Measurement-based Flow Characterization in Centrally Controlled Networks

Zdravko Bozakov
Leibniz Universität Hannover / EMC Corporation
Email: zdravko.bozakov@ikt.uni-hannover.de

Amr Rizk, Divyashri Bhat, Michael Zink
University of Massachusetts, Amherst
Email: {arizk,dbhat,zink}@{engin,ecs.cs}.umass.edu

Abstract—In this work we outline a framework for measurement-based performance evaluation in SDN environments. The SDN paradigm, which is based on a strict separation of the network logic from the underlying physical substrate, necessitates a comprehensive global view of the network state. To augment the network representation, we propose mechanisms for extracting traffic characteristics from network observations which are used to derive performance metrics. Such metrics can be exploited by SDN applications to optimize the performance of SDN services. Given the bursty nature of network traffic and the well known adverse impact of this property on network performance, we propose an approach for extracting flow autocorrelations from switch counters. Our main contribution is a random sampling approach that reduces the monitoring overhead while enabling a fine-grained characterization of the flow autocorrelation structure. We analytically evaluate the impact of random sampling and demonstrate how services may use the estimated traffic properties to compute useful performance metrics.

I. INTRODUCTION

The current trend towards centralized, software defined networks (SDN) relies on a separation of the control and data planes, wherein a logically centralized controller instantiates the forwarding logic of a pool of forwarding devices. The SDN paradigm enables a fine-grained, centralized deployment of network services by providing SDN applications with a global view of the network state. To achieve this, SDN controllers must extract a detailed and up-to-date representation of the traffic carried in the substrate network.

A major advantage of SDN infrastructures is the abundant availability of computing resources in the control plane layer which is typically hosted on high performance commodity servers. We believe that these resources may be exploited to process monitoring data in order to obtain a detailed characterization of the traffic traversing the forwarding plane. To this end, we propose methodologies for a monitoring framework, embedded within the SDN control plane, which automates the extraction of flow properties through random sampling and the derivation of meaningful QoS metrics. We expect that as SDN technology matures SDN applications will increasingly rely on such metrics to improve the utilization of network resources, optimize QoS performance and minimize the need for operator intervention for the deployment of network services.

The dimensioning of network resources is a key aspect of network optimization. Service operators frequently face questions such as “How much capacity should be allocated to a specific network service?” or “What is the loss rate at some interface for a given traffic mix?”. The answer to these questions is made difficult by the strongly correlated structure of network traffic, which manifests itself as traffic burstiness. The adverse impact of this property on network performance has been shown in empirical and theoretical studies, e.g., [14], [26]. Bursty network traffic leads to an accumulation of large queues at network interfaces and consequently results in highly variable latencies. To counter these effects over-provisioning of network resources is widely used.

The goal of this work is to provide mechanisms for estimating the autocorrelation of network flows from monitoring data, which enable SDN applications to quantify the impact of mapping flows to resources in the network substrate. To this end, we rely on sampled flow counter information queried from network switches by a central monitoring entity. We use random sampling, i.e., random inter query times, as it offers several key benefits. Firstly, it reduces the monitoring traffic volume in the network. Furthermore, it mitigates the processing load at the switches. As a consequence, the flow autocorrelation structure may be estimated with a high temporal resolution without excessive stress to the switch control plane. A precise characterization of the traffic behavior over a wide range of time scales is essential for the derivation of accurate QoS metrics.

We emphasize, that while the techniques outlined in this work are applicable to any centralized monitoring architecture which provides fine-grained mechanisms for querying individual network flows, the proposed methodology fits particularly well into the SDN model. The SDN paradigm advocates a higher degree of abstraction for programming network services. Such an abstraction enables the deployment of modular, reusable control logic components. Hence, SDN applications stand to benefit from mechanisms which augment the global network view with QoS metrics which characterize the utilization of substrate network links.

Figure 1 depicts a typical SDN architecture which showcases how a service deployed on top of a controller framework may benefit from such metrics. Consider a service, implemented as an SDN application, which aims to route network flows such that the experienced QoS remains within some predefined range. The controller framework continuously collects switch statistics and extracts per-flow estimates of mean rates, variances, effective bandwidths, backlog bounds, etc.. The derived performance metrics are made available to all interested applications over a northbound SDN interface. If the SDN application detects a violation of a prescribed threshold it may choose to trigger a redirection of a specific flow along one of the available equal cost paths (dashed).
We summarize the contributions of this work as follows:

- We demonstrate how a centralized controller can extract the autocovariance of network flows by monitoring flow counters included in SDN devices.
- We develop an online, random sampling algorithm which reduces the number of counter queries necessary to estimate the covariance matrix of a flow.
- We show how the extracted metrics can be used to numerically estimate queueing performance.
- We implement the proposed approaches as a software package FlowView which includes an OpenFlow controller module.

The paper is structured as follows: Section II provides a high level overview of our approach as well as relevant backgrounds. In Section III we present our main results for extracting the covariance matrix of a traffic flow from randomly distributed flow counter observations. In Section IV we outline our approach for generating independent sample paths from the estimated flow covariance matrix. In addition, we discuss the estimation of the traffic increment distribution from collected flow data. In Section V we present a strategy for monitoring flow statistics across a large number substrate switches. We verify our approach using simulations and tested experiments in Section VI. We conclude the paper with an overview of the related work in Section VII.

II. OVERVIEW OF THE APPROACH

Our focus lies on efficient monitoring and optimization in SDN networks. We envision an SDN controller that collects fine grained flow statistics from data counters and computes corresponding characteristic flow metrics. We enhance the controller’s ability to make decisions which affect the performance of a given switch by including flow properties, specifically its burstiness, in the performance evaluation procedure. We use the flow autocovariance, respectively, autocorrelation to capture the flow burstiness. Equipped with a characterization of traffic burstiness we utilize an existing performance evaluation framework to perform a statistical QoS evaluation. The output metric computed by the controller is thus an approximation of backlog or delay distributions at a switch of interest.

In short, our approach comprises the following steps: i) random sampling of traffic flow counters at SDN switches and traffic covariance matrix estimation; ii) using the estimated traffic covariance matrix to synthesize independent traffic sample paths for a statistical performance evaluation; iii) statistical QoS evaluation based on the generated sample paths using a Monte-Carlo approach. The main contributions of our work lie in steps i) and ii) with a primary focus on i).

We rely on random sampling for the flow autocovariance estimation to minimize the monitoring traffic load that must be generated by the controller and to reduce the load on the switch control logic as each sample corresponds to a counter query. We highlight, that we make no a priori assumptions about the correlation structure of the observed traffic process. The random inter sample times have multiple advantages such as easier construction of unbiased estimators and that flow samples are taken at different time scales. A comprehensive dissection of the estimation method is given in Sect. III.

In any practical scenario, each observed traffic flow yields only a single sample path. Hence, in order to nevertheless obtain a large number of independent sample paths that exhibit the same statistical properties as the currently observed traffic process we use an approach based on the Cholesky decomposition [16]. A prerequisite for the decomposition is an estimate of the autocovariance matrix of the considered traffic flow. To this end, we utilize observations of flow counters exposed in SDN switches and collected by the SDN controller, to extract the autocovariance of traffic flows of interest.

Finally, we consider a performance evaluation framework for general traffic arrival and link service models that is known from works such as [7], [10], [20], to provide bounds on the backlog and delay distributions. As a final metric we consider bounds on the tail decay of these distributions that decay, e.g., as \( P[B(t) > b] \leq \epsilon(b) \) with \( B \) denoting the backlog and a bounding function \( \epsilon(b) \) that decays in \( b \). Essentially, constant rate buffered links with capacity \( C \) in the substrate network are characterized using Lindley’s recursion [7] leading to the following probabilistic expression

\[
P[B(t) > b] = P \left[ \sup_{0 \leq r \leq t} \{ A(t) - A(\tau) - C(t - \tau) \} > b \right]
\]

where \( A(t) \) denotes the cumulated data arrivals up to time \( t \). Note that given a bound \( P[B(t) > b] \leq \epsilon(b) \) for (1), a corresponding delay bound follows analogously [15]. Extensions for more involved link service models [6], [15] are readily obtained from the related work. From the above formulation we approximate the QoS level \( \epsilon(b) \) empirically by observing a large number of arrival sample paths and evaluating how often a sample path exceeds the function \( b + C \tau \) for any \( \tau \in [0, t] \). To this end, in our framework the controller uses a general Monte Carlo approach to simulate the queue length distribution at the switch of interest.

In the following section we provide a method for estimating the autocovariance matrix of traffic flows from randomly sampled flow counters in SDN switches.

III. ESTIMATING THE FLOW COVARIANCE MATRIX

In this section we present our main result, i.e., an analytical treatment of a flow byte counter sampling algorithm used to obtain the autocovariance matrix of a given traffic flow.
First, we discuss general aspects of flow monitoring in current SDN technologies such as OpenFlow. Thereafter, we analyze random sampling of flow counters showing the difficulty of the estimation problem. We provide a memoryless sampling algorithm with a corresponding analytical evaluation.

A. Flow Counter Monitoring

We take advantage of the flow byte counters supported in current SDN technologies such as OpenFlow [23] (i.e., flow counters and meters). Specifically, an OpenFlow controller can obtain the total amount of traffic matched by a specific flow table entry, or a group of entries, on any connected switch by generating an appropriate statistic query message using the OpenFlow protocol.

In order to estimate the autocovariance of some traffic flow, the time dependency of the flow data must be captured continuously using a sufficiently high sampling rate. Capturing, storing and processing of monitoring data is challenging, given the continuously growing network speeds and increasing number of flows. Therefore, we consider random sampling as a key technique for reducing the volume of monitoring data and maintaining scalability. Packet based sampling is widely used, e.g., in sFlow [31] technology, we assume that the traffic increment in time slot \( t \) is denoted by \( x_t \). Consequently, the autocovariance at lag \( \tau \) is given by

\[
\Sigma = \mathbb{E} \left[ xx^\top \right] = \Delta \mathbb{E} \left[ xx^\top \right] \Delta^\top = \Delta \Sigma \Delta^\top.
\]

In order to estimate the traffic covariance matrix \( \Sigma \) from a finite number of counter queries we observe the counter process over a finite monitoring duration of \( N \) time slots. We employ a sliding window vector containing the \( N_{\text{win}} \) recent counter readings of the cumulative traffic process \( \tilde{x} \) at time \( t \in [N_{\text{win}}, N] \) in order to derive the empirical covariance matrix \( \tilde{\Sigma} \) of \( \tilde{x} \). The window length is substantially smaller than the monitoring duration (\( N_{\text{win}} \ll N \)) and determines the dimensions of the estimated covariance matrix, i.e., \( \tilde{\Sigma} \in \mathbb{R}^{N_{\text{win}} \times N_{\text{win}}} \). At each time slot \( t \) we subtract \( \pi_t \) from the collected counter observations to obtain the cumulative arrivals within the sliding window interval \( N_{\text{win}} \). We denote the sliding window vector at time \( t \) as \( \tilde{w}(t) = \left( \pi_{t-N_{\text{win}}+1}, \pi_{t-N_{\text{win}}+2}, \ldots, \pi_{t-1}, \pi_t \right)^\top \subseteq \pi_t \).

Now, we obtain an estimate for each element \( \tilde{\Sigma}_{ij} \) of the covariance matrix \( \tilde{\Sigma} \) as

\[
\tilde{\Sigma}_{ij} = \frac{1}{N-N_{\text{win}}} \sum_{t=N_{\text{win}}}^N \tilde{w}(t)_{ij} \tilde{w}(t)^\top_{ij},
\]

where \( \tilde{w}(t)_{ij} \) represents a realization of the random variable \( \tilde{w}(t) \) at the \( ij \)th element of the sliding window vector \( \tilde{w}(t) \). The rationale behind the estimator (3) can be directly seen when substituting \( \tilde{w}(t) \) into (2) such that

\[
\tilde{\Sigma} = \mathbb{E} \left[ \tilde{w}(t) \tilde{w}(t)^\top \right] = \Delta \mathbb{E} \left[ \tilde{w}(t) \tilde{w}(t)^\top \right] \Delta^\top,
\]

where \( \tilde{w}(t) \) denotes a sliding window over the stationary, non-aggregated traffic process \( \tilde{x} \). It can be shown that for a sufficiently long observation duration \( N \to \infty \) the estimator \( \tilde{\Sigma} \) converges to \( \Sigma \).

Given flow counter values queried at equidistant times an estimate of the traffic covariance matrix is obtained from Eq. (3). However, in this work we aim to decrease the monitoring load on switches and controllers by reducing the query intensity. At the same time, we wish to avoid sacrificing the ability to capture flow characteristics at small time scales. To this end, we use randomly distributed inter query times to eliminate a portion of the required counter samples. We avoid periodic sampling as it was shown to introduce undesirable effects such as aliasing and phase locking [2], [30]. As a consequence of random sampling, the aggregated traffic process observed using sampled switch counters is incomplete, and the estimate of the covariance matrix must be corrected according to the distribution of the inter query times.

B. Random Inter Query Times

We consider the case where the cumulative counter process \( \tilde{x} \) is observed at random intervals, in order to reduce the total number of queries which are generated by the controller. We model the sampling strategy employed at the controller as a random vector \( a \), with elements \( a_k \in \{0, 1\} \), where a value of one indicates that a counter sample was taken at time slot \( k \).

\[1\] In the sequel, we use the terms counter sample, query, and observation interchangeably.
Inter query times are independent and identically distributed (iid) according to some distribution \( F_a \). We define the query intensity \( p \) as the fraction of time slots at which counter queries were generated by the controller. In the following, we evaluate the effects of the sampling strategy on the estimated covariance matrix. We show that the resulting distortion can be reversed in the case of geometric sampling.

Consider inter query intervals that are drawn from a geometric distribution with parameter \( p \in [0,1] \), which corresponds to the query intensity. We model the observation process obtained through the sampling vector \( a \) (or equivalently diagonal matrix \( A \)) with random inter query times as

\[
\mathbf{v}_t = a \circ \mathbf{x}_t = A \mathbf{x}_t, \quad \text{with} \quad \text{diag}(A) = a
\]

using the operator \( \circ \) to denote the element-wise (Hadamard) product of two vectors. Similar to the previous section, we denote \( \mathbf{v}^{(t)} = a^{(t)} \circ \mathbf{w}^{(t)} \) the sliding window of the \( N_{\text{win}} \) previously sampled counter observations at time \( t \). The vector elements \( v_{i}^{(t)} \) with \( l \in [0, N_{\text{win}} - 1] \) of \( \mathbf{v}^{(t)} \) are zero for all times where no sample of the flow counter was taken, i.e., \( a_{i}^{(t)} = a_{i - N_{\text{win}} + t} = 0 \). Note that when the counters are sampled randomly, the sliding window \( \mathbf{v}^{(t)} \) can only be shifted to time periods when a counter reading of the observed flow was collected at time \( k = t - N_{\text{win}} \), as this value is necessary to obtain the cumulative traffic arrivals within the sliding window interval. Specifically, the sliding window vector \( \mathbf{v}^{(t)} = (a_{t-N_{\text{win}}+1} \mathbf{w}_{t-N_{\text{win}}+1}, a_{t-N_{\text{win}}+2} \mathbf{w}_{t-N_{\text{win}}+2}, \ldots, a_{t-1} \mathbf{w}_{t-1}, a_{t} \mathbf{w}_{t}) - a_{t-N_{\text{win}}} \mathbf{w}_{t-N_{\text{win}}} \) is evaluated only if \( a_{t-N_{\text{win}}} = 1 \). The counter value aggregation algorithm is represented graphically in Fig. 2. We highlight an example of the sliding window vector \( \mathbf{v}^{(13)} \) at time \( t = 13 \). Its vector elements contain a zero whenever the aggregated data at the corresponding time cannot be calculated. The evaluated aggregation intervals for all other positions of the sliding window are illustrated below the time axis.

Clearly, the sampling process introduces a significant number of zero elements into the sliding window vector \( \mathbf{v}^{(t)} \). Hence, the resulting covariance matrix estimate is distorted and must be corrected to reverse the impact of sampling. Intuitively, the distortion is due to the fact that the sampling process generates a different number of samples for each element of the covariance matrix. Consider the elements \( \mathbf{v}_{ij}^{(t)} \) of the covariance matrix \( \overline{\mathbf{V}} \) derived from the sliding window of randomly sampled observations \( \mathbf{v}^{(t)} \) at time \( t \). Moreover, recall that the sliding window \( \mathbf{v}^{(t)} \) is evaluated only when \( a_{k} = 1 \) for \( k = t - N_{\text{win}} \). As the sampling process is independent of the observed traffic flow process we get

\[
\overline{\mathbf{V}}_{ij} = E \left[ a_{k} \frac{v_{i}^{(t)}}{v_{j}^{(t)}} \right], \quad \text{with} \quad \frac{v_{i}^{(t)}}{v_{j}^{(t)}} = a_{i}^{(t)} \frac{w_{i}^{(t)}}{w_{j}^{(t)}}
\]

\[
= E \left[ a_{k} a_{i}^{(t)} w_{i}^{(t)} a_{j}^{(t)} w_{j}^{(t)} \right] = E \left[ a_{k} a_{i}^{(t)} a_{j}^{(t)} \right] E \left[ \frac{w_{i}^{(t)}}{w_{j}^{(t)}} \right].
\]

(4)

As a consequence, for geometric, i.e., memoryless, inter sampling intervals we obtain

\[
\overline{\mathbf{V}}_{ij} = \begin{cases} E \left[ a_{k} \right] E \left[ a_{i}^{2} \right] E \left[ \frac{w_{i}^{(t)}}{w_{j}^{(t)}} \right] = p^{2} \Sigma_{ii} & : i = j \\ E \left[ a_{k} \right] E \left[ a_{i} \right] E \left[ a_{j} \right] E \left[ \frac{w_{i}^{(t)}}{w_{j}^{(t)}} \right] = p^{3} \Sigma_{ij} & : i \neq j. \end{cases}
\]

(5)

As a result, the elements of the corrected empirical covariance matrix \( \overline{\mathbf{V}} \) estimated from a sliding window of randomly sampled observations over a time interval \( N' = N - N_{\text{win}} \) are given as

\[
\overline{\Sigma}_{ij} = K_{ij} \overline{\mathbf{V}}_{ij} = \frac{K_{ij}}{N'} \sum_{t = N_{\text{win}}}^{N'} \frac{v_{i}^{(t)} v_{j}^{(t)}}{v_{j}^{(t)}},
\]

(6)

where \( \overline{v}_{i}^{(t)} \) denotes a realization of the random variable at the \( n^{th} \) element of the random vector \( \mathbf{v}^{(t)} \) and \( K_{ij} \) is a element-wise correction factor with \( K_{ii} = p^{-2} \) and \( K_{i\neq j} = p^{-3} \).

Note that from Eq. (4) it follows that a reconstruction of the sampled covariance matrix is not restricted to a geometric sampling strategy. However, the analytic evaluation of the term \( E \left[ a_{k} a_{i} a_{j} \right] \) is significantly simplified for memoryless sampling.

C. Impact of Random Sampling

In the following, we evaluate the impact of the sampling intensity \( p \) on the estimate of the covariance matrix, given a finite measurement duration \( N \). To this end, we consider a synthetic random process\(^2\) with known covariance matrix \( \overline{\mathbf{V}} \). We generate sample paths of the process and randomly sample each one using geometrically distributed inter sample intervals with sampling intensities \( p \in [0.05,1] \). Next, we

\(^{2}\)We generate fractional Brownian motion sample paths with the following parameters: \( H = 0.8, \sigma = 4, \mu = 0 \). We consider processes with non-Gaussian increments in Section IV.
estimate the empirical covariance matrix $\tilde{\Sigma}_p$ for each sampled trace as outlined above. Finally, we calculate the Frobenius norm $\|\Sigma - \tilde{\Sigma}_p\|_F$ to quantify the similarity between the analytical covariance matrix and its estimate that is derived after sampling. The experiment is repeated 50 times using different realizations of the sampling process. In Fig. 3 we depict box plots of the calculated norms. The first insight we draw from the figure is that for a fixed monitoring duration $N$ random sampling is highly beneficial. The plateaus indicate that we can save up to 80-90% of the samples without significantly degrading the quality of the estimate. The second effect seen in Fig. 3 is that the quality of the estimate rises, as expected, for longer sampling intervals $N$. We show similar results for a real-world Internet trace from [31] in Section VI.

IV. SAMPLE PATH GENERATION

In order to synthesize an arbitrary number of independent sample paths $f$ from the covariance matrix that is estimated from traffic flow observations we make use of the Cholesky decomposition [16] for square, symmetric, positive definite matrices. The factorization of the autocovariance matrix $\Sigma$ yields a unique lower triangular matrix $L$ such that $LL^\top = \Sigma$. Further, for any random vector $z$ with elements drawn independently from a standard normal distribution, the sample path vector $x_g$ generated with $x_g = Lz$ exhibits the same covariance structure as $x$ because $E[Lz(Lz)^\top] = E[LL^\top] = \Sigma$. As the elements of $z$ are drawn from a Gaussian distribution the increments of the sample path will also be normally distributed due to the additive property of Gaussian variables.

The normal distribution is frequently used to represent the increments of Ethernet LAN/WAN traffic - since the seminal paper [22] this model has been tested and adopted in numerous works, e.g., [12], [21], [26], [36]. For sufficiently large aggregation intervals the distribution of the increments of any traffic flow $x$ tends towards a normal distribution. This behavior, which arises in its simplest form as a consequence of the central limit theorem, has been examined together with other traffic properties such as long range dependency (LRD) under various conditions, e.g., in [27], [33]. For very fine timescales, empirical increment distributions were shown to deviate considerably from the Gaussian model, e.g., in [19], [21]. Hence, the use of the normal distribution is justified for cases where the minimum time slot between counter queries is sufficiently large or when the queried flow is an aggregate of many fine grained flows. Nevertheless, for traffic flows with an increment distribution which deviates significantly from the normal distribution additional steps must be taken to ensure that the synthesized sample paths accurately capture the covariance characteristics of the considered flow.

While sample paths generated using the Cholesky decomposition approach may match any observed autocovariance structure, a reproduction of the increment distributions is only feasible for the Gaussian case. Hence, for processes with non-Gaussian increments we make use of the so-called Box-Cox transform [5] which modifies the increments such that a normal distribution is obtained for a wide class of input processes. For each traffic increment $x_t$ the Box-Cox transform is defined as

$$g(x_t, \lambda) = \begin{cases} \frac{x_t^{\lambda-1}}{\ln(x_t)} & \text{for } \lambda \neq 0 \\ \ln(x_t) & \text{for } \lambda = 0, \end{cases}$$

The parameter $\lambda$ is obtained from the process observations using a maximum likelihood estimator. Further, let $g^{-1}$ denote the inverse transform $g^{-1}(g(x_t)) = x_t$ and let $g(x_t) = v_t$ denote the elements of the transformed process $v$. In the sequel, we use $v$ to estimate the corresponding covariance matrix $\Sigma_v$ using the approaches outlined in Sect. III-B. Subsequently, we calculate the Cholesky decomposition $L_vL_v^\top = \Sigma_v$ and generate independent sample paths $v_g = L_vz$ in the “normal” domain using different realizations of the random vector $z$ with elements drawn from a standard normal distribution. The subscript in $v_g$ stands for generated processes. Finally, we apply the inverse transformation $g^{-1}(v_g) = x_g$ to each sample path element to obtain sample paths $x_g$ which approximate the increment distribution of the observed flow $x$. The inverse transformation ensures that the autocovariance matrix of the generated sample paths corresponds to the autocovariance of the observed traffic process $x$, i.e.,

$$E\left[g^{-1}(v_{g_i})g^{-1}(v_{g_j})\right] = E\left[x_{g_i}x_{g_j}\right] = \Sigma_{x_{i,j}}.$$  

It can be shown using L’Hôpital’s rule that $\lim_{\lambda \to 0} \frac{\lambda^{\lambda-1}}{\lambda} = \ln(x_t)$. 

- Fig. 3: Impact of sampling intensity and sampling duration on the covariance matrix estimate. The Frobenius norm quantifies the deviation between the empirical and the analytical covariance matrices for synthetic traffic.
- Fig. 4: QQ-plots of the increment distribution of a CAIDA flow before (a) and after (b) the Box-Cox transform.
A. Evaluation of Sample Path Characteristics

We evaluated the increment distributions of traffic flows from several publicly available Internet backbone traces. Our experiments indicate that the Box-Cox transform is suitable to obtain approximately normally distributed increments. Due to space limitations we present results only from a CAIDA trace. We select a flow\(^4\) from this trace with a mean rate of 307 Mbps and evaluate it over 10 ms intervals. Figure 4 contains QQ-plots which compare the increment distribution of the flow before and after the Box-Cox transform. While plot (a) indicates a significant deviation from normality the transformed process with \(\lambda = -0.08\), represented in plot (b), is approximately Gaussian. Next, in Fig. 5 we compare the empirical CDFs \(G_\lambda(\text{data})\) of the flow aggregated over the intervals of \(\tilde{\tau}\) time slots (solid lines) to corresponding CDFs of the generated cumulative sample paths \(\tilde{X}_g\) (dashed). We find that the distributions match very well.

Finally, we compare the sample autocovariance \(\tilde{\Sigma}_x\) of the CAIDA flow to the autocovariances \(\Sigma_{x_k}\) of \(10^5\) independent sample paths \(x_g\) generated using the approach outlined above. Figure 6 depicts the autocovariance structure\(^5\) of the flow from the CAIDA trace as well as the mean of the autocovariances of the generated independent sample paths. We observe that the generated autocovariances closely follow the covariance structure of the original trace.

V. MONITORING DISTRIBUTED RESOURCES

The sampling approach outlined in the previous section enables the estimation of the autocovariance of a single flow by querying the associated byte counters on the forwarding device at random intervals. In this section we discuss aspects of monitoring multiple flows and interfaces in centralized monitoring infrastructures.

A. Monitoring Flows Across Multiple Nodes

To maintain an accurate view of the substrate state an SDN controller must monitor a number of counters across a large number of connected forwarding devices. In the following, we describe a strategy for querying multiple counters with geometrically distributed inter query times. Consider a controller that generates query messages with an allocated maximum rate \(r_c\) which is configured by the network operator. Consequently, \(\delta_C = \frac{1}{r_c}\) denotes the minimum time between two subsequent query messages issued by the controller (to any device). For a specific switch \(S\) let \(\delta_S\) denote the minimum time interval between two subsequent query messages, which corresponds to a maximum query rate of \(r_s = \frac{1}{\delta_S}\) at that switch. Note that \(\delta_S\) corresponds to the discrete time “slot” used in Section III. Typically, the rate \(r_S\) at which the switch control logic can process statistic queries is significantly lower than the rate at which monitoring queries can be generated at the controller which is hosted on high performance server hardware, hence, \(\delta_S \gg \delta_C\).

In the following we outline a multi-query strategy using the example depicted in Fig. 7, where we take into account the varying query timescales \(\delta_S\) at the switches and different query intensities to different switches\(^6\). Consider the query times of three flows \(f_{R}, f_{G}, f_{B}\) depicted in Fig. 7. The controller is configured to query each of the flows every \(\delta_{R,G,B} = 4\delta_C\) time slots, with sampling intensities \(p_R = \frac{1}{\delta_S}, p_G = \frac{r_S}{r_G}, p_B = \frac{r_B}{r_B}\), respectively. The controller allocates a fixed number of tokens every time slot (here 10) and assigns labels to the tokens such that the fraction of tokens for one flow corresponds to its intended sampling intensity. If the sum of the sampling intensities is smaller than one, then some tokens remain unlabelled. Every \(\delta_S\) the controller randomly selects one token, and generates a query message for the corresponding flow counter. The controller remains silent if an unlabeled

\(^4\)The flow is defined by the IP prefix 248.0.0.0/8 (dirB)

\(^5\)For clarity, we plot the autocovariance functions \(c_{\lambda}(\tau)\) associated with the autocovariance matrices \(\Sigma_{\lambda}\), with elements \(\Sigma_{\lambda}(i,j) = c_{\lambda}(i-j)\).

\(^6\)We ignore packet overheads and assume that each flow query is contained within a separate control message.
token is selected. From the point of view of any given flow the token selection can be regarded as a Bernoulli trial such that the inter query intervals to the corresponding counter are geometrically distributed. Next, we add two new flows to the controller’s monitoring list: \( f_Y \) with \( \delta_Y = 3\delta_C \) and \( p_Y = 1/5 \) and \( f_P \) with \( \delta_P = 4\delta_C \) and \( p_P = 1/2 \). Since not enough free tokens are available at time slot 1 the controller periodically allocates tokens for flow \( f_P \) starting at the empty time slot 3. In this scenario the selection scheme is repeated in a round robin fashion every \( 12\delta_C \) time slots. The controller may add further flows for monitoring until all available tokens are labeled.

B. Monitoring Salient Flows

We now consider the fact, that aggregate network traffic typically contains a small number of so-called elephant flows which contribute the largest share the overall traffic [39]. These dominating flows are particularly interesting candidates for monitoring. However, in order to evaluate the impact of a subset of flows on a given interface, e.g., the removal of a dominant flow from the switch, the SDN controller must consider the total traffic traversing the examined link.

We model the total traffic traversing a switch interface as a sum of statistically independent flows \( \mathbf{x}_i \), i.e., \( \mathbf{x}_{IF} = \sum_{i \in S} \mathbf{x}_i \), where the subscript IF denotes the examined interface and \( S \) denotes the set of all flows traversing that interface. From the independence of the flows, it follows that the covariance matrix of the total traffic \( \Sigma_{IF} \) is given by the sum of the covariance matrices \( \Sigma_i \) of the individual flows \( i \in S \). Therefore, given a set \( O \) of observed flows, the covariance matrix of the remaining traffic is obtained as \( \Sigma_{IF} - \Sigma_O \), where \( \Sigma_O \) is the sum of the covariance matrices \( \Sigma_i \) of flows \( i \in O \), i.e., the covariance matrix of the collection of observed flows. Hence, in order to evaluate the impact of the monitored flows on the remaining interface traffic, we let the controller query the switch port counters in addition to the flow counters.

VI. SIMULATION AND EMULATION RESULTS

In this section we use the sample paths generated using the sample autocovariance to estimate backlog bounds for a flow traversing a specific interface.

First, we validate the Monte Carlo approach from Section II based on Eq. (1) using discrete event simulations with synthetic Gaussian increment traffic processes (fractional Gaussian noise) with dimensionless parameters: mean \( \mu = 20 \), standard deviation \( \sigma = 4 \) and Hurst parameter \( H = 0.8 \). We choose this traffic type as it arises often in the traffic modeling and performance evaluation literature that examines data center and Internet core operations [12], [22], [27]. However we emphasize, that our approach is not restricted to this traffic class as discussed in Section IV. In the sequel, we present results for real-world traffic. Figure 8 shows a reference CCDF of the buffer occupancy for a simulated infinite FIFO queue with constant output rate \( C = 24 \). In the next step we generate \( 10^5 \) independent sample paths using the Cholesky decomposition approach, where the covariance matrix \( \Sigma \in \mathbb{R}^{t \times t} \) was obtained analytically using the parameters above. The resulting CCDF is plotted in Fig. 8 together with the queue simulation and a well known approximation for the buffer overflow probability from [26].

Next, we estimate backlog bounds for a publicly available WAN traffic trace from CAIDA [31]. We implemented the methods presented in the previous sections as a software package called FlowView. The software includes a POX [18] controller module which generates OpenFlow counter queries. In addition it supports reading PCAP files directly, which we use in the following in order to accurately reproduce the WAN trace. We analyze 10 minutes of a trace from [31] (dirA) with a mean rate \( \mu \approx 468 \text{Mbps} \). We used a sampling resolution \( \delta = 10 \text{ms} \) and a sampling intensity \( p = 0.1 \) resulting in an average sampling interval of 0.1 s.

We consider a scenario in which an SDN application assesses the capacity requirements for forwarding a specific network flow and the associated impact on the queue length. We estimate backlog bounds for the CAIDA flow at different switches with constant available bandwidths \( C = \{550, 600, 650, 700, 750, 1000\} \text{Mbps} \). To this end, FlowView...
extracts the Box-Cox parameter $\tilde{\lambda} = -0.24$ of the flow and estimates the autocovariance matrix $\Sigma_v$ of the transformed traffic process as described in Sections III and IV. To guarantee the positive definiteness of the estimated covariance matrix, we rely on a convex programming solver [17]. Next, sample paths are generated and finally, the Monte Carlo simulation delivers approximate buffer overflow probabilities from Eq. (1). The experiment is repeated using the sampling intensities $p = 1$ (baseline without sampling) and $p = 0.1$. The results are depicted in Fig. 9. The impact of the link utilization on queuing behavior is, as expected, evident. We find that the results obtained from random sampling closely match the results obtained without sampling while saving 90\% of the query budget.

Finally, we consider a proof-of-concept testbed experiment where we emulate a flow placement task in an SDN network. To this end we deploy an SDN slice within the GENI testbed [4] where the OpenFlow controller has to choose between two disjoint equal cost paths (in terms of utilization) for placing a 10 minute video traffic flow. As video client we use VLC with its DASH plugin [25] to support adaptive bitrate streaming. Each of the two paths carries cross traffic with different burstiness, i.e., Hurst parameter $H \in \{0.6, 0.9\}$, and hence are denoted, “low burstiness” and “high burstiness”, respectively. We fix the utilization of both paths to 0.75. In this experiment we observe that the SDN application successfully collects port stats according to the techniques described in Sect. III - V and chooses the better path for placing the video streaming flow. Note that the DASH streaming application adapts the requested video quality to the perceived network condition, hence, we show the impact of placing the video traffic flow on each of the two paths in Fig. 10. The result of this experiment shows that the quality of the video stream (in terms of bitrate) is influenced by the burstiness of the competing cross traffic.

VII. RELATED WORK

Measurements of the network state are a key aspect of global view abstraction in SDN. Efficient, measurement and management frameworks for have been proposed in, e.g., [11], [38]. Related works that provide traffic engineering and centralized resource allocation results based on OpenFlow in data centers are given in [1], [3]. A related application is known as “routing as a service” [8], respectively, routing based on quality of service metrics, see [9] for a survey. The authors of [34] develop a system for traffic matrix estimation using sampling of OpenFlow counters. Our work can be regarded as complementary as we focus on the estimation of flow auto-correlations. In [35] an SDN controller module for measuring packet loss and delays using counter queries and packet probes is presented.

To the best of our knowledge the extraction of traffic flow correlations from stochastically collected counter samples has not been studied. The presented techniques are highly applicable to resource optimization tasks in SDN and virtual networks, where logical resources must be mapped to some constrained physical substrate (e.g. [37] and the reference therein). In such optimization scenarios our methods may be used to extract delay and capacity requirements, which are typically not known in advance. A related research direction deals with identifying “heavy hitters” [13]. Knowledge of such dominant flows is important for the selection of flows to be monitored for network optimization. In [24] the authors discuss tradeoffs between estimation accuracy and required resources, and consider the impact of traffic at different time scales.

Prior to the emergence of SDN, numerous research studies have been devoted to the measurement and inference of different network performance measures and traffic statistics [2], [29], [30], [32]. The ultimate goal behind these works is the optimization of network performance, protocol enhancement as well as providing guidelines for future network dimensioning. The IETF [28] provides guidelines for collecting samples in the context of network measurements.

Performance evaluation literature provides a set of results on performance metrics such as the buffer overflow probability or the packet delay distribution given known models for the traffic and the queueing system. Depending on the analyzed model, the literature comprises exact or approximate results as well as bounds on relevant performance metrics such as the distributions of buffer occupancy and packet delay [7], [15], [20], [26]. Furthermore, the presented work differs fundamen-
tally from [30] which considers the impact of random sampling on traffic increment processes (e.g., packet sampling). The problem presented here is analytically more challenging as the samples observable through flow byte counters in forwarding devices contain traffic amounts aggregated on different time scales. The difference between the two problem formulations is directly apparent in the following expression

\[ \bar{\Sigma}_W = E[A\Delta x(A\Delta x)^\top] \neq \Delta E[Axx^\top A^\top] \Delta^\top = \Delta\Sigma_x \Delta^\top, \]

which shows that sampling after traffic aggregation is structurally different than aggregation after increment sampling.

VIII. Conclusions

In this work we provided methods for extracting traffic autocorrelations from flow counter queries generated by a logically centralized controller. We outlined an efficient sampling algorithm for estimating flow autocovariances from queries sent at random time intervals and simulated the impact of the used sampling distribution. The use of random sampling enables a fine grained approximation of the flow autocorrelation structure while limiting the load on the switch control logic. To simulate the queueing performance of the desired flows we applied a Monte Carlo approach. To this end, we presented methods for generating an arbitrary number of independent sample paths that reproduce the correlation structure of the observed flow and the distribution of its increments. Finally, we outlined and showed an example of a strategy for monitoring large numbers of devices and flows in centralized environments. This framework augments the SDN global view through enabling SDN services to exploit certain control plane performance metrics to optimize resource utilization and improve perceived QoS.

REFERENCES

