Three Bits Suffice: Explicit Support for Passive Measurement of Internet Latency in QUIC and TCP

Piet De Vaere, Tobias Bühler, Mirja Kühlewind, and Brian Trammell
ETH Zurich, Switzerland

ABSTRACT

Passive measurement is a commonly used approach for measuring round trip time (RTT), as it reduces bandwidth overhead compared to large-scale active measurements. However, passive RTT measurement is limited to transport-specific approaches, such as those that utilize Transmission Control Protocol (TCP) timestamps. Furthermore, the continuing deployment of encrypted transport protocols such as QUIC hides the information used for passive RTT measurement from the network.

In this work, we introduce the latency spin signal as a light-weight, transport-independent and explicit replacement for TCP timestamps for passive latency measurement. This signal supports per-flow, single-point and single direction passive measurement of end-to-end RTT using just three bits in the transport protocol header, leveraging the existing dynamics of the vast majority of Internet-deployed transports. We show how the signal applies to measurement of both TCP and to QUIC through implementation of the signal in endpoint transport stacks. We also provide a high-performance measurement implementation for the signal using the Vector Packet Processing (VPP) framework. Evaluation on emulated networks and in an Internet testbed demonstrate the viability of the signal, and show that it is resistant to even large amounts of loss or reordering on the measured path.

CCS CONCEPTS

• **Networks → Network measurement**; *Transport protocols*;

ACM Reference Format:
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 IMC '18.* ACM, New York, NY, USA, 7 pages. https://doi.org/TBA

1 INTRODUCTION

Round Trip Time (RTT) is a key metric in Internet measurement for network operations and research. This measurement is often performed actively, through the venerable ping utility. However, due to the overhead that large-scale active measurement introduces, measuring latency at scale remains an area of active research [7]. Passive RTT measurement can reduce these overheads, but is limited to transport-specific approaches, such as those that utilize Transmission Control Protocol (TCP) timestamps [15] or exploit the properties of commonly deployed congestion and flow control algorithms [3].

In this work, we introduce the *latency spin signal* as a transport-independent, more efficient, and simpler refinement of the timing information primitive provided by TCP timestamps. The latency spin signal requires three bits per packet and adds negligible complexity to endpoint code, but allows any on-path observer to extract the end-to-end RTT from a flow with minimal state requirements. Its design follows the "Principles for Measurability in Protocol Design" proposed by Allman et al. [1], aiming to provide an explicit, visible, in-band and cooperative signal for passive two-way latency measurement. The signal does not change transport protocol dynamics, nor does it require the transmission of packets that would otherwise not be sent by the protocol. It additionally allows endpoints to signal that a packet is "delayed" (e.g. because the application protocol had no data to send), a problem that leads to issues with existing passive RTT measurement techniques [5]. More specifically, assuming a transport protocol that generates feedback at least once per RTT – an assumption that holds for the vast majority of Internet traffic – an observer can extract one RTT sample per RTT, whether observing one or both sides of the flow.

Explicit support for passive measurability is especially important in the case of QUIC – a new, encrypted transport protocol originally designed by Google that is currently under standardization in the IETF. Google's deployment of a previous version of QUIC reached 35% of its external traffic by the end of 2016 [9]. Though encapsulated in a UDP header for deployability reasons, QUIC provides reliability and congestion control comparable to TCP, and adds features such as stream multiplexing for better support of HTTP traffic. It also integrates TLS deep into its machinery, running transport and security handshakes simultaneously, and encrypting not only its payload but also as much as possible of the transport control information. This makes it fundamentally different from a passive measurement standpoint: most of the information used by passive measurement approaches for TCP is simply not visible to on-path observers.

For example, while QUIC still uses acknowledgment (ACK)-based feedback for reliability and congestion control, packets carrying ACK information are indistinguishable from other packets, as the ACK information is carried in the encrypted part of the packet. During an active QUIC connection, packets flow in both directions as in TCP, however, there is no way to correlate a packet in one direction with the packet that triggered it in the opposite direction, as it can be done with sequence and acknowledgement numbers in TCP. The latency spin signal adds this correlation ability back to the network-visible portion of the QUIC header [16, 17]. Though the ACK remain invisible, the fact that QUIC produces a minimum

IMC '18, October 31–November 2, 2018, Boston, MA, USA

ACM ISBN TBA…$TBA
https://doi.org/TBA

amount of feedback during each RTT for congestion control purposes ensures that that the spin signal is trigged once per RTT and thereby exposed the flow's current RTT to passive observers.

The latency spin signal is also useful in TCP, as the information it exposes has comparable utility on the wire to TCP timestamps, with much less overhead and less inadvertent information exposure. As the spin signal is independent of the internal transport machinery, it is less likely to lead to ossification and, in contrast to timestamps, it does not expose enough information about each endpoint's clock to lend itself to fingerprinting [19]. For passive RTT measurements, the latency spin signal therefore represents a more privacy-friendly method than the current state of the art.

We evaluate the latency spin signal with implementations of the signal both in QUIC and TCP. Experimentation with the QUIC implementation focuses on an emulated environment, as QUIC's design and the maturity of implementation is in a state of rapid flux due to the standardization process. Experimentation with TCP focuses on results on the open Internet from a variety of access networks.

With this paper, we make the following contributions:

- The definition of a novel and efficient method to signal per flow RTT together with approaches to passively measure this signal.
- The implementation of the latency spin signal in an open-source QUIC library[1] (implementing the draft-05 version of QUIC) as well as in the Linux kernel's TCP stack[2].
- The implementation of a device based on the Vector Packet Processing (VPP)[3] library that uses the latency spin signal to measure application RTT.
- The evaluation of a QUIC latency spin signal against related approaches in a wide variety of emulated network conditions.
- The evaluation of a TCP latency spin signal against TCP timestamps in an Internet testbed.

2 MECHANISM

The latency spin signal is composed of two parts: a spin *bit* that changes onces every RTT, and a 2-bit *VEC* that indicates the validity of the latency information between two spin bit toggle events which are called edges. These three bits appear on every packet sent by each side of a transport connection. The generation of the signal does not require the generation of additional packets not otherwise sent by the transport, and does not interfere with the transport's own transmission scheduling algortihms. The mechanism is lightweight, and can be added to a transport protocol with minimal effort[4].

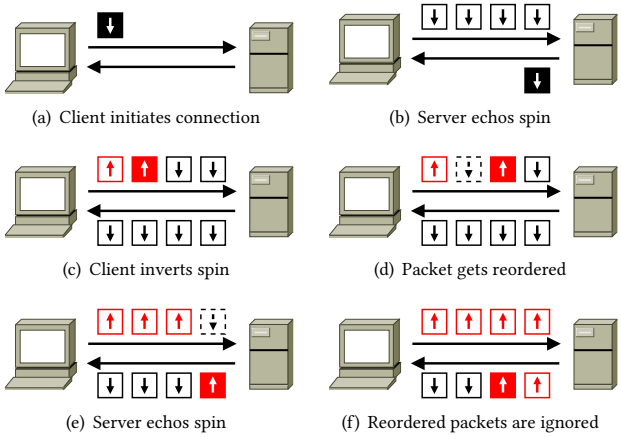
2.1 The spin *bit*

The spin bit itself is the part of the spin signal that is used to actually monitor the RTT of a flow, as it is toggling once per RTT. A passive on-path observer can log the period between two transitions and thereby extract a flow's RTT.

[1]https://github.com/pietdevaere/minq
[2]https://github.com/mami-project/three-bits-suffice
[3]https://fd.io/technology/
[4]Our addition of the signal to minq touched approximately 80 lines of code.

```
1  spin_next, vec_next, t_last, PN_max ← 0, 1, 0, 0
2  Function OnPacketReceive():
3      if PN > PN_max then
4          if spin_next ≠ spin_rcv then
5              vec_next ← min(vec_rcv + 1, 3)
6              t_last ← t_sys
7          if is_client then spin_next ← ¬spin_rcv
8          else spin_next ← spin_rcv
9          PN_max ← PN
10 Function OnPacketSend():
11     spin_snd ← spin_next
12     if t_sys − t_last > delay_max then
13         vec_snd ← min(vec_next, 1)
14     else vec_snd ← vec_next
15     vec_next ← 0
```

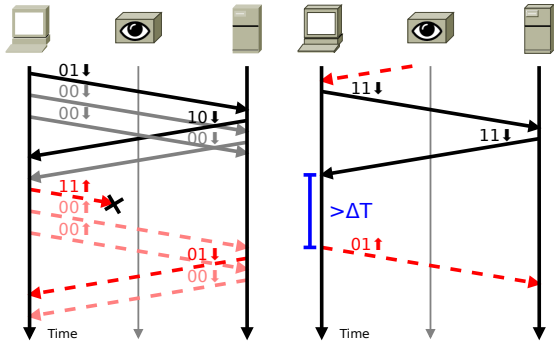
Algorithm 1: Logic of the spin signal. t_{sys} is the current system time, PN_{max} the currently highest packet number.



(a) Client initiates connection (b) Server echos spin (c) Client inverts spin (d) Packet gets reordered (e) Server echos spin (f) Reordered packets are ignored

Figure 1: The spin bit mechanism: arrows indicate spin values, filled packets have a non-zero VEC value, and dashed lines indicate reordered packets.

This toggling behavior is illustrated in Figures 1(a) to 1(c). When the client initiates a connection, it will start sending packets with spin 0 (Figure 1(a)). Once the server starts sending packets, it will echo back the spin value it last received from the server (Figure 1(b)). Conversely, once the client receives a packet from the server, it will set the spin of its outgoing packets to the opposite of the spin value it last received from the server (Figure 1(c)). This asymmetric behavior will cause the transition point of the spin values to 'spin' through the network, resulting in exactly one spin edge per RTT at any point in the network and thereby exposing the flow's RTT between two edges. Furthermore, when logging the duration between two edges on different flow directions, the delays from the observation point up- and downstream to each of the endpoints can be measured separately — we refer to these as *component RTT*. This part of the algorithm is described in Lines 1, 7, 8 and 11 (yellow) of Algorithm 1.



(a) Increment VEC (with edge loss) (b) Reset VEC on invalid edge

Figure 2: The VEC mechanism: VEC values are represented in binary format; spin values as arrows.

Table 1: Overview of VEC values and usage.

01	10	11	Spin bit transition can be used as
✓	✓	✓	Start of an RTT measurement
	✓	✓	End of an RTT component measurement
		✓	End of an RTT measurement

In order to provide resilience to packet reordering (Figure 1(d)), the algorithm ignores all packets that were received out of order. Figures 1(e) and 1(f) illustrate how this mechanism removes distortions in the spin bit sequence. In Lines 3 and 9 (green) of Algorithm 1 we assume that every packet has a unique, in-order sequence number (PN) which is at least visible to the endpoints, such as the sequence number in TCP or packet number in QUIC.[5]

2.2 The Valid Edge Counter (VEC)

While the endpoints can trivially detect reordering by observing the packet sequence number, this does not always hold for on-path observers, as sequence information may be encrypted (as it is the case with the QUIC packet number). An observer considering only the spin bit would then incorrectly report two very short RTT samples when observing a reordered packet, as illustrated in Figure 1(d). To address this problem, we introduce the VEC, a two bit signal that explicitly marks packets that carry a valid spin bit edge set by the endpoint, and increases its value with the number of non-distorted transitions of the spin signal. Figure 2(a) illustrates this mechanism which is described in Lines 4, 5, 14 and 15 (blue) of Algorithm 1. Non-edge packet carry a VEC of zero. All spin edge carry a nonzero VEC value, which is set either to one plus the VEC value of the packet that triggered the edge transision or at maximum three. Thereby, the VEC indicates the number of network transitions during which the spin signal has not been distorted. Observers can use this information to decide if a spin edge can be used to begin or end an RTT measurement as shown in Table 1. As an example, a VEC of two ('10') can either be used to start a new RTT measurement or end a RTT component measurement as it indicates two network transitions during which the spin signal was not distorted. With this mechanism, even an observers that can only monitor one direction of a flow can also detect distortions that occurred in the other flow direction. In Section 3.1 we compare component and full RTT measurements in more detail.

As observers estimate the validity of their measurements based on the VEC, the end hosts can also use the VEC to signal when they *know* that the spin bit does *not* carry valid RTT information. This situation occurs when an endpoint introduces excessive delay between receiving and transmitting a packet carrying a spin edge (e.g. because of application-limited traffic). This is done by the remaining (uncolored) lines in Algorithm 1, and is illustrated in Figure 2(b). The endpoint will reset the VEC to its initial value of '01'.

Endpoints can also explicitly indicate that they have opted out of RTT measurement by setting the VEC to '00' on all packets, or control the sample rate by probabilistically marking an otherwise non-'00' edge as '00'.

2.3 Efficient Passive Measurement of the Latency Spin Signal

Our passive observer implementation is based on Vector Packet Processing (VPP)[6], a library for high-speed packet processing in userspace. Our plugin adds a node to VPP's packet processing graph that processes all traffic in four steps: *(i)* **detecting** spin-enabled transports, using User Datagram Protocol (UDP) port pairs to detect QUIC, and the spin bit in the SYN to detect TCP (as in Section 4) with the spin signal[7]; *(ii)* **retrieving or creating state** for the observed spin-enabled flows using the 5-tuple as hash key; *(iii)* **extracting** the measurement bits from the header; and *(iv)* **estimating** RTT using the spin bit, VEC and the previous values from the flow state. Our implementation either writes per-packet RTT estimations to a file or alternatively live stats of all active flows can be retrieved using CLI commands (`sudo vppctl spinbit stat`).

Our observer implementation tracks flows based on the 5-tuple (source and destination Ip address and port as well as IP protocol number) which results in 13 bytes of state for IPv4-based flows (37 bytes for IPv6). In addition, the observer saves the initial source port (2 bytes) to identify the direction of the observed VEC, the current spin bit value (1 bit) as well as the start time of the current RTT measurement and the previous RTT value. In our implementation, we use two 64 bit variables to save these time values. Rounding up, we need 32 bytes to measure the RTT of one IPv4 flow, allowing an on-path device to measure the RTT of roughly 32k concurrent flows per megabyte of memory.

The observer is also very lightweight in terms of computational complexity. Each observed packet results in retrieval of associated flow state (e.g. via hash table lookup), followed by at most three boolean comparisons to determine the direction of the packet, to find a new spin bit value and to validate the spin transition based on the observed VEC value. In case of a valid transition, we either save the starting time or perform a single subtraction of two 64 bit values to compute the observed RTT.

[5] The correctness of the spin bit is shown in the following thesis report [4].

[6] https://fd.io/technology/

[7] Note that our observer monitors all TCP traffic, for comparison with timestamp-based RTT measurements.

2.4 Sample Rate Analysis

The VEC increases the accuracy of the signal by allowing observers to reject spin bit transitions that may result in bad samples. In other word, it trades off sample rate for accuracy. More precisely, given independent random packet loss and reordering probabilities p_L and p_R and average flow RTT $\overline{\text{RTT}}$, Equation (1) gives a first order approximation of the sample rate per flow direction.

$$\text{sample rate} = \frac{(1-p_L)^3(1-p_R)^3}{\overline{\text{RTT}}} \quad (1)$$

The numerator of Equation (1) represents the probability that an endpoint generated spin transition caries a VEC value of 3. That is, the probability that the spin signal has not been disturbed by loss or reordering for at least three network transitions. When measuring component RTTs, the numerator is be squared rather then cubed.

Equation (1) shows once more that the maximal number of RTT samples per RTT is limited to one (zero loss and reordering probability) which enables an observer to track changes in RTT due to e.g. congestion control adaptations but at the same time filters out high frequency oscillations.

Of course the sample rate has an additional upper bound of the sending rate of the underlying transport, as the latency spin signal does not change the traffic pattern. This is a property of all passive latency measurement: one cannot measure what's not there.

3 THE LATENCY SPIN SIGNAL IN QUIC

We added the latency spin signal to an open source QUIC implementation. Because the QUIC header format was in constant flux when we did these experiments, and for flexibility of experimentation, we added an additional *measurement byte* to the header. This byte carried — along with other experimental signals — the spin bit and VEC. Because a flow's initial RTT can be measured based on handshake semantics, the spin signal is only carried in the QUIC short header (i.e. any packet after the initial QUIC version and key negotiation which uses long headers).

3.1 Evaluation: Accuracy and Sample Rate in Emulated Networks

We evaluated the QUIC implementation, enhanced with the latency spin signal, on an emulated network using Mininet [10]. The emulated network has a base RTT (propagation delay without queuing) of 40 ms (which we take as a typical regional-continental internet RTT given the growing importance of content-delivery networks). In each of our tests, a client continuously uploads data to a server with a constant rate below the maximum link capacity, while various impairments are introduced to the network. We only present the results for non-adaptive traffic in this paper, as high loss rates or high degrees of reordering reduces the sending rate of adaptive traffic to a minimum, making the impact of these impairments less visible.

We implemented four mechanisms to passively observe the RTT, for comparison purposes:

Spin bit The observer monitors only the spin *bit* to get a base measurement to rate the impact of reordering and loss.

Packet number The observer uses packet sequence information to rejects reordered packets, similar to Algorithm 1[8].

Heuristic The observer monitors only the spin *bit*, but rejects RTT samples below one tenth of the current estimate.

VEC The observer observes the full spin signal, i.e. spin bit and VEC, and rejects invalid edges based on the VEC value.

The quality of the spin signal is evaluated using two metrics: error relative to client estimated RTT as per RFC 6298 [13], and the number of samples obtained per RTT. Furthermore, we also consider how many samples can be taken by the VEC observer when down- and upstream delay or *component RTTs* are measured separately. Because our observers can see both flow directions, they should ideally measure two samples (one for each direction) per RTT. However, spin bit transitions with VEC values below two can lower the sample rate, while superfluous transitions can increase it incorrectly.

3.1.1 Packet reordering. We first evaluate the tolerance of the spin signal to packet reordering. We use NetEm [8] to randomly delay a configured fraction of packets by 1 ms. The results are shown in Figure 3.

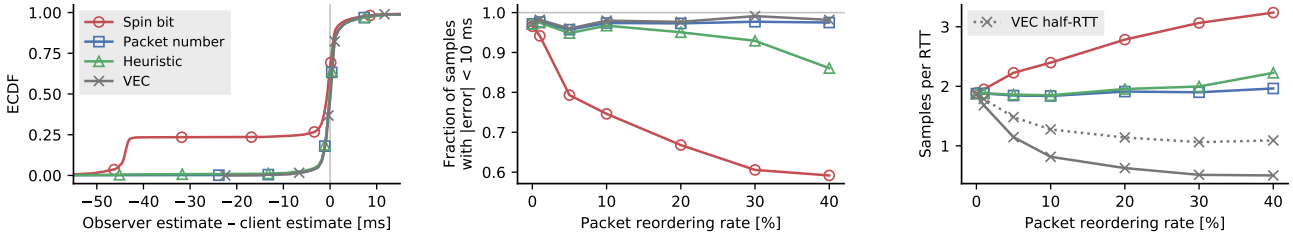
Figure 3(a) shows the distribution of the RTT estimation error at a 10% reordering rate. It can be seen that the spin-bit-only observer often produces RTT estimates with an error around -40 ms. As this corresponds to the network's RTT, this means that many near zero RTT samples are taken. As can be seen in Figure 1(d), this is exactly what is to be expected, as reordered packets cause rapid transitions in the spin bit signal. Although this is a problem for the basic spin bit observer, at this reordering rate, all other observers are able to filter out this effect. However, as can be seen in Figure 3(b), this does not hold anymore for higher reordering rates. That is, for reordering rates above 10 % the accuracy of the heuristic observer starts to deteriorate.

Although their error performance is similar, the VEC and packet number observers are fundamentally different: when the spin signal is disturbed, the VEC observer will drop the sample, avoiding all samples that could lead to incorrect measurements. On the other hand, the packet number observer continues to take RTT samples, leading to additional error in the sample. This effect is very pronounced in Figure 3(c). This figure also shows that the VEC observer rejects less samples when the up- and downstream RTTs are measured separately as we can additionally use spin bit transitions with VEC values of two (compare Table 1).

3.1.2 Packet loss. To evaluate loss tolerance, we configured NetEm to emulate burst loss using the simple Gilbert model [6]. The good reception periods have an average length of 100 packets. The average length of the loss bursts is varied. The results are shown in Figure 4.

Figure 4(a) shows that loss leads to overestimated RTTs. This is because when a packet carrying a spin edge is lost, the RTT measurement is not stopped until the next packet with the new spin bit value is observed. The long tail in the ECDF is caused by

[8] At the time of our experiments, QUIC's packet numbers were exposed in the unencrypted QUIC header. However, at the time of this writing, the QUIC Internet-Drafts specify encrypted packet numbers.

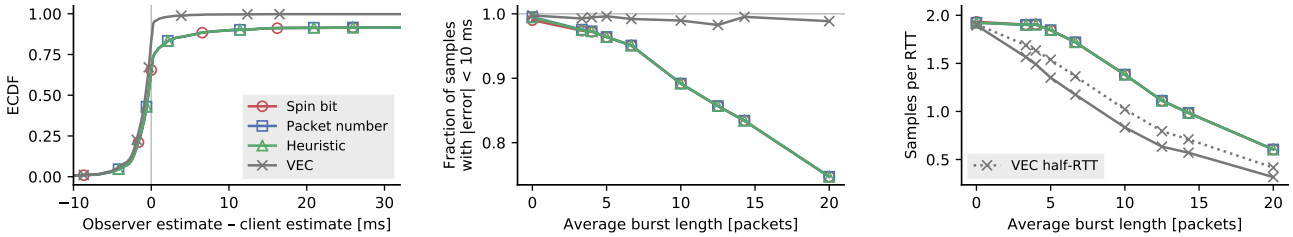


(a) Empirical Cumulative Distribution Function (ECDF) for a 10% packet reordering rate.

(b) The error in the observed RTTs for increasing reordering rates.

(c) The normalized number of RTT samples for increasing reordering rates.

Figure 3: The effects of reordering on spin bit based RTT measurements of a flow with a 40 ms RTT.



(a) ECDF for an average burst length of 10 packets.

(b) The error in the observed RTTs for increasing loss burst lengths.

(c) The normalized number of RTT samples for increasing loss burst lengths.

Figure 4: The effects of burst loss on spin bit based RTT measurements of a flow with a 40 ms RTT.

retransmission timeouts. Because these timeouts are significantly larger than the network's RTT, they also reduce the number of RTT samples that can be taken, as can be seen in Figure 4(c). Looking at Figure 4(b), we see that only the VEC observer remains accurate under burst loss, as the VEC indicates that the spin signal has been disturbed, allowing the observer to reject the incorrect samples.

4 THE LATENCY SPIN SIGNAL IN TCP

The latency spin signal was originally designed to ensure that passible latency measurements remain possible when using encrypted transport headers. However, its simplicity and low overhead make it a suitable transport-independent and explicit signal, regardless of whether or not encryption is used. To demonstrate this, we added the signal to TCP. Though TCP already supports passive RTT measurement through the timestamp option [15], this approach is less efficient (requiring eight bytes per packet, ten including options overhead; as opposed to three bits for the latency spin signal, or three bytes if implemented as a separate TCP option).

In addition to these efficiency gains, the VEC's ability to signal known bad samples addresses accuracy issues related to acknowledgment optimizations in modern TCP implementations [5]. Furthermore, the timestamp option exposes much more information about the operation of the endpoint than it appears at first glance. TCP implementations simply drive the timestamp signal from the kernel's interrupt counter, and different endpoints generally have a characteristic clock drift which can be used to determine which addresses belong to the same endpoint [14] and to fingerprint endpoints [19]. In addition, discontinuities in the timestamp sequence over long periods of time can be used to detect reboots or equivalent node replacements [2]. Currently, endpoints wishing to avoid these inferences have no choice but to disable timestamps, losing passive RTT measurability at the same time.

To explore the applicability of the latency spin signal to TCP, we implemented the signal in a patch for Linux 4.9 and 4.15, using the the three reserved flag bits in the TCP header for the spin bit and the VEC.

Compared to QUIC the TCP spin signal also appears on handshake packets. The client initializes the spin value to one on initiation, in order to quickly detect issues arising from the use of the reserved bits during experimentation. In addition, only packets with a sequence number greater than or equal to the last seen (as opposed to the maximum packet number) are considered. Our passive measurement device (Section 2.3) required only minimal changes to support TCP as well, accounting only for the fact that the bits are found at a different offset in a different header.

4.1 Evaluation: Comparing Spin and TCP Timestamp RTT Measurement

We deployed virtual machines running our patched Linux kernel on cloud nodes in seven networks with a global distribution[9] on 25 May 2018, and set up a simple test web server[10] on each of them. All traffic between these nodes was routed through an observation node running our VPP code in our local, on-campus infrastructure[11]. This approach allows us to verify mid-path passive measurability while still maintaining diverse link characteristics on legs from and

[9] DigitalOcean VMs in regions NYC1, SFO2, AMS3, SGP1, BLR1, FRA1, and LON1
[10] see https://github.com/mami-project/three-bits-suffice
[11] This routing was achieved by two-way network address translation on the machine running the VPP measurement observer, and has the effect of composing $n \times m$ paths from each access network via ETH to each cloud network.

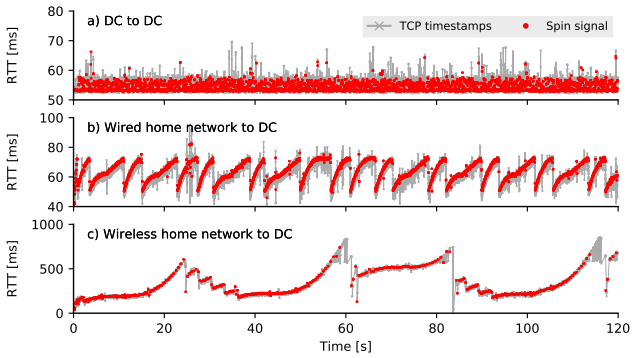


Figure 5: Spin- vs. TS-based RTT estimation over time.

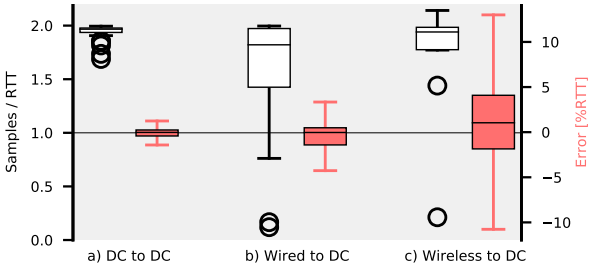


Figure 6: Sample rate per RTT (left plot/axis) and relative error between VEC- and TS-based estimation (right plot/axis).

to our midpoint. From these measurements we have 49 traces of 120 seconds each.

In addition, we accessed each server from five access networks in Europe[12], also steering traffic past our measurement node. On those networks where both wired and wireless endpoint connections were possible, we measured both. This yielded 53 additional traces.

In Figure 5, we illustrate the operation of the latency spin signal using traces of three examples of RTT measurements taken using the latency spin signal, compared to per-packet RTT measurements taken using TCP timestamps (TS). Subfigure (a) shows a typical inter-datacenter trace: the spin signal measurement stays fairly close to the minimum of the noisier TS measurement. Subfigure (b), taken from a node connected via Ethernet to a residential access router, shows a more typical situation with larger buffers. Subfigure (c), from a wireless network with a bad case of bufferbloat, shows an extreme situation with high delay and moderate loss. Here, the latency spin signal's sample rate reduces as the RTT increases; yet, it still provides accurate enough information for rough intra-flow measurement.

Overall the relative errors between VEC- and TS-based RTT measurements are small. As we do not have ground truth for the end-to-end latency, we compare each TS-based RTT sample to the latest value derived from the spin signal. As shown in the right-side (red) boxplots in Figure 6, the median error for the inter-datacenter measurements is -0.03% and -0.04% for the wired access case. For wireless access, we observed a slightly higher median error of 1.04%, because the spin signal overestimates the RTT measured by TS. However, in wireless networks RTT is highly variable and in some measurement runs we also observed fewer valid samples due to reordering or loss. As shown by the left-side (white) boxplots, the median number of samples per estimated average RTT for wireless runs is still high with 1.94, compared to 1.97 for the data center case. In the wired case, one of our access nodes experienced a high amount of packet reordering, probably due to traffic shaping, which led to a relatively high number of invalid VEC edges and a median sample rate of 1.82 with high variance. This shows the expected behavior: invalid samples are filtered out.

[12]We note that at least one of the involved access routers performed TCP header manipulation that resulted in a mangled latency spin signal. Handling edge cases in on-path TCP manipulation is a matter for future work.

5 CONCLUSION

In this paper, we presented a three-bit latency spin signal, explicitly enabling comparable passive RTT measurement for both TCP and QUIC. Though this signal is necessary in QUIC, to replace information lost through encryption of the transport headers with respect to TCP, we show that it is also a useful enhancement to TCP, providing equivalent passive RTT accuracy at a lower sample rate than timestamp-based passive measurements, with lower overhead and less potential for endpoint fingerprinting. As this is initial work on an overarching approach to protocol measurability in the spirit of Allman et al [1], we now look to future developments. The signal has a few limitations to address, which ongoing work is focused on.

First, the simple RTT sample generation method suggested by Table 1 will reject any RTT samples that experienced a lost or reordered edge within 1.5 RTT, which may result in an unacceptably low sample rate for precisely those flows with interesting problems to debug. Simple heuristics at the observer may allow edges with VEC 1 and 2 to be used to generate full RTT samples, decreasing the impact of loss and reordering on the sample rate.

Our implementation of the signal on TCP uses three reserved bits in the TCP flags word; we chose this approach as opposed to TCP options for reasons of efficiency, ease of implementation, and comparable deployability of formerly reserved flags [18] and new options [11, 12]. While our initial experimentation noted no stripping of these bits or dropping of packets based on them, middleboxes that handle these bits in ad-hoc ways could lead to duplication of signals or other oddities that a production-ready TCP implementation would need to detect and correct.

As the signal is intentionally separate from the rest of the transport machinery, any endpoint can simply refuse to participate without negative consequences for end-to-end connectivity. Any plan for Internet-scale deployment of the signal on endpoints must therefore consider the incentives for endpoints to participate by generating the signal.

Looking toward deployment, the Internet Engineering Task Force (IETF) QUIC working group has approved experimentation with the spin bit. Two independent implementations, including one from a major operating system vendor, have support for the spin bit, and one telecommunications operator and one network equipment vendor are actively working on experimental measurements using the signal.

Acknowledgments: The one-bit portion of the latency spin signal was initially proposed for QUIC by Christian Huitema, following discussions at the June 2017 interim meeting of the IETF QUIC Working Group in Paris. Thanks to Christian and the QUIC WG for input on the design leading to the mechanism described in this paper. Thanks to Manya Ghobadi, our shepherd, and the anonymous reviewers for their feedback, which significantly improved the paper. This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 688421, and was supported by the Swiss State Secretariat for Education, Research and Innovation (SERI) under contract number 15.0268. The opinions expressed and arguments employed reflect only the authors' views. The European Commission is not responsible for any use that may be made of that information. Further, the opinions expressed and arguments employed herein do not necessarily reflect the official views of the Swiss Government.

REFERENCES

[1] Allman, M., Beverly, R., and Trammell, B. Principles for Measurability in Protocol Design. *SIGCOMM Comput. Commun. Rev. 47*, 2 (May 2017), 2–12.

[2] Beverly, R., Luckie, M., Mosley, L., and Claffy, K. Measuring and Characterizing IPv6 Router Availability. In *Passive and Active Measurement* (Brooklyn, USA, 2015), J. Mirkovic and Y. Liu, Eds., Springer International Publishing, pp. 123–135.

[3] Carra, D., Avrachenkov, K., Alouf, S., Blanc, A., Nain, P., and Post, G. Passive Online RTT Estimation for Flow-Aware Routers Using One-Way Traffic. In *Proceedings of NETWORKING 2010* (Chennai, India, 2010), M. Crovella, L. M. Feeney, D. Rubenstein, and S. V. Raghavan, Eds., pp. 109–121.

[4] De Vaere, P. Adding Passive Measurability to QUIC. Master's thesis, ETH Zürich, 2018.

[5] Ding, H., and Rabinovich, M. TCP Stretch Acknowledgements and Timestamps: Findings and Implications for Passive RTT Measurement. *SIGCOMM Comput. Commun. Rev. 45*, 3 (July 2015), 20–27.

[6] Gilbert, E. N. Capacity of a burst-noise channel. *The Bell System Technical Journal 39*, 5 (Sept 1960), 1253–1265.

[7] Guo, C., Yuan, L., Xiang, D., Dang, Y., Huang, R., Maltz, D., Liu, Z., Wang, V., Pang, B., Chen, H., Lin, Z.-W., and Kurien, V. Pingmesh: A Large-Scale System for Data Center Network Latency Measurement and Analysis. In *Proceedings of the 2015 ACM SIGCOMM Conference* (New York, NY, USA, 2015), SIGCOMM '15, ACM, pp. 139–152.

[8] Hemminger, S., et al. Network emulation with NetEm. In *Linux conf au* (2005), pp. 18–23.

[9] Langley, A., Riddoch, A., Wilk, A., Vicente, A., Krasic, C., Zhang, D., Yang, F., Kouranov, F., Swett, I., Iyengar, J., Bailey, J., Dorfman, J., Roskind, J., Kulik, J., Westin, P., Tenneti, R., Shade, R., Hamilton, R., Vasiliev, V., Chang, W.-T., and Shi, Z. The QUIC Transport Protocol: Design and Internet-Scale Deployment. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication* (Los Angeles, CA, USA, 2017), SIGCOMM '17, ACM, pp. 183–196.

[10] Lantz, B., Heller, B., and McKeown, N. A Network in a Laptop: Rapid Prototyping for Software-defined Networks. In *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks* (New York, NY, USA, 2010), Hotnets-IX, ACM, pp. 19:1–19:6.

[11] Paasch, C. Network support for TCP Fast Open. Presentation at NANOG 67, January 2016.

[12] Raiciu, C., Paasch, C., Barre, S., Ford, A., Honda, M., Duchene, F., Bonaventure, O., and Handley, M. How Hard Can It Be? Designing and Implementing a Deployable Multipath TCP. In *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation* (San Jose, CA, 2012), NSDI'12, USENIX Association, pp. 29–29.

[13] Sargent, M., Chu, J., Paxson, D. V., and Allman, M. Computing TCP's Retransmission Timer. RFC 6298, June 2011.

[14] Scheitle, Q., Gasser, O., Rouhi, M., and Carle, G. Large-scale classification of IPv6-IPv4 siblings with variable clock skew. In *Network Traffic Measurement and Analysis Conference, TMA 2017, Dublin, Ireland, June 21-23, 2017* (2017), pp. 1–9.

[15] Strowes, S. D. Passively Measuring TCP Round-trip Times. *Queue 11*, 8 (Aug. 2013), 50:50–50:61.

[16] Trammell, B. A Transport-Independent Explicit Signal for Hybrid RTT Measurement. Internet-Draft draft-trammell-tsvwg-spin-00, IETF Secretariat, July 2018. http://www.ietf.org/internet-drafts/draft-trammell-tsvwg-spin-00.txt.

[17] Trammell, B., and Kuehlewind, M. The QUIC Latency Spin Bit. Internet-Draft draft-ietf-quic-spin-exp-00, IETF Secretariat, April 2018. http://www.ietf.org/internet-drafts/draft-ietf-quic-spin-exp-00.txt.

[18] Trammell, B., Kühlewind, M., Boppart, D., Learmonth, I., Fairhurst, G., and Scheffenegger, R. Enabling Internet-Wide Deployment of Explicit Congestion Notification. In *Passive and Active Measurement* (Brooklyn, USA, 2015), J. Mirkovic and Y. Liu, Eds., Springer International Publishing, pp. 193–205.

[19] Zander, S., and Murdoch, S. J. An Improved Clock-skew Measurement Technique for Revealing Hidden Services. In *Proceedings of the 17th Conference on Security Symposium* (San Jose, CA, 2008), SS'08, USENIX Association, pp. 211–225.