Performance measurements of QUIC communications

Performance Measurements of QUIC Communications

Fabio Bulgarella[†], Mauro Cociglio[‡], Giuseppe Fioccola[*], Guido Marchetto[†], Riccardo Sisto[†]
Politecnico di Torino[†], Telecom Italia[‡], Huawei Technologies[*]

ABSTRACT

Performance measurement in terms of packet loss, delay, and jitter is key in modern packet switched networks. These values give a clear indication of the quality of service (QoS) perceived by users, thus being helpful to service providers to properly support, in particular, real-time communications such as voice and video conferences. This paper addresses these issues in QUIC-based communications, introducing a novel performance measurement methodology and comparing it with existing proposals in this field. The new solution for delay measurement uses only one more bit in addition to the Spin Bit, rather than the two additional bits required by the Valid Edge Counter (VEC) solution. Despite this, it is equally effective in overcoming the limitations of the Spin Bit solution and it performs similarly to the VEC solution. These results are shown by means of an experimental validation and evaluation on a proper testing environment.

1 INTRODUCTION

IP Networks are inherently not reliable and not real-time. Their operating principles, based on a best effort strategy, do not guarantee data delivery in a predefined time window or the delivery itself. This means that every time a packet is transmitted there is a certain probability that this will be delayed or lost during its journey through the network. When a communication service — in particular, a real-time voice or data service such as call, video conference, etc. — is provided by means of a packet-switched IP network, a performance measurement in terms of packet loss, delay and/or jitter on packet flows carrying the service provides an indication of the quality of service (QoS) perceived by the end users of the communication. Although network providers already rely on different tools to measure the quality of their links, the use of passive techniques is capturing their interest as it guarantees the measurement of traffic on a large scale without introducing any additional traffic into the network.

Almost all the Internet protocols have not been conceived from the beginning with the intent to provide explicit characteristics or functionality exploitable to obtain information about delay and loss rate of their connections. In fact, the few features available in this field have been introduced later on, as optional fields.

QUIC [2] is a recent experimental transport-layer UDP-based protocol proposed by Google and adopted by the IETF with the intent to standardize its features and use it as a faster substitute of TCP. As QUIC encrypts almost the whole transmission (including the control data and the packet number field), passive measurements are possible only by exploiting the few unencrypted bits exposed on the wire. In order to achieve this result, just three bits — placed in the QUIC short header — can be used. The first one is already assigned to the latency spin bit; the remaining two are available for future purposes. As QUIC is already extensively used by Google services and it is going to be standardized by the IETF with likely widespread adoption in the near future, it would be a shame not to take the advantage to provide QUIC with explicit support for measurability. A first step in this direction has already been made by the introduction of the previously mentioned spin bit[5], a latency signal which enables passive measurability of network delay. However, the algorithm behind its functioning does not guarantee proper measurements in all network conditions. To face this issue, an additional two-bit validation signal (i.e., the Valid Edge Counter (VEC) [4]) has been proposed with the purpose to give an instrument to correctly detect those spin-bit measurements that, due to network impairments, must be discarded because incorrect.

In this paper we are proposing an alternative way of improving the spin bit performance in delay measurements, which uses only one additional bit instead of the two required by the VEC. In this way, not all the reserved bits are used, but one remains free for other purposes, such as for example, measurement of loss rate.

The remaining part of the paper is organized as follows. Section 2 recalls the related work, most notably the passive measurement techniques previously introduced for QUIC (spin bit and VEC). Then, section 3 describes the new proposed technique, which we call Delay Bit, and section 4 presents an experimental validation with comparison between this solution and the previous ones. Finally, section 5 draws some conclusions.

2 RELATED WORK

The *latency spin bit* [5] is a single bit signal that toggles once per RTT. Basically, client and server maintain an internal per-connection spin value (i.e., 0 or 1) used to set the spin bit on outgoing packets for that connection. Then, when a packet is received, the client sets the connection spin value to the opposite value contained in the received packet; on the contrary, the server sets the connection spin value to the same value contained in the received packet. So, in a nutshell, the client inverts whereas the server reflects. This simple mechanism allows the endpoints to generate a square wave such that, by measuring the distance in time between pairs of consecutive transitions (which are called edges) observed in the same direction, a passive on-path observer can compute the round-trip delay of the connection. Moreover, if the observer is symmetrically placed on the channel — so it has visibility on both the upstream and downstream channels — it can determine the components of the RTT[1] by measuring the delay between the edge observed in the upstream direction and the one previously detected in the downstream direction, and vice versa. The latency spin bit performs well in the absence of network impairments and when senders are neither application nor flow control limited. On the contrary, packet loss may tend to cause wrong estimates of RTT due to periods width changes. Whereas reordering can lead an observer to incorrectly report two very short RTT samples when observing a reordered packet that has generated spurious edges[1]. Finally, application-limited senders cause the spin bit to measure no longer the delay of the network but the period of the application (if the latter is larger than the former).

The *Valid Edge Counter (VEC)* [4] was designed precisely to address these problems. The VEC is a two-bit signal added to each packet whose purpose is to explicitly report whether an edge was valid when transmitted by the endpoint. The VEC is set by the endpoints using the same logic. First of all, a value greater than zero is assigned exclusively to valid edges[2]; therefore, every transmitted packet between two consecutive spin edges carries a VEC value of 0. Then, when an endpoint detects an incoming packet carrying a spin transition, the VEC value of the next generated edge is set to the value contained in the received packet incremented by 1 (holding at 3). Basically, the value of the VEC is increased every time a valid edge is reflected by one of the two endpoints, actually counting the number of semi-paths (i.e., the path between client and server or between server and client) correctly crossed by the edge without incurring network impairments. Instead, when the endpoint detects an impairment such as a reordered or lost edge, the VEC is set back to 1 so that the observer avoids completing incorrect measurements. Moreover, to address the spin bit limitation where the duration of the period is overestimated in case of application-limited senders, the VEC algorithm introduces a delay threshold in edge reflection exceeded which the outgoing edge is transmitted, even in this case, with a VEC of 1. A passive observer, looking at the VEC value of each spin edge packet, can decide if that edge can be used to start or to complete an RTT measurement. An edge carrying a VEC of 1 or 2 can be used to start a new RTT measurement, whereas an edge carrying a VEC of 3 to complete a previously started one (and to start a new one). An edge carrying a VEC of 2 can also be used to complete an RTT component measurement.

Through our delay measurement mechanism, we provide a valuable instrument to solve the limitations shown by the latency spin bit while proposing an alternative solution to the Valid Edge Counter, which, using both the two remaining bits available in the header, prevents the addition of further features within the QUIC protocol (e.g., a loss signal).

[1] They represent, indeed, the components of the end-to-end RTT concerning the paths between the client and the observer, and between the observer and the server.

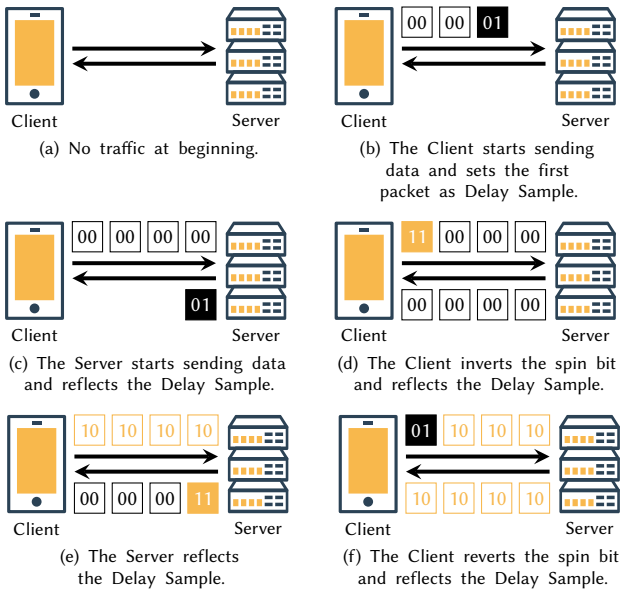
[2] An edge is considered valid when generated by one of the two endpoints. All the additional edges present in the spin signal are the result of reordering.

3 DELAY MEASUREMENT

Our delay measurement solution is based on an additional single-bit signal that we name *delay bit*. This signal can be used by passive observers to measure the RTT of a network flow, avoiding the spin bit ambiguities that arise as soon as network conditions deteriorate. Unlike the spin bit, which is set in every packet transmitted on the network, the delay bit is set only once per round-trip period (and so per spin bit period). Therefore, the main idea is to have a single packet, with a second marked bit (the delay bit indeed), that bounces between client and server during the entire connection life. This single packet takes the name of *delay sample*. Figure 1 shows the entire mechanism. A simple observer placed in an intermediate point, tracking the delay sample and the time in which it is encountered in every spin bit period, can measure the end-to-end round trip delay of the connection as the difference in time between two consecutive delay samples. In order to describe the delay sample working mechanism in detail, we have to distinguish two different phases which take part in the delay bit lifetime: generation and reflection. The former leads to the generation of the delay sample, while the latter is in charge of realizing the bounce behavior of this single packet between the two endpoints.

3.1 The generation phase

Since a single delay sample should bounce on a round-trip period, the delay sample generation is performed only by the client. This one, when the connection starts and the spin



(a) No traffic at beginning.
(b) The Client starts sending data and sets the first packet as Delay Sample.
(c) The Server starts sending data and reflects the Delay Sample.
(d) The Client inverts the spin bit and reflects the Delay Sample.
(e) The Server reflects the Delay Sample.
(f) The Client reverts the spin bit and reflects the Delay Sample.

Figure 1: The delay bit mechanism: spin bit and delay bit values represented on each packet, filled packets indicate a delay sample.

bit is set to 0, initializes the delay bit of the first outgoing packet to 1, making it the delay sample for that spin period.

Theoretically, in absence of network impairments, the delay sample should bounce between client and server continuously, for the entire duration of the connection. Actually, this is highly unlikely mainly for two different reasons. First, the packet carrying the delay bit might be lost during its journey on the network which is unreliable by definition. Second, one of the two endpoints could stop (or delay) sending data because the application is limiting the amount of traffic transmitted (in this case delay sample reflection is aborted as explained in next section). To deal with these problems, the algorithm provides a procedure to regenerate the delay sample and to inform a possible observer that a problem has occurred so that it can restart the measurement process. Being a delay sample strictly related to its spin bit period, the client verifies that every spin bit period ends with its delay sample. If that does not happen and a spin period terminates without a delay sample, the client waits an *empty period* in which no delay sample is generated. Then, in the following period, it restarts the generation phase setting the delay bit of the first outgoing packet to 1. The empty period is needed to inform intermediate observers that, due to an issue, a new measurement session is starting.

3.2 The reflection phase

Reflection is the process by which a delay sample is bounced between client and server. It takes place just for packets carrying a delay bit set to 1. At this stage, the behavior of the two endpoints is slightly different. Whenever a packet with the delay bit set to 1 arrives: **the server** marks the first packet in the opposite direction as the delay sample *if it has the same spin bit value*. While if it has the opposite spin bit value this sample is considered lost. Whereas **the client** marks the first packet in the opposite direction as the delay sample, *if it has the opposite spin bit value*. While if it has the same spin bit value this sample is considered lost. These conditions of validation — emphasized in the description — are necessary to identify and discard those samples that, due to reordering, might move to a contiguous period. In both cases, if the outgoing marked packet is transmitted with a delay greater than a predetermined threshold since the reception of the incoming delay sample (1ms by default), reflection is aborted and this sample is considered lost. Thanks to this mechanism, which is similar to the one seen in the VEC algorithm, it is possible to reject all those measurements that, due to application-limited senders, would be overestimated and not true.

3.3 Observer logic

Unlike what happens with the spin bit for which it is necessary to validate or at least heuristically evaluate the goodness of an edge, the delay sample can be used by intermediate observers as a simple marker between a period and the following one.

We already mentioned that the end-to-end RTT measurement of a QUIC flow can be determined by merely computing the difference in time between two delay samples observed in a single direction. However, it must be specified that a measurement, to be valid, must take into account the timestamps of *two consecutive delay samples belonging to adjacent spin-bit periods*. For this reason, an observer, in addition to intercepting and analyzing the packets containing the delay bit set to 1, must maintain awareness of each spin period in such a way as to be able to assign each delay sample to its period and, at the same time, identifying those periods that do not contain it (see section 3.3.1).

An on-path passive observer that is sniffing traffic in both directions — from client to server and from server to client — can also use the delay sample to measure "upstream" and "downstream" RTT components. It does this by measuring the delay between a delay sample observed in the downstream direction and the one observed in the upstream direction, and vice versa. Also in this case, it should verify that the two delay samples belong to two adjacent periods, for the upstream component, or to the same period for the downstream component.

3.3.1 The waiting interval. Like stated previously, every time an empty period is detected, the observer must restart

the measurement process and consider the next delay sample that will come as the beginning of a new measure. As a result, being able to assign the delay sample to the corresponding spin period becomes a crucial factor for the proper functioning of the entire algorithm. Considering that the division into periods is realized by exploiting the spin bit square wave, it is easy to understand that the presence of spurious spin edges — caused by packet reordering — would inevitably lead the observer to overestimate the amount of periods actually present in the transmission. This results in a greater number of empty periods detected by the observer and the consequent decrease of the actual RTT samples achievable. Therefore, in order to maximize the performance of the whole algorithm, the observer must implement a mechanism to filter out spurious spin edges.

To face this problem the waiting interval has to be introduced. Basically, every time a spin bit edge is detected, the observer sets a time interval during which it rejects every potential spurious edge observed on the wire. While at the end of the interval, it starts again to accept changes in the spin bit value. This guarantees proper protection against spurious edges in relation to the size of the interval itself. For instance, an interval of 5ms is able to filter out edges that have been reordered by a maximum of 5ms. Clearly, the mechanism does its job for intervals smaller than the RTT of the observed connection. Using an interval bigger than the RTT of the connection reduces considerably the RTT sample rate and could lead an observer to produce wrong estimations under certain conditions. This value can be manually set or configured automatically based on the average RTT experienced by the connection (e.g., a passive observer could simply set the waiting interval to half the average delay).

4 EVALUATION

In order to evaluate our solution we implemented the delay bit algorithm in QuicGo[3], an open source QUIC implementation that roughly implements the IETF QUIC draft. The resulting source code is available at https://github.com/fabiobulgarella/quic-go. The whole evaluation has been performed by means of Mininet [3], a network emulation tool that, using specific Linux Kernel features, can emulate — on a single host — a complete network topology composed of virtual hosts, switches and links interconnecting them. Mininet internally uses NetEm to introduce network impairments such as delay, jitter, reordering and loss. The emulated network is configured to introduce a base fixed delay of 40ms. Depending on which test is performed, network impairments such as loss and reordering are introduced into the network. All links are bandwidth limited to 1 Gbps. Note that in order to obtain realistic results, all tests are performed using the default settings of QuicGo. This means that no changes have been done on congestion control algorithm of QUIC protocol, then the transmission window size is free to change without any limitation according to experienced network conditions.

To test the operation of the delay bit algorithm, we compare its measurement results with those obtained using the VEC algorithm, which produces consistent samples in every network conditions[1]. In this case, QuicGo has been modified to mark at the same time packets with both mechanisms. Being the number of bits available in the header less than the four required to simultaneously implement spin bit, delay bit and VEC, the second most significant bit of the first byte — always set to one and therefore useless in an experimental context — was sacrificed for the purpose. By doing this, it was possible to compare the two algorithms and their behavior using exactly the same QUIC flows.

All tests proposed involve the download of 200 MB of data from the server to a client. Without reordering and loss applied, all proposed methods produce almost the same measurements. By default, in fact, the delay bit is set in the first packet of the spin-period. This means that, in absence of impairments, spin-edge, delay bit and VEC are always carried by the first packet of each period. As a consequence, RTT samples are computed taking into account the same edges for each measurement.

4.1 The effect of random losses

Being the RTT value determined by measuring the difference in time between two consecutive delay samples, a packet loss affects the number of achievable measurements only when a delay sample is lost. To evaluate the effects of random losses on delay bit functionality, links are configured to drop a specified percentage of packets. This percentage is varied to obtain an overall network loss rate ranging from 0% to 20%. As stated previously, congestion control is enabled and not limited.

Figure 2 shows how the computed average RTT value varies under different loss conditions. As expected, the solely spin bit is not enough to correctly calculate the RTT value of the network. The determined value increases with increasing loss rate due to the greater influence exerted by the QUIC recovery process which causes the entire application to slow down. The same problem is not experienced by the delay bit and the VEC algorithms, as both use a time threshold in reflection. Their produced measurements are consistent with the configured network delay and perfectly comparable. This proves the reliability of the delay bit in the presence of even important losses.

Figure 3 instead offers an overview of the number of RTT samples obtainable using the two algorithms. Here two remarks can be made. Firstly, the VEC observer produces more

[3]https://github.com/lucas-clemente/quic-go/

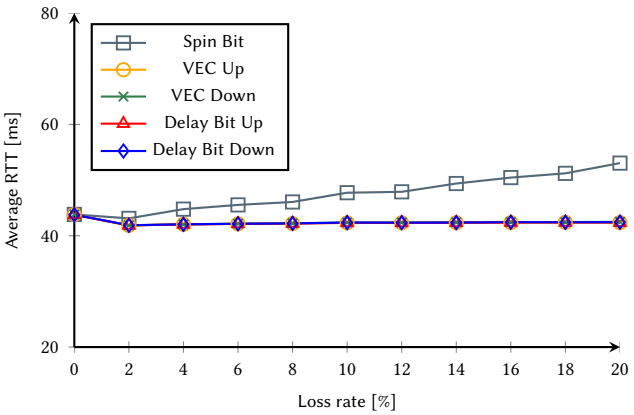


Figure 2: Average RTT value pace determined by each observer considering different random loss rates.

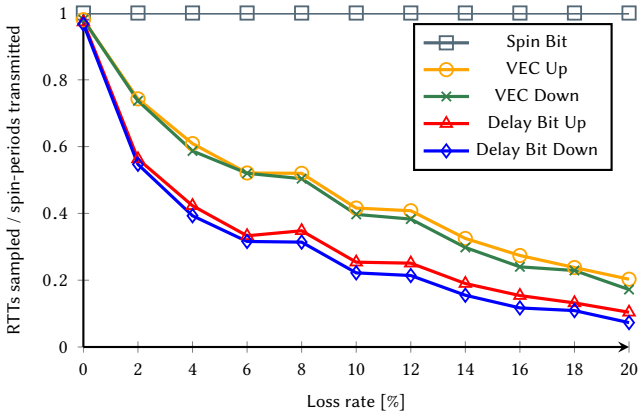


Figure 3: Number of RTT samples taken by each observer (normalized to the amount of spin-periods produced by the endpoints) considering different random loss rates.

valid RTT samples compared to the delay bit observer. This difference is to be attributed to the recovery process of the delay sample. When a delay sample loss is detected by the client (the only one in charge of performing this task), this one introduces an empty period in which no delay sample is transmitted before regenerating it. On the contrary, the VEC solution takes half round-trip to detect the loss of an edge (i.e., both endpoints can detect losses) and instantly restarts the algorithm, generating a new valid edge with VEC equal to 1. Secondly, the upstream observer produces slightly more samples than the downstream counterpart. Since the client is downloading data from the server, the amount of packets transmitted in the upstream direction is significantly less than those sent in the opposite direction. For this reason, the client is more often led to generate traffic holes — which cause lacks of delay-sample/valid-edge reflection — than the server, especially in the case when the application is limited by the QUIC recovery mechanism. As a consequence, the observer placed in the upstream direction will complete a greater number of measurements as the measurement process is restarted mostly by the client.

4.2 The effect of reordering

The spin bit alone in the presence of reordering can lead to the generation of very small fake periods and to the contraction of those adjacent. The VEC, as already demonstrated in [1], allows the observer to reject spurious edges an detect lost edges. However, in case of reordering, its behaviour is highly restrictive to the point of discarding any edge that has been rearranged. In the case of the delay bit, being the measurement performed on a single packet (i.e., the delay sample) that is bounced between the two endpoints, no measurements are rejected in case of reordering.

To compare these two approaches, reordering is introduced into the emulated network by holding back a certain percentage of packets for an additional millisecond. Each packet traverses two impaired links, one before and one after the observer, and might therefore be reordered up to two times per direction.

Figure 4 shows the effects of different reordering rates on the computed average RTT value. First of all, the spin bit curves show how reordering seriously affects its reliability. Although on the upstream direction the average measurements are not too far from the real RTT value of the network, in the downstream direction the results worsen dramatically. This is due to the fact that fewer packets are forwarded in the upstream direction with respect to the downstream direction. For this reason, the distance between packets is higher whereas the probability of being reordered is lower.

Looking at the results produced by the other two algorithms, the VEC returns values close to the base delay experienced by the network. The delay bit instead produces measurements whose RTT value tends to slightly increase as reordering rate increases. This is due to the fact that no measurement is discarded in case of packet reordering. As a consequence, with high reordering rates, the computed average RTT value is aligned to the base network delay plus the reorder delay (1 ms) introduced into the network. This approach guarantees results closer to the actual delay experienced by network users. On the contrary, discarding the measurements in case of reordering — as VEC does — undoubtedly produces measurements more faithful to the base network delay, but at the same time more distant from the experienced one. This shows the effectiveness of our approach in case of packet reordering, which is expected to become a more significant event in a near future with the wide spread of software-defined networks.

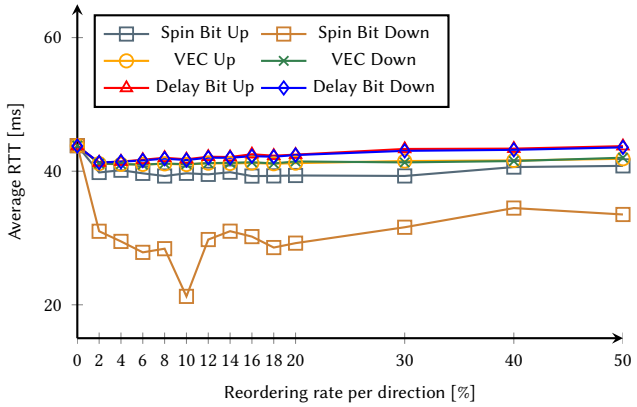


Figure 4: Average RTT value pace determined by each observer considering different reordering rates.

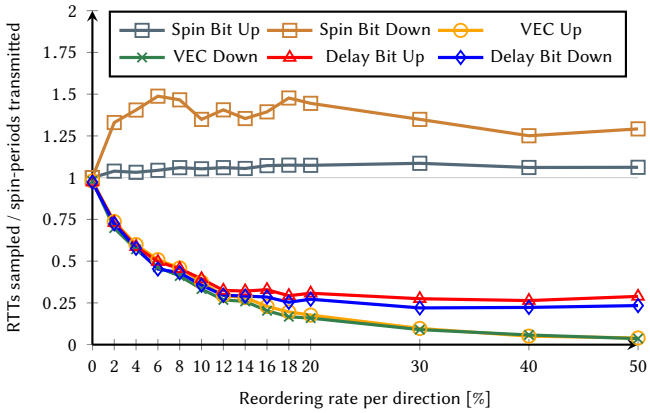


Figure 5: Number of RTT samples taken by each observer (normalized to the amount of spin-periods actually produced by the endpoints) considering different reordering rates.

Regarding the amount of samples achievable by the different observers, Figure 5 shows a significant overview. As expected, the spin bit observer produces more samples than those actually transmitted (also in this case the gap is greater for the downstream channel for the same reasons explained above). On the contrary, as expected, the number of samples decreases significantly for both delay bit and VEC observers. The delay bit performs better than the VEC as it guarantees a good number of samples even in conditions of heavy reordering (more than 25%). However, for small reordering values the difference between the two systems is minimal. This behavior is to be attributed to the continuous transmission delays introduced by the congestion control which, experiencing reordering, tends to vary the transmission window size with consequent slowing down of the whole application; this leads inevitably to the introduction of traffic holes. In the presence of these ones, it has already been shown that the delay bit is slower in restarting the measurement process.

5 CONCLUSION

In this paper we have proposed a new method for passive delay measurement of QUIC, which is based on the combined used of a delay bit and the spin bit. We have shown that this new method performs similarly to the VEC method, being able to overcome the limitations of the measurements based on the spin bit only. However, this new delay-bit method uses only one more bit in addition to the spin bit, in contrast with the VEC method, which uses two additional bits. As a consequence, the delay-bit method does not exhaust all the bits reserved in the QUIC short header, leaving one free bit that could be used for other measurements, such as loss rate. From the experimental evaluation results, it is possible to observe some small differences between the delay-bit method and the VEC method. In the presence of increasing packet loss, up to 20%, both methods give the same (correct) RTT measurements, but the VEC method loses less samples. Instead, in the presence of packet reordering caused by some packets being affected by extra delay, the delay-bit method performs better than the VEC method, because it gives measurements that also take the extra delay of reordered packets into account, while the VEC method totally disregards them. For the same reason, in these conditions, the delay-bit method loses less samples than the VEC method, especially when the reordering rate is high. This better performance of the delay-bit solution under reordering is particularly interesting, considering the higher impact that packet reordering may have in next generation SDN-based networks.

REFERENCES

[1] Piet De Vaere, Tobias Bühler, Mirja Kühlewind, and Brian Trammell. 2018. Three Bits Suffice: Explicit Support for Passive Measurement of Internet Latency in QUIC and TCP. In *Proceedings of the Internet Measurement Conference 2018 (IMC '18)*. ACM, New York, NY, USA, 22–28. https://doi.org/10.1145/3278532.3278535

[2] Jana Iyengar and Martin Thomson. 2019. *QUIC: A UDP-Based Multiplexed and Secure Transport.* Internet-Draft draft-ietf-quic-transport-19. Internet Engineering Task Force. https://datatracker.ietf.org/doc/html/draft-ietf-quic-transport-19

[3] Bob Lantz, Brandon Heller, and Nick McKeown. 2010. A Network in a Laptop: Rapid Prototyping for Software-defined Networks. In *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks (Hotnets-IX)*. ACM, New York, NY, USA, Article 19, 6 pages. https://doi.org/10.1145/1868447.1868466

[4] Brian Trammell. 2019. *An Explicit Transport-Layer Signal for Hybrid RTT Measurement.* Internet-Draft draft-trammell-ippm-spin-00. Internet Engineering Task Force. https://datatracker.ietf.org/doc/html/draft-trammell-ippm-spin-00

[5] Brian Trammell and Mirja KÃijhlewind. 2018. *The QUIC Latency Spin Bit.* Internet-Draft draft-ietf-quic-spin-exp-01. Internet Engineering Task Force. https://datatracker.ietf.org/doc/html/draft-ietf-quic-spin-exp-01