Vleeschauwer, D. de, Chrysa Papagianni, D. y Walid, A. (2020). Decomposing SLAs for Network Slicing. *IEEE Communications Letters*.

DOI: https://doi.org/10.1109/LCOMM.2020.3033042

Decomposing SLAs for Network Slicing

Danny De Vleeschauwer, Chrysa Papagianni, Anwar Walid

***Abstract*—When a network slice is requested, multiple technology and/or administrative domains are invoked to ensure that the slice end-to-end service level agreement (SLA) is met. Therefore, this SLA requirement needs to be decomposed in portions that each of the domains can support. In this paper we consider a management architecture consisting of an end-to-end service orchestrator responsible for decomposing the SLA, and domain controllers that govern their respective domain. The orchestrator has no detailed knowledge of the state of the resources in each of the domains when the network slice is requested. The orchestrator is only aware of the responses of the domains to previous requests, and captures this knowledge in a risk model associated with each domain. In this study, we propose an approach for decomposing the end-to-end SLA based on the best current estimate of the risk models of all involved domains. We further describe how the risk model for a particular domain is determined (and updated) based on historical data.**

***Index Terms*—network slicing, service level agreement, risk models**

I. Introduction

A network slice is an end-to-end logical network that runs on a shared physical infrastructure, serving a particular service type with agreed upon Service-level Agreement (SLA). It is typically supported over multiple domains as it could span across multiple parts of the network (e.g. access network, core network and transport network) and could be deployed across multiple operators. Therefore, the end-to-end SLA associated with a slice, agreed with the tenant, needs to be decomposed in portions attributed to each of these domains. According to [1], decomposing an end-to-end SLA into each domain's requirements is an inevitable step in resource allocation. Thus, SLA decomposition has been acknowledged as one of the main challenges in resource allocation for network slicing.

Figure 1 shows the setting in which we tackle this problem. The management architecture consists of an end-to-end (service) orchestrator and domain controllers. The orchestrator is responsible for decomposing the SLA. Each domain controller is responsible for accepting or rejecting the part of the network slice over its domain, given the portion of the SLA it was attributed by the orchestrator. We assume that the orchestrator is not aware of the state of the resources in the domains involved at the moment the SLA needs to be decomposed. The only data available to the orchestrator is the feedback (e.g., accept or reject) from each of the domains from prior requests. Based on that data, the orchestrator constructs a risk model per domain and uses that model to decompose the SLA.

D. De Vleeschauwer and C. Papagianni are with Nokia Bell Labs, Antwerp, Belgium. e-mail: danny.de_vleeschauwer,chrysa.papagianni@nokia-bell-labs.com,A. Walid is with Nokia Bell Labs, Murray Hill, U.S.A. email: anwar.walid@nokia-bell-labs.com

Work partially funded by the EU H2020 5GROWTH Project (grant no. 856709).

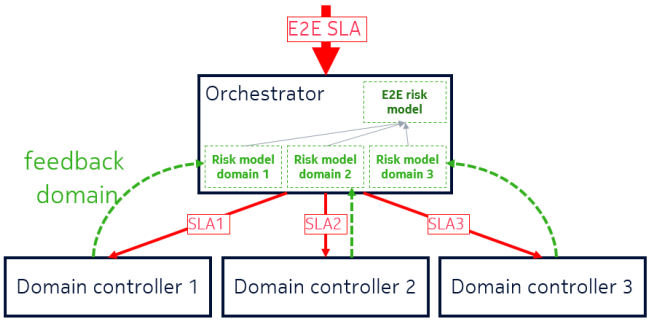


Fig. 1: Network slice management system

SLA decomposition has been previously tackled in the context of SLA management in Cloud computing, in particular for the deployment of multi-tier applications. In this sense, it involves the translation of (high-level) service-level objectives to (low-level) system-level thresholds [2] [3] [4]. Chen et al. [2] determine "component performance profiles", that describe which resources are needed to attain a performance for each particular component. The "performance model" describes how each component contributes to the overall performance. Given the performance bounds, the resources in each domain can be determined. Similarly, in the context of programmable networks and 5G systems, SLA frameworks have been proposed that decompose SLA objectives to associated policies for run-time QoS enforcement (i.e., resource properties or configurations) [5]. In our study, we simply decompose the performance bounds for each domain and leave it up to the domains themselves to allocate resources, making the problem significantly more difficult.

A. Contributions

The main contribution of this paper is that we propose an approach for SLA decomposition introducing a parameter-free risk model, appropriate for state of the art management platforms that enable and support life cycle management of network services. The parameter-free risk model per domain is created based on historical data, hence it is an offline, time-intensive process depending on its accuracy. Regarding the proposed SLA decomposition approach, although we provide no closed-form expression for the objective function of the problem, the management platform can support real time decisions given its scale. A typical SLA consists of delay, throughput and reliability constraints [6]. However, the approach that we describe in the rest of this paper is generic enough to support any type of SLA. In order to benchmark the proposed approach using the parameter-free model, we use data generated by a parameterized model that is based on simple assumptions and data generated using a simulator.

Section II presents the SLA decomposition problem as well as the parameter-free risk model, including a discussion on the complexity of the corresponding optimization problems. In section III we (i) describe a parameterized model and the simulator we used to generate synthetic data (based on analysis and simulation respectively), (ii) use the synthetic data to benchmark the parameter-free risk model and (iii) evaluate the SLA decomposition using this model. Finally, section IV presents the conclusions of this work.

II. Problem description

A. SLA Decomposition

We model the end-to-end SLA s_{e2e} as a vector of bounds for certain performance metrics. For instance, in case the SLA is composed of end-to-end delay, throughput and reliability (defined as the probability that the system will not fail for a given period of time), i.e., $s_{e2e} = (\tau_{e2e}, \theta_{e2e}, \pi_{e2e})$, this means that the network slice needs to be instantiated and operated such that its end-to-end delay τ, throughput θ and reliability π obey the constraints $\tau \leq \tau_{e2e}$, $\theta \geq \theta_{e2e}$ and $\pi \geq \pi_{e2e}$.

We further define the vector s_n as the SLA portion attributed to the n-th domain, where $n = 1, \ldots, N$ and N is the number of involved domains. The rule $s_{e2e} = G(s_n; n = 1, \ldots, N)$ that describes how to combine the SLA portions s_n to form the end-to-end SLA s_{e2e}, is known. For instance, individual delays need to be summed to form the end-to-end delay $\tau_{e2e} = \sum_n \tau_n$, the end-to-end throughput is the minimum of all individual throughputs $\theta_{e2e} = \min_n\{\theta_n\}$, while the end-to-end reliability probability is the product of the individual reliability probabilities $\pi_{e2e} = \prod_n \pi_n$.

The ability of a domain to support the portion s_n of the SLA attributed to it, is captured by a risk model where the variable a_n expresses the corresponding decision in the admission control process. The risk is defined as $-\ln \Pr[a_n = 1|s_n]$, where $\Pr[a_n = 1|s_n]$ is the probability that a request in the n-th domain with SLA s_n is accepted. When the SLA s_n is such that this acceptance probability is close to 1, the associated risk is low, otherwise it is high. Assuming that the decisions made in the domains are statistically independent, so that the end-to-end acceptance probability is the product of the acceptance probabilities of the individual domains, yields that domain risks can be summed to obtain the overall risk.

Therefore, decomposing the s_{e2e} in portions s_n allocated to each domain n can be formulated as an optimization problem under the constraint that the s_{e2e} is met, as follows:

$$\text{Minimize: } -\sum_n \ln \Pr[a_n = 1|s_n] \tag{1}$$

$$\text{Subject to: } G(s_n; n = 1, \ldots, N) = s_{e2e} \tag{2}$$

The objective of the decomposition policy is to minimize the overall risk of rejecting the corresponding request with SLA s_n for each domain n involved.

B. Determining the risk model per domain

In this subsection we scrutinize a single domain, thus we drop the subscript n for ease of notation. Likewise, in this section the term SLA signifies the SLA portion assigned to the domain.

The acceptance probability $\Pr[a = 1|s]$ in a domain emerges as follows. At the moment the request is made the infrastructure under the control of the domain controller is in a certain state ω. This state is characterised by the loads on the links and servers, the delays incurred over network hops and the calculated backup paths in case some links or routers may fail. Remark that the domain controller has detailed knowledge of this state and the impact of the decision it will make, but the orchestrator does not. The domain controller is faced with the following decision to make: given the state ω and the SLA s of the newly presented request, can the request be accepted or not. The controller has various traffic engineering tools (i.e., placement of work loads on servers, routing and scheduling traffic, calculation of back-up paths) at its disposal to make that deterministic decision. Hence, certain (s, ω) pairs will lead to acceptance and other to a rejection, i.e., $\Pr[a = 1|s, \omega] = 1$ over some (s, ω) region and 0 elsewhere. However, even though the decision of the domain controller is deterministic, the orchestrator still experiences this as stochastic, because it does not know the state ω of the infrastructure, i.e., the acceptance probability that the orchestrator sees is $\Pr[a = 1|s] = \int \Pr[a = 1|s, \omega]\Pr[\omega|s]d\omega$. The status of the infrastructure ω is non-deterministic as it is determined by the randomness of the SLAs of all previous requests and the decisions the domain made with respect to those. Often we can assume that the SLA of the new request s is statistically independent from the previous ones in which case we have that $\Pr[\omega|s] = \Pr[\omega]$.

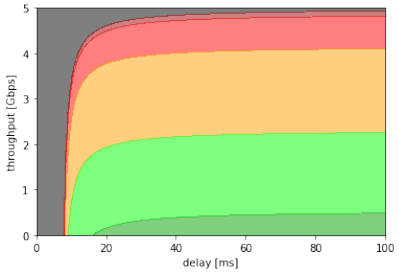
This acceptance probability $\Pr[a = 1|s]$ can be determined by observing the behaviour of the domain controller to previous requests. In particular, given K observations $(s^{(k)}, a^{(k)}), k \in \{1, 2, \ldots, K\}$ where $s^{(k)}$ is the offered SLA to the domain and $a^{(k)}$ (i.e., accept or reject) is its associated response, the acceptance probabilities for these SLA vectors $s^{(k)}$ are then determined such that the likelihood of this set of observations is maximized, as follows (assuming that the observations are statistically independent):

$$\begin{aligned}\text{Maximize: } \sum_k (a^{(k)} \ln (\Pr[a^{(k)} = 1|s^{(k)}]) \\ + (1 - a^{(k)}) \ln (1 - \Pr[a^{(k)} = 1|s^{(k)}))\end{aligned} \tag{3}$$

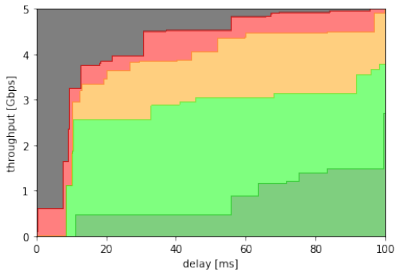
A partial order relation $\preceq$, which incorporates the notion of a stricter SLA, can be defined on the set of all possible SLA vectors s; ergo the acceptance probability has the property:

$$\Pr[a = 1|s'] \leq \Pr[a = 1|s] \text{ if } s' \preceq s \tag{4}$$

which means that a stricter SLA s' is less likely to be accepted. Notice that this is a partial, but not total order, because only a part of the SLAs vectors stand in such a relation to each other. As an example, in case the SLA is characterised by a (delay, throughput, reliability) triplet, an SLA vector $s' = (\tau', \theta', \pi')$ is stricter than an SLA vector $s = (\tau, \theta, \pi)$, i.e., $s' \preceq s$, if and only if $\tau' \leq \tau$, $\theta' \geq \theta$ and $\pi' \geq \pi$. So, in this case eq. (4) states that an SLA with lower delay, larger throughput and larger reliability requirements is less likely to get accepted.

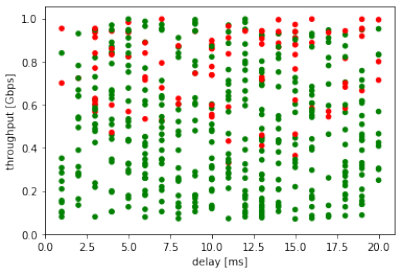


(a) Ground truth

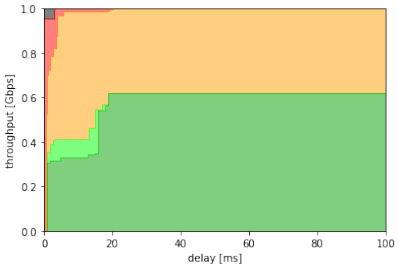


(b) Parameter-free estimation

Fig. 2: Analysis: Contour plots of the risk function



(a) Admission control: Responses to requests



(b) Parameter-free estimation

Fig. 3: Simulation: Contour plots of the risk function

Maximizing the likelihood of eq. (3) under the monotonicity constraint of eq. (4) determines the acceptance probability in the SLA vectors $s^{(k)}$. For other SLA vectors s, a lower bound and an upper bound on $\Pr[a=1|s]$ can be determined:

$$\max_{s^{(k)} \preceq s} \{\Pr[a=1|s^{(k)}]\} \leq \Pr[a=1|s] \leq \min_{s \preceq s^{(k)}} \{\Pr[a=1|s^{(k)}]\}$$

C. Complexity

We described two optimization problems; the estimation of a parameter-free risk model and SLA decomposition based on the use of the parameter-free risk model.

The estimation of a parameter-free risk model, by maximizing the objective (3) under restrictions (4), is computationally intensive. The objective function is non-linear, smooth for its arguments in the interval $(0,1)$ but tends to infinity when its arguments tend to 0 or 1. The number of constraints is proportional to K^2. Using the transitivity of the partial order relation the number of restrictions can be kept relatively small. We employ Sequential Quadratic Programming to solve the problem, and we also explicitly provide the gradient to the solver.

SLA decomposition based on parameter-free models (by minimizing the objective (1) under constraints (2) is intractable with standard methods, as the objective function is not known in all points, i.e., as explained only a lower and an upper bound is known. Therefore, the only way to find a solution is by applying exhaustive search, which makes the problem computationally hard. It is only feasible if the search space is limited by choosing a large enough, but still acceptable, granularity for every component of the SLA (e.g., delay).

III. Results

A. Data Generation

To evaluate our approach we need data. To our knowledge there is no data (publicly) available on the admission control process of networks slices with associated SLAs. Therefore we generated such data using (i) an analytic model and (ii) simulation.

1) Analytic model: In the Appendix we derive a parameterized risk model with four parameters $(\alpha, \tau_{prop}, \tau_{ref}, C)$. In particular, the acceptance probability is:

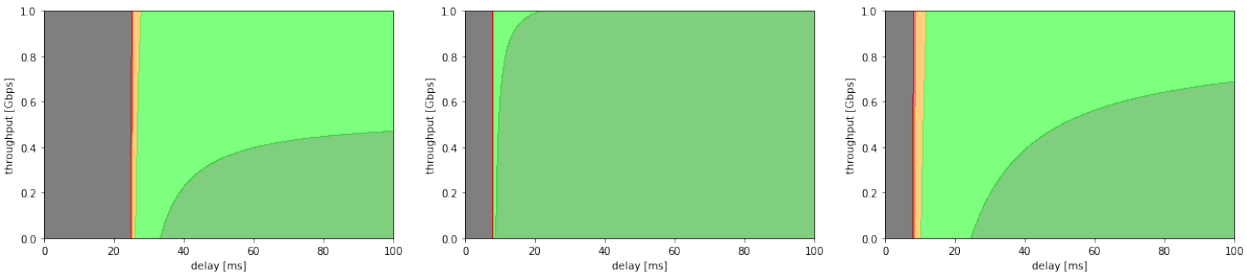
$$\Pr[a|(\tau,\theta)] = \mathrm{CDF}_\rho[\chi] \quad \text{with } \chi = \max\left\{ f^{-1}\left(\frac{\tau-\tau_{prop}}{\tau_{ref}}\right) - \frac{\theta}{C}, 0 \right\} \tag{5}$$

where $\mathrm{CDF}_\rho[\chi] = \Pr[\rho_0 \leq \chi]$ is the CDF of the instantaneous loads on the domain and function f depends on the nature of the traffic (see Table III). Eq. (5) describes the equi-probability contours, i.e., those pairs (τ,θ) for which $\Pr[a|(\tau,\theta)]$ is the same. We assume the $\mathrm{CDF}_\rho[\chi]$ takes the following form:

$$\mathrm{CDF}_\rho[\chi] = \frac{1-\exp(-\alpha\chi)}{1-\exp(-\alpha)} \quad \forall \chi \in [0,1]$$

with $\alpha > 0$. This choice promotes higher acceptance probability when the domain is lightly loaded, than when the load at the bottleneck link is close to 1. Notice that in this model the monotonicity constraint (4) is automatically satisfied.

With this parameterized model we generate data (i.e., a list of samples (τ_k, θ_k, a_k) as follows. For each sample k, the delay τ_k and the throughput θ are randomly drawn from a uniform distribution over the interval $[\tau_{prop}, \tau_{max}]$ and $[0, \theta_{max}]$ respectively. This (delay, throughput) pair is then plugged into the eq. (5) to obtain the acceptance probability $\Pr[a_k|(\tau_k,\theta_k)]$. Finally, a coin is tossed with this probability to determine whether that SLA (τ_k, θ_k) is accepted or rejected, i.e., whether $a_k = 1$ or $a_k = 0$. With this this model we can generate data for which the ground truth, i.e., the underlying risk model through its parameters $(\alpha, \tau_{prop}, \tau_{ref}, C)$, is known. This allows us to (at least qualitatively) compare the estimated risk model to this ground truth.



(a) Domain 1 (b) Domain 2 (c) Domain 3

Fig. 4: SLA decomposition: Contour plot of the risk function per domain

.

2) Simulation: For the simulations, we use an event-based simulator implemented in Java (introduced in [7], [8]), including a network slice request and data-center topology generator. The simulation setup is similar to the one presented in [8]. In this study however, an SLA (τ_k, θ_k) is associated with each request, composed of the delay τ_k and throughput θ_k bounds that are uniformly distributed between 1 to 20 msec and 70Mbps to 1Gbps respectively. Requests arrive according to a Poisson process to the domain, with an average rate of 8 requests per 100 time units, each having an exponentially distributed lifetime with an average of 1,000 time units.

TABLE I: Parameters for risk model for three domains.

domain	α	τ_{prop}	τ_{ref}	C
1	4	25 ms	1 ms	5 Gbps
2	4	8m s	0.1 ms	10 Gbps
3	4	8m s	2 ms	8 Gbps

B. Domain risk model evaluation

Analysis. Based on the parameterized model with parameters $(\alpha, \tau_{prop}, \tau_{ref}, C)$ = (4, 8ms, 1ms, 5Gbps) we generated 500 samples. The ground truth risk model is shown in Fig. 2(a). We use filled contour plots to display risk models, where the dark green, light green, orange, light red and dark red region corresponds to a acceptance probability larger than 0.99, 0.9, 0.5, 0.1 and 0.01, respectively. The grey region has an acceptance probability lower than 0.01. We estimated the parameter-free risk model with the maximum likelihood estimator of eq. (3), based on these 500 samples and the results is shown in Fig. 2(b).

The estimation of the parameter-free model (where the value shown is the average between the lower and upper bound) approximates the ground truth reasonably well taking into account the fact that the only restriction that are imposed, are the monotonicity constraints of eq. (4).

Simulation. We generated 500 consecutive network slice requests using the simulator and simulation setup described in the previous subsection, and recorded the responses as shown in Fig. 3(a), where a green dot designates an accepted request and a red dot a rejected one. Notice that there is no propagation delay in the simulator ($\tau_{prop} = 0$).

Based on these simulated observations, we can estimate the parameter-free risk model with the maximum likelihood estimator of eq. (3), and the result is shown in Fig. 3(b).

C. Decomposing an end-to-end SLA

We consider the scenario where a network slice needs to be deployed over three domains. The parameters $(\alpha, \tau_{prop}, \tau_{ref}, C)$ describing the risk model for each domain are given in Table I while the risk models are shown in Fig. 4.

In order to evaluate the efficiency of the approach, we defined the "ground truth" decomposition by partitioning an end-to-end SLA of $(\tau_{tot}, \theta_{tot})$ = (100ms, 0.5Gbps) based on these parameterized models of Fig. 4. This ground truth decomposition is given in the first row of Table II.

To show the performance of the decomposition based on parameter-free risk models, we generated K requests per domain and recorded how the domain reacted (i.e., accept or reject). Based on these requests we can determine the parameter-free risk models for each domain.

Following, using these parameter-free models, we decompose the SLA of $(\tau_{tot}, \theta_{tot})$ = (100ms, 0.5Gbps) and compare it to the ground truth decomposition. Since the risk models depend on the number (K) of requests, there is statitical variability. Therefore, we repeated each experiment 100 times for each K and calculated the mean and standard deviation. Table II shows the results for K = 250, 500 and 1000 requests. The table illustrates that the mean of the SLA decomposition based on the estimated parameter-free risk models approaches the ground truth decomposition for all values of K and that the standard deviation decreases (slowly) as K increases.

IV. Conclusions

In this paper we describe a methodology for decomposing an end-to-end SLA associated to a network slice request and assigning the corresponding SLA portions to the involved domains. The approach is applicable to any two-level network slice management system. The orchestrator is unaware of the state of each domain. At the orchestrator level we developed (i) an algorithm that partitions the SLA based on a risk model per domain and (ii) a method to gradually build such risk model based on data showing how a domain previously reacted to SLA partitions that were attributed to it. The methodology was based on a number of assumptions, the main one being

TABLE II: SLA portions determined by the orchestrator.

	metric	(τ, θ) domain 1	(τ, θ) domain 2	(τ, θ) domain 3
ground truth		(46.7 ms, 0.5 Gbps)	(15.1 ms, 0.5 Gbps)	(38.2 ms, 0.5 Gbps)
$K = 250$	mean	(46.4 ms, 0.5 Gbps)	(16.8 ms, 0.5 Gbps)	(36.8 ms, 0.5 Gbps)
	stdev	(6.4 ms, 0 Gbps)	(4.4 ms, 0 Gbps)	(6.3 ms, 0 Gbps)
$K = 500$	mean	(46.9 ms, 0.5 Gbps)	(15.4 ms, 0.5 Gbps)	(37.8 ms, 0.5 Gbps)
	stdev	(6.1 ms, 0 Gbps)	(3.5 ms, 0 Gbps)	(5.7 ms, 0 Gbps)
$K = 1000$	mean	(46.4 ms, 0.5 Gbps)	(15.1 ms, 0.5 Gbps)	(38.5 ms, 0.5 Gbps)
	stdev	(4.2 ms, 0 Gbps)	(3.1 ms, 0 Gbps)	(4.7 ms, 0 Gbps)

TABLE III: Delay curves for various traffic types.

traffic type	$K_X(s,t)$	ρ	$f(\rho)$	τ_{ref}
Poisson	$\lambda t(\exp(sS) \quad 1)$	$\frac{\lambda S}{C}$	$\frac{\rho}{(1 \quad \rho)}$	$\frac{\log(P)S}{2C}$
fractal	$\mu(st + \frac{a}{2}s^2t^{2H})$	$\frac{\mu}{C}$	$\frac{\rho^{\frac{1}{2(1 \quad H)}}}{(1 \quad \rho)^{\frac{H}{1 \quad H}}}$	$(\quad \ln(P)\frac{2a}{C}H^{2H})^{\frac{1}{2(1 \quad H)}}(1 \quad H)$

that the acceptance probabilities in the involved domains are statistically independent. As there is bound to be correlation between domains, e.g., in the case of a single operator covering (radio access network) RAN, transport and core domains, we will further investigate this aspect in future work.

APPENDIX A

This appendix justifies the parameterized model that we use to generate data and to determine a ground truth decomposition.

We first assume that the behaviour of a domain only depends on its bottleneck router. From the theory of effective bandwidth [9], the probability that the buffer occupancy Q in a node served at a rate C is larger than a value B is given by:

$$\Pr[Q > B] \approx \exp\left(-\inf_{t\geq 0}\sup_{s\geq 0}\left[s(Ct + B) - K_X(s,t)\right]\right)$$

where $K_X(s,t)$ is the cumulant generating function of the amount of traffic $X(t)$ generated in an interval of length t

$$K_X(s,t) = \log\left(E\left[\exp(sX(t))\right]\right)$$

and $E[.]$ denotes the expectation value. The maximum delay (or rather the (1-P)-percentile of the delay) in the bottleneck node is determined by identifying the value of B for which the overflow probability is P and setting the delay equal to $\frac{B}{C}$, where C is the capacity of the bottleneck node. This yields a relation expressing how the queuing delay is impacted by the amount (and the nature) of traffic, often of the form $f(\rho)\tau_{ref}$ where ρ is the load on the queue and τ_{ref} is some unit time. The cumulant generating function $K_X(s,t)$ and the resulting queuing delay dependence on the load ρ are shown in Table III for the cases of Poisson [10] and fractal traffic [11] with $\{$arrival rate λ, packet size $S\}$ and $\{$input rate μ, variance coefficient a, Hurst parameter $H\}$ respectively.

Remark that on top of this queuing delay there is also propagation delay τ_{prop} over the rest of the network.

Suppose that at the moment that the slice request arrives at the domain controller, it is characterized by an SLA comprised of a (delay, throughput) pair (τ, θ), while the bottleneck node is already loaded to a value of ρ_0. The additional throughput θ will increase the load by $\frac{\theta}{C}$ on this node and consequently introduce an additional queuing delay. The request will be accepted, if the resulting delay is smaller than the target delay for this current request and all previously accepted requests that are still active. The probability of accepting the request can be approximated by considering the delay of the request itself, without taking into account the previously accepted ones as $\Pr[A \cap B] \leq \Pr[A]$. Hence, the SLA will be accepted with probability

$$\Pr[a|(\tau,\theta)] \approx \Pr\left[\tau_{prop} + f\left(\rho_0 + \frac{\theta}{C}\right)\tau_{ref} \leq \tau\right]$$

Rewriting the inequality between the brackets in terms of ρ_0 (which can be done because function f is monotonically increasing) yields eq. (5).

REFERENCES

[1] R. Su, D. Zhang, R. Venkatesan, Z. Gong, C. Li, F. Ding, F. Jiang, and Z. Zhu, "Resource allocation for network slicing in 5g telecommunication networks: A survey of principles and models," *IEEE Network*, vol. 33, no. 6, pp. 172–179, 2019.

[2] Y. Chen, S. Iyer, X. Liu, D. Milojicic, and A. Sahai, "Sla decomposition: Translating service level objectives to system level thresholds," in *Fourth International Conference on Autonomic Computing (ICAC'07)*. IEEE, 2007, pp. 3–3.

[3] ——, "Translating service level objectives to lower level policies for multi-tier services," *Cluster Computing*, vol. 11, no. 3, pp. 299–311, 2008.

[4] N. Tiwari, "Sla aware 'on-boarding'of applications on the cloud," *cloud computing: pinnacle of IT Infrastructure democratization*, p. 27.

[5] E. Kapassa, M. Touloupou, and D. Kyriazis, "Slas in 5g: A complete framework facilitating vnf-and ns-tailored slas management," in *2018 32nd International Conference on Advanced Information Networking and Applications Workshops (WAINA)*. IEEE, 2018, pp. 469–474.

[6] GSMA, "Generic network slice template version 1.0," GSAM, Tech. Rep., 01 2019. [Online]. Available: https://www.gsma.com/newsroom/wp-content/uploads//NG.116-v1.0-1.pdf

[7] C. Papagianni, P. Papadimitriou, and J. S. Baras, "Towards reduced-state service chaining with source routing," in *2018 14th International Conference on Network and Service Management (CNSM)*, 2018, pp. 438–443.

[8] N. Torkzaban, C. Papagianni, and J. S. Baras, "Trust-aware service chain embedding," in *2019 Sixth International Conference on Software Defined Systems (SDS)*, 2019, pp. 242–247.

[9] F.P.Kelly, S.Zachary, and I. Ziedins, Eds., *Notes on Effective Bandwidth*. Oxford University Press, 1996, pp. 141–168.

[10] L. Kleinrock, *Queueing Systems*. Wiley Interscience, 1975, vol. I.

[11] I. Norros, "A storage model with self-similar input," *Queueing Systems*, vol. 16, no. 3, pp. 387–396, Sep 1994.