Evaluation of an End-to-End Delay Estimation in the Case of Multiple Flows in SDN Networks

Huu-Nghi Nguyen
Univ Lyon, UCB Lyon 1,
ENS Lyon, Inria, CNRS,
LIP UMR 5668 - Lyon, France
huu.nguyen@ens-lyon.fr

Thomas Begin
Univ Lyon, UCB Lyon 1,
ENS Lyon, Inria, CNRS,
LIP UMR 5668 - Lyon, France
DIVA Lab, University of Ottawa,
thomas.begin@ens-lyon.fr

Anthony Busson
Univ Lyon, UCB Lyon 1,
ENS Lyon, Inria, CNRS,
LIP UMR 5668 - Lyon, France
anthony.busson@ens-lyon.fr

Isabelle Guérin Lassous
Univ Lyon, UCB Lyon 1,
ENS Lyon, Inria, CNRS,
LIP UMR 5668 - Lyon, France
isabelle.guerin-lassous@ens-lyon.fr

***Abstract*—Though SDN (Software Defined Network) provides the executive building blocks for programming data-plane appliances, controller decisions must be grounded in an accurate outlook on the network topology and performance. In this context, we focus on the possibility of providing accurate measurements for the end-to-end (E2E) delay in SDN networks. In practice, like many variable quantities, a good description of the E2E delay requires characterizing its first two moments, i.e., expectation and variance. We propose to estimate the E2E delay by making use only of measurements collected locally on each node of the network. We extend a procedure that has been proposed to estimate the E2E delay in the case of one flow to handle the case of multiple competing flows. We compare its accuracy using several scenarios, with different types of traffic following real traces, different topologies and bandwidth. Also, an analysis of the computational and networking costs of our solution is proposed.**

I. Introduction

Software Defined Networking (SDN) is a disruptive technology that thoroughly redefines the way computer networks are managed. SDN has two key features. First, within an SDN, the control plane is decoupled from the data plane. Second, the control plane of an SDN enables a controller to directly control data-plane appliances (e.g., routers and switches). Controllers decide rules describing how data-plane appliances handle packets and flows, and then communicate them to the data-plane appliances using an Application Programming Interface (API) such as OpenFlow [1]. SDN computer networks can leverage this unified architecture in order to have a simple, efficient and scalable management of their resources [2].

Though SDN provides the executive building blocks for programming data-plane appliances, controller decisions must be grounded in an accurate outlook on the network topology and performance. For instance, routing decisions for flows with stringent requirements of Quality of Service (QoS) could be supported by an estimate of the end-to-end (E2E) delay. Other potential examples include admission control and load balancing. Hence, SDN controllers would highly benefit from having at their disposal a performance dashboard that they can query prior to take a decision. This dashboard will typically include metrics such as CPU utilization at virtual switches, current number of active flows at virtual switches, consumed throughput on link, and E2E delay.

In this paper, we focus on the possibility of providing accurate measurements for the E2E delay to the SDN controller. Broadly speaking, the E2E delay refers to the time taken for a packet to be transmitted across a network from its source to its destination. For flows like interactive video or audio streams, the E2E delay is a critical parameter [3] as opposed to the round-trip time (RTT), which is of little interest in these cases.

Because packets taking the same path across a network typically may experience different amounts of delay, one should consider statistical metrics to characterize the E2E delay of a flow. In practice, like many variable quantities, a good description of the E2E delay requires characterizing its first two moments, i.e., expectation and variance. The expected value (or mean value) is the most important parameter to consider. However, estimating its variance (or, similarly, its standard deviation) can be of much interest because it enables a better understanding on the variation of the E2E delay, as well as probabilistic bounds on the range of possible values (e.g., Chebyshev's inequality).

Because the E2E delay is a one-way delay, its direct measurement requires two synchronized clocks at different locations. Unfortunately, this direct measurement approach would lead to inaccuracies due to the problem of clock synchronization, unless expensive GPS device are used [4], [5], [6]. Moreover, most of the proposed solutions for estimating the E2E delay insert probing packets that may affect the performance of the existing traffic [7]. Alternately, we propose to estimate the E2E delay of a given path across the network by making use only of measurements collected locally on each node of the network. We assume that each node can monitor its buffers so that the SDN controller has local delay measurements for each node interface.

Given measurements of the delay at each node composing the path under study, the expected value of E2E delay is simply obtained by adding them all together. One could attempt to apply a similar approach to approximate the variance, or better yet, the whole distribution of the E2E delay. Though direct measurement approach would work if the delays at each node of the path were independent, it will lead in practice to large errors, usually underestimating the actual value of the variance. We illustrate the inapplicability of this

approach by calculating the variance (resp. the distribution) of the E2E delay by summing up the delay variances (resp. convoluting the delay distributions) measured at each node composing a path of 5 nodes (similar to that considered in Section III). The approximate variance was found to be almost three times smaller than the exact one found by sampling the path. As for the whole distribution of the E2E delay, the approximate solution significantly differs from the exact one (see Fig 1). This example demonstrates that independency between the local delays composing a path cannot be assumed when estimating the variance of the E2E delay. Their strong correlation compels a more holistic approach.

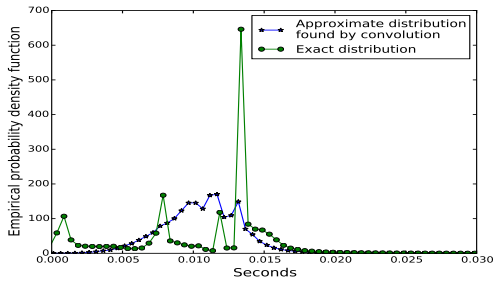


Fig. 1. Approximating the distribution of E2E delay by convoluting the delay distributions found on each node generally leads to poor results.

In a previous paper [8], we described a procedure to estimate the first two moments of the E2E delay in case of a single flow transmitted on a path. In a separate work [9], we proposed another procedure that pursues the same objective but using a different approach. In this paper, we extend the former procedure to handle the case of multiple competing flows. We compare its accuracy with that of the latter procedure using several scenarios, as well as their computational and networking costs. The rest of this paper is organized as follows. In Section II, we describe the proposed solution to estimate the first two moments of the E2E delay. Numerical results and comparison to another existing solution are reported in Section III. Section IV concludes this paper.

II. Variance of the End-to-End delay

A. Assumptions and definitions

Considering that the SDN controller attempts to monitor the E2E delays for different flows in the network, we propose a procedure to estimate their mean and variance. Without loss of generality, we describe our procedure to obtain this quantity for a particular flow, hereafter named the *primary flow*, and denoted by f_1. Note that in this paper, our definition of a flow refers to a set of packets entering and leaving the network at the same nodes, and, importantly, taking the same path. Thus, this flow can aggregate multiple IP flows, which happen to take the same path on their way to their destination.

We refer to the *primary path* as being the set of nodes and links traversed by the primary flow. We denote by N the number of nodes composing the primary path, numbered from 1 to N. The link between node k and node $(k+1)$ is characterized by its transmitting capacity C_k (in Mbps), and its propagation time R_k (in sec). Packets queued in each node and waiting for transmission are processed according to a First-In-First-out discipline.

Beside the primary flow, *secondary flows* may come across the primary path. Their packets compete with those of the primary flow in the access to the links. We use $f_{k,l}$ to denote a secondary flow entering the primary path at node k and leaving it at node l (with $1 \leq k < l \leq N$). Figure 2 illustrates two examples of primary path with several flows, including f_1.

From a practical standpoint, our proposed method requires some information from the nodes composing the primary path. First, each node knows the transmitting capacities of the links composing the primary path. Second, nodes are able to measure the size, inter-arrival time and the waiting time (time queued in the buffer of the node) of incoming packets. Note that this latter quantity can be obtained using a single clock. Finally, based on the IP source and destination addresses, nodes are able to distinguish the different flows from each other.

B. Proposed solution by decomposition

We now describe our solution to estimate the variance of the E2E delay for the primary flow f_1. Note that we do not address the case of the expectation of the E2E delay (i.e., its mean value) because its best estimate is simply obtained by summing all together the mean delays measured on each node composing the primary path. However, the variance of the E2E delay contains a complex covariance term, which captures the correlation between the local measurements. As we rely on a decomposition approach to approximate its value, we refer to our solution as the *Decomposition method*.

Let D be the random variable representing the E2E delay of f_1, and D_k be the random variables accounting for the times spent by packets of f_1 between nodes k and $k+1$. Each D_k has 3 main components:

- The waiting time, W_k, corresponding to the amount of time packets are queued in node k prior to their transmission;
- The transmission time, S_k, expressing the time needed to transmit packets on the link between nodes k and $k+1$ once their waiting times are over;
- The propagation time, R_k, on the link between nodes k and $k+1$.

Note that the processing times, including the times to process the packet headers and to look for the routing rules, can be neglected in comparison with the other quantities. The prominent framework to process packets, DPDK library, enables the processing of a packet in less than 80 CPU cycles [10]. Besides, propagation times are constants, and hence their variances are zero.

By definition, the E2E delay for f_1 can be expressed as: $D = \sum_k D_k$. Applying the variance operator on this equation,

we obtain:

$$\mathbb{V}[D] = \mathbb{V}[\sum_{k=1}^{N-1} D_k] = \mathbb{V}[\sum_{k=1}^{N-1} (W_k + S_k)] \quad (1)$$
$$= \mathbb{V}[\sum_{k=1}^{N-1} W_k] + \mathbb{V}[\sum_{k=1}^{N-1} S_k] + \sum_{k=1,l=1}^{N-1} \mathrm{Cov}(W_k, S_l).$$

We now explain how we approximate each of these terms. Let denote by $\mathrm{P}_k^{f_1}$ the random variable representing the packet size for the flow f_1 at node k. The transmission time on the k-th link is given by $S_k = \frac{\mathrm{P}_k^{f_1}}{C_k}$. The term $\mathbb{V}[\sum_{k=1}^{N-1} S_k]$ is computed by node N such as:

$$\mathbb{V}[\sum_{k=1}^{N-1} S_k] = \left(\sum_{k=1}^{N-1} \frac{1}{C_i}\right)^2 \mathbb{V}[\mathrm{P}_N^{f_1}]. \quad (2)$$

Using the definition of the covariance, $\mathrm{Cov}(W_k, S_l)$ can be estimated by each node k with the following estimation:

$$\mathrm{Cov}(W_k, S_l) \approx \frac{1}{C_l}\left(\mathbb{E}[W_k \mathrm{P}_k^{f_1}] - \mathbb{E}[W_k]\mathbb{E}[\mathrm{P}_k^{f_1}]\right). \quad (3)$$

At this point, the only quantity left unknown is $\mathbb{V}[\sum_{k=1}^{N-1} W_k]$. Its value is more complex as it contains the covariance capturing the correlation between the waiting times among the different nodes. By expanding it in different terms, we get:

$$\mathbb{V}[\sum_{k=1}^{N-1} W_k] = \sum_{k=1}^{N-1} \mathbb{V}[W_k] + 2 \sum_{k=1,l=2,k<l}^{N-1} \mathrm{Cov}(W_k, W_l). \quad (4)$$

The variance terms appearing in Eq. (4) can be calculated locally by each node using the classical estimator for variance. However, it is not possible to obtain a simple empirical estimator of the covariance terms. It would suppose to know the waiting times at the different nodes for the same packets, which requires a complex synchronization between nodes.

Our solution relies on a recurrent function that allows a given node k to infer the waiting time that a packet is expected to experience in subsequent nodes l (with $l > k$). Arguments of this function include the packet sizes, the link capacities as well as the time elapsing between two successive arrivals. For the sake of clarity, in the following, quantities being found by this function are marked with a "hat" (e.g., $\hat{w}_{k+1}^m$).

To explain the inner-working of the recurrent function, let us consider the m-th packet of the primary flow at node k. We denote its size by p_k^m, and its transmission time on the k-th node by s_k^m. These quantities are estimated locally. Then, node k infers $\hat{I}_{k+1}^m$ the inter-arrival time between packets $m-1$ and m at the next node $k+1$. Also, node k estimates $\hat{w}_{k+1}^m$ the waiting time of packet m at node $k+1$, and $\hat{s}_{k+1}^{m-1} = \frac{p_k^{m-1}}{C_{k+1}}$ the service time of packet $m-1$ at node $k+1$. Eq. (5) details the recurrent function.

$$\begin{cases} \hat{I}_{k+1}^m & = s_k^m + \max\{0, I_k^m - w_k^{m-1} - s_k^{m-1}\} \\ \hat{w}_{k+1}^m & = \max\{0, \hat{w}_{k+1}^{m-1} + \hat{s}_{k+1}^{m-1} - \hat{I}_{k+1}^m\}, k \geq 1. \end{cases} \quad (5)$$

Any node k can apply this formula recursively in order to forecast the waiting time on a subsequent node $(k+i)$, i.e., $\hat{w}_{k+i}^m$.

We now discuss the applicability of this recurrent function. In fact, the estimates returned by the recurrent function may differ from the actual waiting times for two main reasons. First, in order to keep the calculations tractable, it only considers the primary flow, and hence explicitly neglects the secondary flows. By doing so, we assume that the additional delays caused by secondary flows are sufficiently independent from the waiting time in queue k so that they do no significantly affect the covariance terms. However, note that we implicitly do take into account the influence of secondary flows when we compute the variance of the waiting times, W_k. Second, we do not consider the actual finite size of buffers in Eq. (5), thus neglecting potential packet losses. However, packet losses tend to occur rarely in wired computer networks due to their general oversizing [11]. Nonetheless, based on our numerical results (including different types of traces, network topologies, buffer sizes, and levels of load), it appears that this recurrent function provides fair predictions for our purpose.

Overall, every node k of the primary path can calculate $\mathbb{V}(W_k)$, $\mathrm{Cov}(W_k, S_l)$ (with $l > k$), as well as $\mathrm{Cov}(W_k, W_l)$ (with $l > k$) using the proposed recurrent function. Node N can compute $\mathbb{V}[\sum_{k=1}^{N-1} S_k]$. Then, they periodically communicate these values to the SDN controller that is in charge of summing them in order to obtain an estimate of the variance of the E2E delay of flows f_1 as given by Eq. (1).

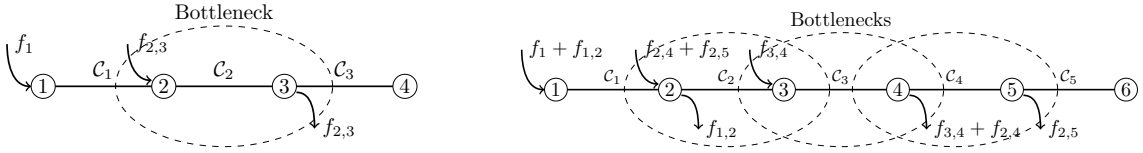
III. Numerical results

In this section, we assess the goodness of our proposed solution, i.e., *Decomposition method*, to approximate the second moment of the E2E delay. However, instead of considering the variance, we focus on the standard deviation of the E2E delay, denoted by σ hereafter. Note that the latter is simply the square root of the former. Dealing with the standard deviation is often more convenient from a practical standpoint as it is expressed in the same units as the mean.

The results provided by our Decomposition method are indexed by *Dec*. For the sake of completeness, we compare its found values for the standard deviation of the E2E delay, i.e., σ_{Dec}, with those delivered by three other means:

- σ_{Exact}, whose values are found by using the classical empirical estimator for the standard deviation on the actual E2E delay of packets. Though awkwardly implementable in a real network, this Oracle is used as the reference and allows us to estimate the error of the different methods and to compare them objectively;
- $\sigma_{Trivial}$, whose values are simply obtained by neglecting the correlation between the waiting times and between the transmission times. In other words, covariance terms in Eq. (2) and in Eq.(4) are set to zero;
- σ_{Cond}, whose values are found using the solution described in [9], which relies on a conditional approach.

We implemented these four solutions in the discrete-event simulator NS-3.



(a) Scenario 1: 2 flows and 4 nodes.

(b) Scenario 2: 5 flows and 6 nodes.

Fig. 2. Considered scenarios.

A. Scenarios

To illustrate the behavior of our solution, we consider two different scenarios. Scenario 1 represents a small topology wherein the primary path is made up of four nodes, and only one secondary flow competes to access some links. Fig. 2(a) illustrates this first scenario. In our second scenario, the primary path consists of 6 nodes and, aside the primary flow, four secondary flows cross the path. Fig. 2(b) depicts Scenario 2. Each scenario can operate in two modes: low-capacity and high-capacity. In the *low-capacity* mode, the link capacities are expressed in Mbps, whereas when we consider the *high-capacity* mode, capacities are thousand fold larger, and thus expressed in Gbps. More precisely, the link capacities are as follows: $C_1 = 8, C_2 = 5, C_3 = 3$ for Scenario 1, and $C_1 = 6, C_2 = 4, C_3 = 7, C_4 = 3, C_5 = 5$ for Scenario 2.

Secondary flows fill the primary path at a fixed and constant (in average) rate. In Scenario 1, $f_{2,3}$ is set to 2 Mbps (resp. Gbps) when considering the low-capacity (resp. high-capacity) mode. As for Scenario 2, $f_{1,2}$, $f_{2,4}$, $f_{2,5}$ and $f_{3,4}$ are configured to with a mean rate of 1, 0.8, 0.2 and 1.5 Mbps (resp. Gbps) for the low-capacity (resp. high-capacity) mode. Unless stated otherwise, all buffers are of size $15,000$ bytes when the low-capacity mode is on, and $75,000$ bytes for the high-capacity mode.

We now discuss the statistical profile of the traffic used to generate the packets of the flows (primary and secondary flows alike). Packet sizes and inter-arrival times are drawn using one of the two real traces at our hand. Our first trace was captured on the campus of the Stuttgart University (Germany) [12]. The part of the trace we used spans over four hours of communications, from 6 p.m to 10 p.m, and contains 44 millions packets with a maximum workload measured at 60 Mbps (measured in one second). The second trace was downloaded from the Center for Applied Internet Data Analysis website (CAIDA) [13]. This trace originates from the Equinix exchange points in Chicago. It is composed of several files, each one contains one minute of traffic that represents between 15 and 30 millions of packets. Table I reports the distribution of packet sizes for each trace. While small packets account for almost half of the total packets in the Campus trace, they represent a third in the CAIDA trace wherein large packets are the majority.

By considering these two scenarios combined with two different real traces, we seek to evaluate the impact of the number of secondary flows, the path size, and the low/high transmission rates on the accuracy of our method, thereby providing a fair outlook on the accuracy of our proposed solution.

TABLE I
DISTRIBUTION OF THE PACKET SIZES.

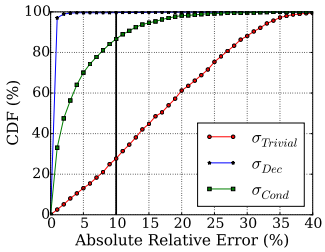
Packet sizes (bytes)	< 200	200 - 1400	> 1400
Campus trace	45.96 %	23.74 %	30.30 %
CAIDA trace	32.86 %	16.10 %	51.04 %

B. Distribution of the accuracy

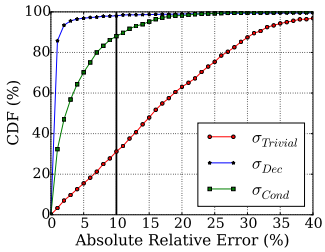
We begin our analysis by calculating the standard deviation of the E2E delay found by our proposed solution, σ_{Dec}, on each scenario wherein the traffic may either be extracted from the Campus trace or from the CAIDA trace. This leads to a total of four combinations.

As the value found for σ_{Dec} depends on the particular considered excerpt of trace (besides the whole trace itself), we repeat its calculation over hundreds of independent excerpts. More precisely, we conduct 600 independent experiments using the CAIDA trace, and 1440 using the Campus trace, resulting in 600 and 1440 samples for each method. Each individual sample of standard deviation was found by running our proposed solution over 100 consecutive packets of the primary flow. We then use these samples to derive the median error, and the cumulative distribution function of the relative error committed by the different solutions. Note that the relative error is simply calculated as follows: $\frac{|\sigma - \sigma_{Exact}|}{\sigma_{Exact}}$ where σ can be σ_{Dec}, σ_{Cond}, or $\sigma_{Trivial}$.

1) Scenario 1 with Campus trace: To begin with, we consider Scenario 1 wherein the traffic is generated from the Campus trace. We set links to the low-capacity mode. We consider two different levels of loads for the primary flow, f_1: moderate and high, corresponding respectively to an average rate of 1 and 1.5 Mbps. Fig. 3 depicts the cumulative distribution for the relative error committed by our solution (σ_{Dec}), as well as by the trivial solution ($\sigma_{Trivial}$) and by the Conditional method (σ_{Cond}). Fig. 3(a) reports the case when the rate of f_1 is moderate, while Fig. 3(b) exhibits the case of a high load. Either way, the trivial solution leads to very poor predictions, thereby invalidating this approach. Indeed, only around 30% of samples differed by less than 10% from the exact value. As for the Conditional solution, it tends to lead to satisfactory results with almost 90% of samples being less than 10% away from the exact value. Finally, our proposed solution delivers a very high level of accuracy. Virtually all its samples are within less than 5% of the exact value.



(a) Rate of f_1: 1.0 Mbps.

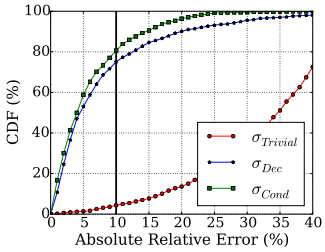


(b) Rate of f_1: 1.5 Mbps.

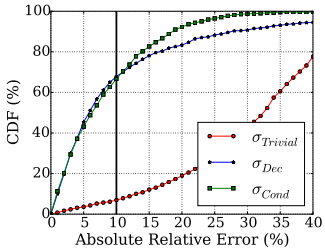
Fig. 3. Scenario 1 with Campus trace: Cumulative distribution function of the relative errors for the standard deviation of the E2E delay.

Due to space limitation, we do not present the results obtained with Scenario 1 with CAIDA trace in the high-capacity mode. Note that, our proposed solution provides very accurate predictions for the standard deviation of the E2E delay, that are even better than with the Campus trace.

2) Scenario 2 with Campus trace: We now consider the more complex Scenario 2. Fig. 4 represents the corresponding results when the Campus trace is used to generate the traffic and the links are set to high-capacity mode. First, and regardless of the rate of the primary flow f_1, the trivial solution fails to provide accurate results. Second, with this scenario, the difference between the Decomposition method (σ_{Dec}) and the Conditional method (σ_{Cond}) is much less marked. Both tend to deliver around 80% of samples within less than 10% when the rate f_1 is equal to 1 Mbps, while this number drops to less than 70% when the rate of f_1 grows to 1.5 Mbps.



(a) Rate of f_1: 1.0 Mbps.



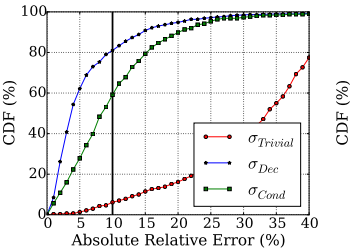
(b) Rate of f_1: 1.5 Mbps.

Fig. 4. Scenario 2 with Campus trace: Cumulative distribution function of the relative errors for the standard deviation of the E2E delay.

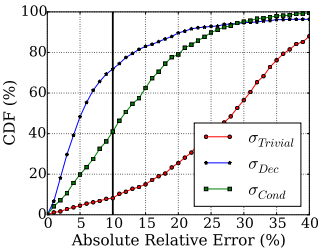
3) Scenario 2 with CAIDA trace: In our last case, we use the CAIDA trace to generate the traffic while maintaining the Scenario 2, in high-capacity mode though. Here, we observe, on Fig. 5, that our proposed solution (σ_{Dec}) steadily outperforms the Conditional method (σ_{Cond}). Indeed, with f_1 = 1.0 Gbps, more than 80% of the found samples σ_{Dec} differ by less than 10% to the exact value, against 60% of samples σ_{Cond}. When the rate of f_1 is larger, both methods tend to be less accurate but the drop is less pronounced for our proposed solution.

C. Workload influence on the accuracy

In this section we study more carefully the influence of the rate of the primary flow f_1 on the accuracy of our proposed solution. To do that, we consider several rates for f_1, ranging from a low level to a high level and we study the median of the empirical standard deviation of the E2E delay. Fig 6 clearly shows that the evolution of σ_{Dec} as well as σ_{Cond} are very close to σ_{Exact}, which is not the case of $\sigma_{Trivial}$. In Fig. 6(b), we see that the accuracy of σ_{Dec} decreases. This can be explained by packet losses that increase with the workload and that skew our estimator.

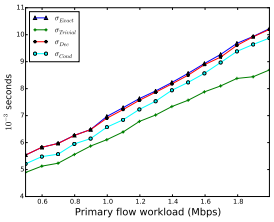


(a) Rate of f_1: 1.0 Gbps.

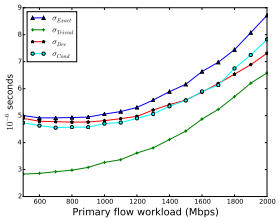


(b) Rate of f_1: 1.5 Gbps.

Fig. 5. Scenario 2 with CAIDA trace: Cumulative distribution function of the relative errors for the standard deviation of the E2E delay.



(a) Scenario 1 with campus trace, low capacity (in Mbps).



(b) Scenario 2 with Caida trace, high capacity (in Gbps).

Fig. 6. Median of the empirical standard deviation of the E2E delay.

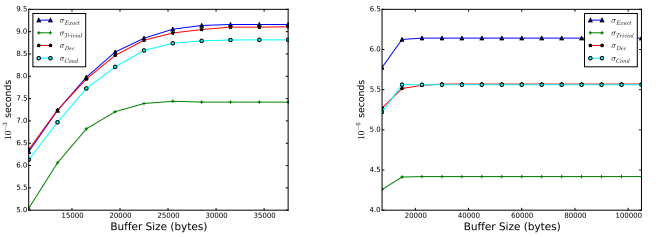
D. Buffer size influence on the accuracy

Finally, we study the influence of the buffer sizes at each node on the accuracy of our proposed solution. This is of important matter because the most difficult component of the end-to-end delay is the waiting time that highly varies as it depends on the buffer state found upon arrival (e.g., its value is null if the buffer is found empty upon arrival). The workload of f_1 is set to 1.5 Mbps for the low-capacity mode (1.5 Gbps for the high-capacity mode). Fig. 7 shows the evolution of the median of σ_{Exact}, σ_{Dec}, σ_{Cond} and $\sigma_{Trivial}$ for different sizes of buffer, ranging from 7,000 to 40,000 bytes for the low-capacity mode (7,500 to 105,000 bytes for the high-capacity mode). As the size of buffers increases, so does the actual values of standard deviation of the E2E delay, as shown by σ_{Exact}. We observe that, in this case, σ_{Dec} tends also to increase with growing values of buffers, and is kept very close to the exact values in Scenario 1. With Scenario 2, the standard deviation median converges immediately to reach a plateau, meaning that the buffer size has a weak influence on the accuracy of our method in this case. The two standard deviations, σ_{Cond} and σ_{Dec}, present a similar behavior with

TABLE II
NUMBER OF OPERATIONS TO APPLY THE DIFFERENT METHODS ASSUMING A NETWORK OF N NODES AND M PACKETS. IN THE CONDITIONAL METHOD, $L_k \in [1..M]$ IS THE NUMBER OF BINS CONSIDERED IN THE CONDITIONAL EXPECTATION AT k-TH NODE.

	Decomposition method	Conditional method	Trivial method
Number of operations on the k-th node ($1 \leq k < N$)	$N(17M+5)-(k+1)(14M+3)$	$N(3M+2)+2(11M+3)$ if $1<k<N-1$ $N(3M+2)+11M+3$ if $k=1$ or $k=N-1$	$5M+2$
Computations at the N-th node only	$2N+3M+1$	$2N+3M+1$	0
Number of operations on the SDN controller	$N-1$	$4(N-3)L_k+N-1$	$N-2$

an error of approximately 10%. The difference with σ_{Exact} is greater than the one found in Scenario 1. It is due to the path size and the number of secondary flows that are greater for this scenario.



(a) Scenario 1 with Campus trace. (b) Scenario 2 with CAIDA trace.

Fig. 7. Evolution of the standard deviation against buffer sizes.

E. Computational and networking complexity

In Table II, we show the number of operations required on the nodes (SDN switches) and controller to implement the three methods. An operation can be an addition, subtraction, multiplication, division, or the max operator (the one of the recurrence function). The computations that concern only node N correspond to the computations of Eq. 2. Otherwise, all the other local computations (i.e. $\mathbb{V}[W_k]$, $Cov(W_k, W_l)$ and $Cov(W_k, S_l)$) concern all the other nodes. We can notice that the complexity differs from one node to another. The trivial method is the less complex as it neglects some covariance terms. The difference between the Decomposition method and the Conditional method is the computation of $cov(W_k, W_l)$.

Concerning the networking complexity, the message sent to the controller contains only one value for the trivial and Decomposition methods, since a node sums all its estimates required to compute the variance. Consequently, the operations on the controller consists only in summing the values from the N nodes. For the Conditional method, a node has to send the sum of the different estimates and one or two histograms that describe the conditional expectation (one for nodes 1 and N and two for the others). In terms of packet monitoring, as already mentioned, each node has to measure three quantities for each packet: length, waiting time and inter-arrival time.

IV. CONCLUSION

Providing SDN controllers with estimates of the end-to-end (E2E) delay for flows can enable forthcoming SDN networks to have a better management of their resources, and to better deal with flows with stringent requirements of Quality of Services. In addition to the expectation (mean value), an estimate of its variance (or, similarly, its standard deviation) would allow a better characterization of the E2E delay.

In this paper, we extend a previous approach to estimate the E2E delay of a given path across the network by making use only of measurements collected locally on each node of the network. The initial approach was restricted to the case of a single flow, whereas, in this paper, we handle the case of multiple competing flows. We compare its accuracy with other solutions using several scenarios with different numbers of nodes, workload levels, buffer sizes and traffic traces. We also include a complexity study to assess the computational and networking cost of our proposed solution. Numerical results show the general good accuracy of our proposed solution under various conditions.

V. ACKNOWLEDGMENT

This work is partly funded by the French ANR INFRA DISCO under the "ANR-13-INFR-013" project.

REFERENCES

[1] N. McKeown *et al.*, "Openflow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, 2008.
[2] N. Feamster, J. Rexford, and E. Zegura, "The road to SDN: an intellectual history of programmable networks," *ACM SIGCOMM Computer Communication Review*, 2014.
[3] G. Almes *et al.*, "A One-way Delay Metric for IP Performance Metrics," RFC 7679, Tech. Rep., 2016.
[4] S. B. Moon, "Measurement and Analysis of End-to-end Delay and Loss in the Internet," Ph.D. dissertation, University of Massachusetts at Amherst, 2000.
[5] B.-Y. Choi *et al.*, "Analysis of Point-To-Point Packet Delay In an Operational Network," in *IEEE INFOCOM*, 2004.
[6] ——, "Practical Delay Monitoring for ISPs," in *ACM CoNEXT*, 2005.
[7] J. Wang, M. Zhou, and Y. Li, "Survey on the End-to-End internet Delay Measurements," in *HSNMC*, 2004.
[8] N. Nguyen, T. Begin, A. Busson, and I. Guérin Lassous, "Towards a passive measurement-based estimator for the standard deviation of the end-to-end delay," in *NOMS 2016*, 2016.
[9] ——, "Approximating the end-to-end delay using local measurements: a preliminary study based on conditional expectation," in *IEEE ISNCC*, 2016.
[10] G. Pongrácz, L. Molnár, and Z. L. Kis, "Removing roadblocks from SDN: openflow software switch performance on intel DPDK," in *EWSDN*, 2013, pp. 62–67.
[11] "ICFA SCIC Network Monitoring Report," 2016.
[12] D. Sass, "The dormitory network "Selfnet" of the University of Stuttgart," Oct. 2004. [Online]. Available: http://www.ikr.uni-stuttgart.de/~sass/traces/Welcome.html#selfnet
[13] "The CAIDA Anonymized Internet Traces 2015 Dataset," 2015. [Online]. Available: http://www.caida.org/data/passive/passive_2015_dataset.xml