in Software Defined Networks" project no. LIDER/30/0006/L-7/15/NCBR/2016.

P. Jaglarz, G. Rzym, P. Jurkiewicz, P. Boryło and P. Chołda are with the Department of Telecommunications, AGH University of Science and Technology, Al. Mickiewicza 30, 30-059 Kraków, Poland.

Correspondence to P. Jaglarz, e-mail: pjaglarz@agh.edu.pl.

provides a brief description of the prediction module. Section IV introduces the optimization task. The main part of the paper, dealing with the validation module, is presented in Section V. The obtained results are carefully studied in Section VI, whereas the last section concludes the paper.

II. Results reported in related work vs. our proposal

Per-subnet routing, which was and still is the predominant approach, imposes significant limitations on implementable routing policies. However, per-flow forwarding provides more freedom for traffic engineering than per-subnet wildcard routing. This stems from the fact that due to per-flow forwarding multiple paths can be utilized simultaneously without the looping risk [3]. Nevertheless, the related requirement for keeping the system distributed poses significant challenges. That is the reason why research on distributed traffic engineering systems has been limited. The most prominent examples of such systems are MPLS-based MATE [4] and TeXCP [5], as well as flow-based FAMTAR [6].

The situation changed with the emergence of centralized per-flow software-defined networking paradigm. When the distributed control requirement is relaxed, we can simplify the system architecture by using a single central controller. Additionally, a centralized controller has a view of the whole network and, thus, possesses a comprehensive knowledge about the current and previous network state. This can result in better routing decisions. That is why the introduction of hardware and software implementing the SDN concept (i.e., OpenFlow switches and controllers) resulted in a boom of various routing and traffic engineering systems proposals. Since it is impossible to review all of them here, we only provide an overview of the most prominent examples.

Hedera [7] was proposed as a dynamic SDN flow scheduling system for data centers, aimed at going beyond ECMP limitations of the traditional IP routing. In the Hedera system, at the beginning all flows are load-balanced onto ECMP paths. Such paths are used until flows grow and meet a predefined threshold rate. After reaching the threshold, such flows (which are now considered as *elephants*) are rerouted in mid-connection onto flow-specific paths, which are computed dynamically by the controller. The path are computed using global first fit or simulated annealing algorithms.

Another influential proposal is DevoFlow [8]. Its key concept also consists in the reduction of OpenFlow overhead by focusing on significant flows. Unlike in the case of Hedera, the proponents of DevoFlow focus on implementation and scalability aspects. They propose a modified OpenFlow switch architecture in order to keep as much operation in the data plane as possible. They consider the usage of decreasing best-fit bin packing algorithm or oblivious routing precomputed according to a given demand matrix to reroute detected significant flows.

Another similar proposal is OpenSample [9], a traffic engineering system based on the rerouting of elephant flows, which are detected, similarly as in SDNRoute, by sampling packet headers on switches with sFlow. The authors claim that by using sFlow with TCP sequence numbers, the OpenSample system can achieve a low latency of measurements with a high degree of accuracy. OpenSample can be implemented without end host modifications and, unlike Hedera, it does not require the use of expensive OpenFlow counters.

Even though being most noticeable, the examples presented above are targeted and evaluated on data center networks. The SDNRoute system, which is the subject of this paper, is designed to operate in networks with irregular topologies and predictable traffic patters, such as wide area networks (WAN). SDN WAN systems are less popular as a research topic than data center systems. The most prominent papers related to SDN WANs are reports of real-world deployments in big inter-datacenter corporate networks, which prove the feasibility of proposed solutions. Nevertheless, they lack specific details.

SWAN [10] is a centralized traffic engineering solution designed by Microsoft and used to transfer data between data centers. It performs network-wide traffic engineering basing on a global network view. The path assignments for various services are calculated using a global optimization algorithm. Similarly as SDNRoute, SWAN performs the optimization on traffic matrices predicted for future time windows. It predicts the interactive traffic in 5-minutes periods. The authors of [10] claim that by doing so, the need for over-subscription is drastically reduced, and resources are used more effectively.

B4 [11] is a Google's traffic engineering solution deployed in their inter-datacenter wide area network. B4 adopts a two-level hierarchical control plane to execute the global traffic scheduling. Each switch is associated with a low-level controller, which performs distributed routing, similarly as with classical routers. The central high-level network controller dynamically reallocates bandwidth for specific applications by creating flow-specific entries and also provides dynamic rerouting in the case of link or switch failures. Unlike SWAN or SDNRoute, which perform global resources allocation based on linear programming, B4 uses fast and easy-to-implement heuristics. Google claims that it is able to drive links to near 100% utilization in practice, which is comparable to the capability of SWAN.

Surveys [12], [13] provide a general overview of SDN traffic engineering systems. SDN systems similar to SDNRoute, targeted specifically to WAN deployments, are dealt with in [14], [15].

III. Prediction

SDNRoute works independently of underlying network topologies. Thus, the prediction module has to provide information on expected traffic in the upcoming window in a time-efficient manner. In addiction, the prediction module should be able to accurately forecast traffic in contemporary networks that are characterized with highly variable traffic patterns.

The prediction algorithm and its evaluation are described in details in [16]. The proposed solution is based on the Fourier series. The proposal selects the proper coefficients for each harmonic of a trigonometric function, i.e., important harmonics are selected in order to forecast traffic, while insignificant harmonics are zeroed to avoid overestimation.

Moreover, the adaptive selection of the coefficient threshold allows to predict not only a few future points at a satisfactory time but also many more.

The achieved results have proven that this approach is not only faster than the known models (such as ARMA, ARIMA, etc.), but simultaneously achieve prediction accuracy at a comparable level. It is worth noticing that the solution is up to fifty times faster in same cases. This result is particularly important for complex systems, such as SDNRoute, where prediction is just one element, and the associated computation time with the obtained accuracy level strongly affect the speed of new routing policies enforcement in the network.

At the output of the prediction module, we get the matrix $\mathbf{P}$ of the expected traffic in the upcoming window. Each row in this matrix represents the predicted traffic volume over time for a single demand. Thus, the number of rows in this matrix is equal to the number of flow demands in the network (which we denote as D, see the details in Sec. IV). For a given demand d, the subsequent values of $h_{d1}, h_{d2}, \ldots, h_{dt}, \ldots, h_{dT}$ estimate the traffic volume at a given time (where the time samples are numbered by $t = 1, 2, \ldots, T$). In our research case, we assume 120 samples per 24-hour window ($T = 120$), so the interval between successive samples is 12 minutes. This way, we obtain the following sample matrix:

$$\mathbf{P} = \begin{bmatrix} h_{11} & h_{12} & h_{13} & \dots & h_{1T} \\ h_{21} & h_{22} & h_{23} & \dots & h_{2T} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ h_{D1} & h_{D2} & h_{D3} & \dots & h_{DT} \end{bmatrix}$$

At the input of the optimization module we can get only one value, known as the traffic volume for each demand (see Sec. IV below). Hence the aggregation of a traffic vector related to a demand into a single value to be fed in the optimization problem. This way, the demand matrix $\mathbf{H}$ is obtained as the transformation of the matrix $\mathbf{P}$ with use of a statistical function f, such as mean, ℓ^{th} percentile, etc.:

$$\mathbf{H} = \begin{bmatrix} f(h_{11}, \dots, h_{1T}) \\ f(h_{21}, \dots, h_{2T}) \\ \vdots \\ f(h_{D1}, \dots, h_{DT}) \end{bmatrix}$$

IV. Optimization

The results obtained from the prediction module are treated as the input data for the optimization module. The latter module receives aggregated data on expected network traffic. There can be up to one aggregated flow with a given traffic volume (a single value for the entire time window) between any pair of nodes in the network. This volume can be represented by various statistical measures calculated on the basis of the predicted traffic vector. For the purposes of this study, several different statistical measures were used: *mean* value (*average*), *median*, *maximum* value, as well as some *percentiles*. As a result of minimizing the objective function, the optimization module produces routing policies, which are then passed to the validation module.

The optimization module performs the task of static optimization with a multi-criteria objective function. This model uses the node-link formulation and was briefly introduced in our previous work [2]. It is given below.

Indices

$v = 1, 2, \dots, V$	the nodes in a network;
$d = 1, 2, \dots, D$	the demands (flows) between pairs of nodes;
$e = 1, 2, \dots, E$	the interfaces in the network (network arcs/links);
$k = 1, 2, \dots, K$	the linear segments approximating a convex function of link delay vs. traffic volume.

Constants

$A(e, v)$	$= 1$ if link e starts in node v; otherwise 0;
$B(e, v)$	$= 1$ if link e finishes in node v; otherwise 0;
$S(d)$	the source node of demand d;
$T(d)$	the destination node of demand d;
$H(d)$	the predicted traffic volume of a flow demand d to be satisfied;
$C(e)$	the capacity of link e;
$\Psi(e)$	the energy cost of switching on (opening) link e;
$\Xi(e)$	the fixed unit cost of energy for link e;
$F(e, k)$	the slope of the k-th linear segment approximating a convex function of link delay vs. traffic load on link e;
$G(e, k)$	the intercept of the k-th linear segment approximating a convex function of link delay vs. traffic load on link e.

Continuous non-negative variables

$x_{e,d}$	the volume of a flow satisfying demand d on link e (provides information about a fraction of a new flow that should be switched to a selected router interface);
y_e	the total capacity allocated on link e to serve the existing and new flows;
$y_{\sum}$	the total amount of resources used in the network;
w_e	the capacity utilization on link e;
$w_{\max}$	the maximum value over all link capacity utilizations;
z_e	the delay experienced by packets transmitted via link e;
$z_{\max}$	the maximum value over all link delays;
$z_{\sum}$	the aggregated link delays;
n	the total amount of energy used in the network.

Binary variables

u_e	$= 1$ if link e should be switched on, otherwise 0.

Constraints

$$\sum_e A(e, v) x_{e,d} - \sum_e B(e, v) x_{e,d} = \begin{cases} H(d) & \text{if } v = S(d) \\ 0 & \text{if } v \neq S(d), T(d) \\ -H(d) & \text{if } v = T(d) \end{cases} \quad (1)$$
$$d = 1, 2, \dots, D \quad v = 1, 2, \dots, V$$

$$y_e = \sum_d x_{e,d} \quad e = 1, 2, \dots, E \quad (2)$$

$$y_{\sum} = \sum_e y_e \tag{3}$$

$$w_e = \frac{y_e}{C(e)} \quad e = 1, 2, \ldots, E \tag{4}$$

$$w_{\max} \geq w_e \quad e = 1, 2, \ldots, E \tag{5}$$

$$z_e \geq F(e,k)y_e + G(e,k) \quad e = 1, 2, \ldots, E \quad k = 1, 2, \ldots, K \tag{6}$$

$$z_{\sum} = \sum_e z_e \tag{7}$$

$$z_{\max} \geq z_e \quad e = 1, 2, \ldots, E \tag{8}$$

$$u_e \geq w_e \quad e = 1, 2, \ldots, E \tag{9}$$

$$n = \sum_e [\Psi(e)u_e + \Xi(e)y_e] \tag{10}$$

Eq. (1) represents basic constraints which are the core of the node-link formulation. The constraints are responsible for enforcing flow conservation in intermediate nodes, and satisfying demands from the viewpoint of its source and destination nodes. Eq. (2) represents link constraints used to find an aggregated volume of traffic. According to Eq. (3), the sum of all traffic loads in the network provides a value of the total resource use. Link utilizations are found with help of Eq. (4) as a fraction of link capacity occupied by assigned flows. The maximum link utilization is determined according to Eq. (5).

In the packet networks with statistical multiplexing the larger is link utilization, the larger delays are experienced by packets. We decided to cover this aspect in order to introduce quality improvements to the system. We use a model based on the M/M/1 queue to map the traffic loads to the link delays, followed by linearization using the Fortz and Thorup approach (see [17]). The model approximates the delay with the linear segments according to Eq. (6). We can use a standard linearization procedure without necessity of applying non-continuous variables as the convexity of the problem is not disturbed. Eq. (7) and Eq. (8) determine the summarized and maximum delays, respectively.

Non-continuous variables represent the decision to switch on some links in the network. Eq. (9) enforces that the link must be switched on if only some non-zero traffic is going to be sent through it. Eq. (10) finds the total energy usage cost based on the costs of switching on the links (interfaces) and the variable (proportional) cost relative to the link loads. We assume that the energy cost of switching port on equals 180 W (watts), while every single Mb/s of load adds 0.02 W according to models given in [18] and [19].

In our approach, we use the aggregated (weighted sum) multi-objective optimization. The following criteria are included in the goal function: $y_{\sum}, w_{\max}, z_{\sum}, z_{\max}, n$. Each criterion is normalized by a coefficient $\beta(i)$, which takes the minimum objective value of a single-criterion optimization (with respect to i, where $i = y_{\sum}, \ldots$) to ensure that all criteria are equally valid. The goal function takes then the following form:

$$\sum_{i \in \{y_{\sum}, w_{\max}, z_{\sum}, z_{\max}, n\}} \frac{i}{\beta(i)} \tag{11}$$

and the final optimization task is to minimize its value.

V. Validation

It is not easy to determine which of the statistical measures used at the input of the optimization module will provide the best routing policies. By 'the best' we mean 'the most adequate from the traffic control viewpoint'. Therefore, a decision to introduce a validation module was made. Its role is to verify the policies proposed by the optimizer in terms of their impact on the quality parameters of the network. The final decision regarding the paths used in the production network is made based on the results from the validation module.

The outcome of the prediction module is a traffic matrix that specifies the estimated traffic volume over time for each demand. An example of the traffic envelope for the selected demand is shown in Fig. 2 (the `polska` network was used, for details see Section VI). On the basis of this traffic envelope, we are able to calculate all the statistical measures mentioned earlier (mean, etc.). Subsequently, the optimization process is started and key performance indicators (KPI) are calculated separately for each statistical measure being the basis for the input data.

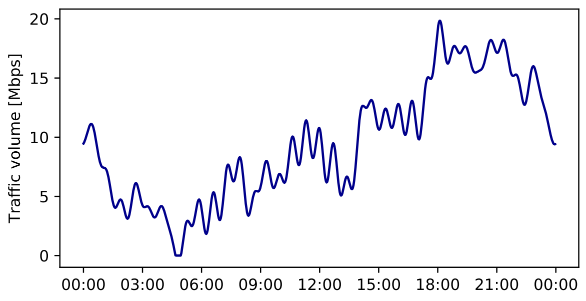


Fig. 2: Expected amount of the traffic over time (example for the Krakow–Gdansk demand)

Based on the routing policies for a given statistical measure proposed by the optimizer and the traffic volume demand between each pair of nodes in the network, it is possible to calculate the load on each link at a given moment. Furthermore, the link load can be used to determine network quality parameters. Fig. 3 shows the load on the selected link in the `polska` network and the delay it caused.

Seven different network quality indicators (KPI) were adopted for testing. The first five are directly related to the criteria used in the objective function of the optimization task. These KPIs are defined as follows: (1) the *total resource usage*, (2) the *maximum link utilization* level, (3) the *summarized delay*, (4) the *maximum delay*, and (5) the *energy consumption*. The calculations of them were based on models consistent with those assumed in the optimization model (see Section IV).

The next two indicators are related to congestion occurrence in the validated network. The KPIs are given as follows: (6) A *congestion* is assumed by us as a situation in which the link load exceeds 80% of the available bandwidth. An environment with no congestion in the network can be considered as the one with redundant resources allocated. In order to determine it, a new indicator was introduced: (7) It describes how much

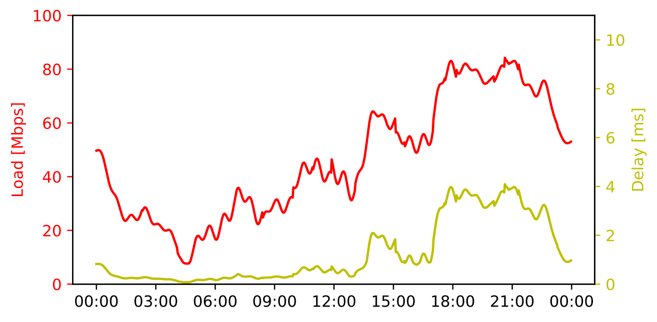


Fig. 3: Load on the Gdansk–Bialystok link and the resulting delay

the network traffic (*percentage*) could be increased without creating any congestion.

VI. Results

The research was carried out for three topologies selected from the SNDlib library [20]. The selected networks are: (a) `polska` (number of nodes $V = 12$, number of links $E = 18$, number of demands $D = 132$); (b) `nobel-eu` ($V = 28$, $E = 41$, $D = 756$), and (c) `germany50` ($V = 50$, $E = 80$, $D = 1324$). Most of the results presented in this paper concern the `polska` network. However, a summary of the results for other networks is also presented in Fig. 5 to show the repeatability of the results. The generated traffic was varying over time in accordance with the adopted daily envelope (see Fig. 2). The distribution of the length and size of network flows (defined as flows in the transport layer, the so-called *5-tuple*s) corresponds to the distribution models presented in [21].

Fig. 4 presents the values of subsequent KPIs over time for the `polska` network. The results are based on optimization using average traffic values at its input. It can be observed that the summarized resources usage and power consumption are correlated. They differ only by the constant energy cost associated with the need to switch on interfaces. Delays are heavily dependent on the load. Their value increases significantly when the maximum capacity of a given link is approached. In the figure, we can also observe that the network was overloaded (i.e., congested) three times on one of the links and this state lasted for almost three hours in total.

Table I presents a comparison of all indicators for subsequent statistical measures that are used on the input of the optimization module. The first three indicators are the averages of the results in time (means of instantaneous values from the entire time window). The next two represent the maximum value that occurred in the time window. It can be observed that the values of latency and maximum link load are much higher for optimization based on average volumes than for optimization based on maximum values. On the other hand, optimization based on maximum values causes a small increase in energy consumption, which is the result of the need to switch on additional links.

The results of the most favorable statistical measure chosen for optimization are very interesting. It turns out that the 90th

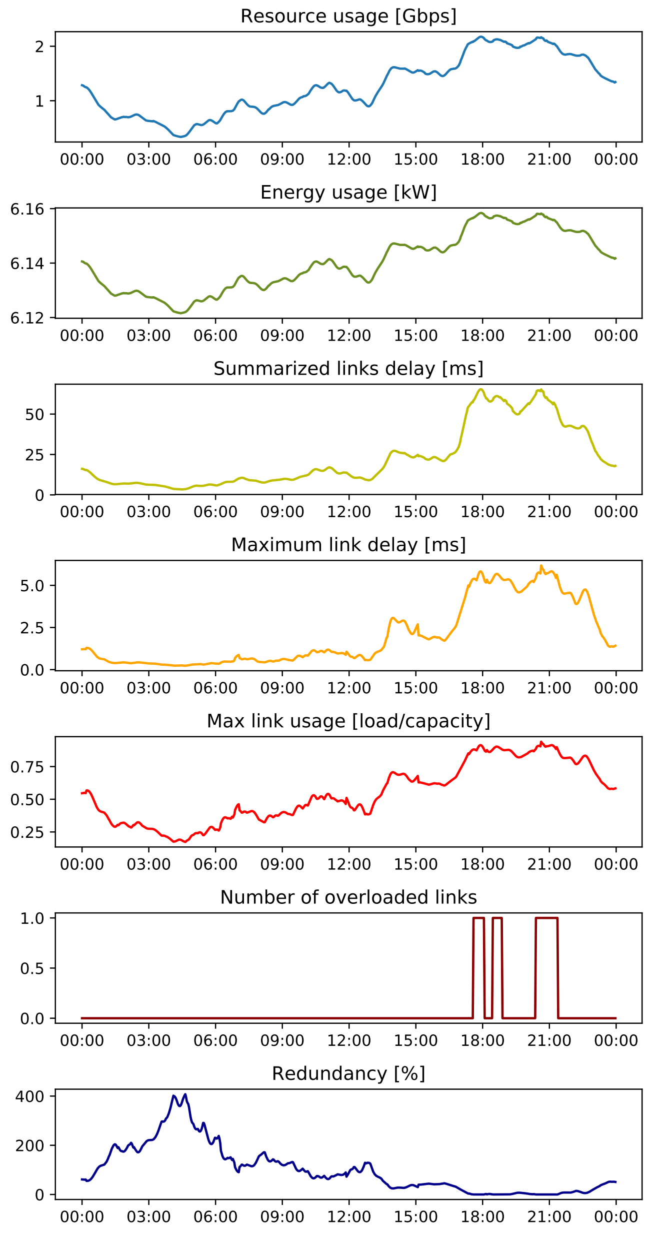
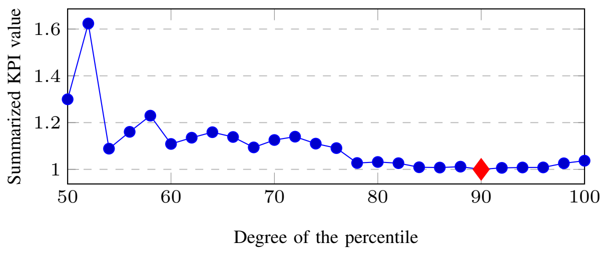


Fig. 4: Values of subsequent KPIs over time for optimization at average traffic values

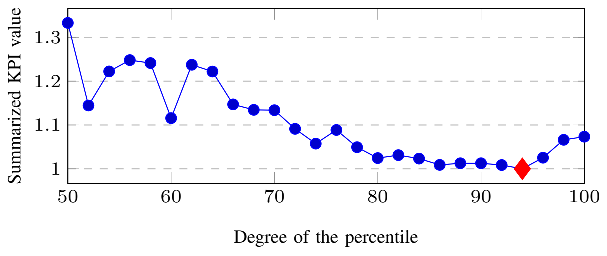
percentile is best for the `polska` network. If routing policies optimized based on volumes calculated using this percentile vale were implemented into the production network, there was no congestion in the network. Furthermore, the volume of the traffic transmitted was even slightly increased. The decrease in the value of delays and the number of occurring congestion events (overloads) is associated with the related increase in energy consumption. However, this increase is less relevant when compared with the mentioned significant network quality gains. This result is much better than for the optimization based on average traffic values, in which case there were three overloads with a total duration of 167 minutes. The median value results are even worse.

TABLE I: Values of KPIs for various statistical measures, as observed in the polska network

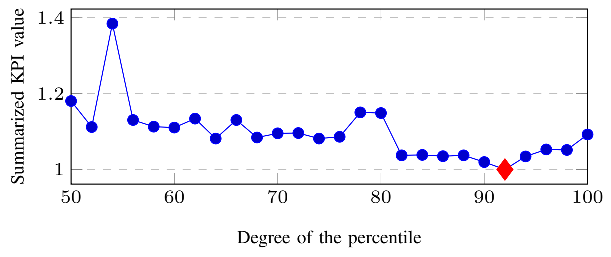
	Resource usage [Mbps]	*Energy usage [Wh]*	*Summarized delay [ms]*	*Max. delay [ms]*	*Max. usage*	*Number of overloads*	*Time of overloads [min]*	*Redundancy*
Mean	1266	6140	21.86	6.20	0.90	3	167	0%
Median	1260	5958	22.11	7.24	0.93	4	410	0%
75th percentile	1275	6502	21.44	5.60	0.85	3	44	0%
85th percentile	1278	6503	21.49	4.84	0.80	0	0	0%
90th percentile	1274	6502	21.45	4.68	0.78	0	0	2%
95th percentile	1275	6503	21.49	4.78	0.79	0	0	1%
Max.	1273	6502	21.54	4.96	0.81	2	10	0%



(a) polska



(b) nobel-eu



(c) germany50

Fig. 5: Total values of KPIs for different percentile degrees, as observed in various networks

Fig. 5 presents the total value of all indicators (KPIs) for different percentiles used at the input of the optimization module. These results were normalized in accordance to the procedure used in the optimization module before being summed. As expected, the best results for the polska network (Fig. 5a) were achieved for the 90th percentile. Slightly worse results were obtained for percentiles from the range $[78, 98]$. The results for the other two networks are very similar. In the case of the nobel-eu network (Fig. 5b), the best quality parameters were achieved for the 94th percentile, and in the range $[80, 96]$ they did not differ much from the best result. For the germany50 network (Fig. 5c), the best statistical measure passed to the optimization module appeared to be the 92th percentile. Slightly different results were obtained for the range $[82, 96]$.

The presented analysis shows that percentiles in the range $[90, 95]$ provide the best results. Their use at the input of the optimization module guarantees achieving the best quality parameters in the network. Using average or median values gives much worse results of reduction even up to 50%. The maximum value in most cases gives about 6% worse results.

VII. Conclusion

The article presents a concept of the integrated system supporting routing in software defined networks (SDNRoute) and describes in detail the functioning of the validation module. Based on the collected results, it can be concluded that in the case when routing policies are optimized for networks with dynamically changing traffic, the best option concerning the input values feeding the optimization module is to use percentiles in the range of $[90, 95]$ calculated based on the predicted traffic matrix. Due to the high amplitude of the traffic volume transferred at different times of the day, the optimization based on average values does not provide satisfactory outputs and may result in network congestion. The results obtained for median are at the similar unsatisfactory level. The research was carried out for cases where the peak hour traffic resulted in using almost as much as 100% of the available network capacity. As the future work, it is planned to verify the impact of the network load on the selection of optimal statistical measures.

Acknowledgment

This work was funded by the Polish National Centre for Research and Development under “SDNRoute: integrated system supporting routing in Software Defined Networks” project no. LIDER/30/0006/L-7/15/NCBR/2016.

References

[1] D. Kreutz, F. M. V. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, “Software-defined networking: A comprehensive survey,” *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, Jan 2015.

[2] P. Boryło, P. Chołda, J. Domżał, P. Jaglarz, P. Jurkiewicz, A. Lasoń, M. Niemiec, M. Rzepka, G. Rzym, and R. Wójcik, “SDNRoute: Integrated System Supporting Routing in Software Defined Networks,” in *2017 19th International Conference on Transparent Optical Networks (ICTON)*, July 2017, pp. 1–4.

[3] R. Wójcik, J. Domżał, Z. Duliński, P. Gawłowicz, and P. Jurkiewicz, "Loop resolution mechanism for Flow-Aware Multi-Topology Adaptive Routing," *IEEE Communications Letters*, vol. 19, no. 8, pp. 1339–1342, Jun 2015.

[4] A. Elwalid, C. Jin, S. Low, and I. Widjaja, "MATE: MPLS Adaptive Traffic Engineering," in *IEEE INFOCOM 2001*, vol. 3, 2001, pp. 1300–1309.

[5] S. Kandula, D. Katabi, B. Davie, and A. Charny, "Walking the Tightrope: Responsive Yet Stable Traffic Engineering," *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 4, pp. 253–264, 2005.

[6] P. Jurkiewicz, R. Wójcik, J. Domżał, and A. Kamisiński, "Testing implementation of FAMTAR: Adaptive multipath routing," *Computer Communications*, vol. 149, pp. 300–311, 2020.

[7] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, A. Vahdat *et al.*, "Hedera: dynamic flow scheduling for data center networks." in *NSDI'10*, 2010.

[8] A. R. Curtis, J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and S. Banerjee, "Devoflow: Scaling flow management for high-performance networks," in *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4. ACM, 2011, pp. 254–265.

[9] J. Suh, T. T. Kwon, C. Dixon, W. Felter, and J. Carter, "Opensample: A low-latency, sampling-based measurement platform for commodity sdn," in *2014 IEEE 34th International Conference on Distributed Computing Systems*. IEEE, 2014, pp. 228–237.

[10] C.-Y. Hong, S. Kandula, R. Mahajan, M. Zhang, V. Gill, M. Nanduri, and R. Wattenhofer, "Achieving high utilization with software-driven WAN," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4, pp. 15–26, 2013.

[11] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu *et al.*, "B4: Experience with a globally-deployed software defined WAN," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4, pp. 3–14, 2013.

[12] I. F. Akyildiz, A. Lee, P. Wang, M. Luo, and W. Chou, "A roadmap for traffic engineering in SDN-OpenFlow networks," *Computer Networks*, vol. 71, pp. 1–30, 2014.

[13] M. R. Abbasi, A. Guleria, and M. S. Devi, "Traffic engineering in software defined networks: a survey," *Journal of Telecommunications and Information Technology*, 2016.

[14] O. Michel and E. Keller, "SDN in wide-area networks: A survey," in *2017 Fourth International Conference on Software Defined Systems (SDS)*. IEEE, 2017, pp. 37–42.

[15] Z. Yang, Y. Cui, B. Li, Y. Liu, and Y. Xu, "Software-defined wide area network (SD-WAN): Architecture, advances and opportunities," in *2019 28th International Conference on Computer Communication and Networks (ICCCN)*. IEEE, 2019, pp. 1–9.

[16] G. Rzym, P. Boryło, and P. Chołda, "Time-efficient shrinkage algorithm for Fourier-based prediction enabling proactive optimization in Software Defined Networks," 2019. [Online]. Available: http://kt.agh.edu.pl/~rzym/publications/prediction.pdf

[17] M. Pióro and D. Medhi, *Routing, Flow, and Capacity Design in Communication and Computer Networks*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., July 2004.

[18] Y. Yang, M. Xu, D. Wang, and S. Li, "A hop-by-hop routing mechanism for green Internet," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 1, pp. 2–16, Jan 2016.

[19] P. Ruiu, A. Bianco, C. Fiandrino, P. Giaccone, and D. Kliazovich, "Power comparison of cloud data center architectures," in *2016 IEEE International Conference on Communications ICC'16*, May 2016.

[20] S. Orlowski, R. Wessäly, M. Pióro, and A. Tomaszewski, "SNDlib 1.0—Survivable Network Design Library," vol. 55, no. 3, pp. 276–286, May 2010.

[21] P. Jurkiewicz, G. Rzym, and P. Boryło, "How Many Mice Make an Elephant? Modelling Flow Length and Size Distribution of Internet Traffic," *arXiv:1809.03486*, 2018. [Online]. Available: http://arxiv.org/abs/1809.03486