Switches Know the Exact Amount of Congestion

Serhat Arslan
Stanford University
sarslan@stanford.edu

Nick McKeown
Stanford University
nickm@stanford.edu

ABSTRACT

Deployed congestion control algorithms rely on end hosts to figure out how congested the network is. Initially, for TCP, the congestion signal was packet drops; a binary signal that kicks in only after the congestion is well underway. More recently, the trend has been towards using RTT as a congestion signal instead (e.g. Timely and BBR). But RTT is a noisy surrogate for congestion; it contains a valuable signal about congestion at the bottleneck but also includes noise from the queuing delay at the non-bottlenecked switches. Taking a step back, it is worth asking: *Why don't the switches and routers simply tell us the actual congestion they are experiencing?* After all, they must keep track of the precise occupancy of their own queues anyway; they can directly tell the end-hosts. Conventional wisdom said this is too expensive (in terms of additional bits in headers, or complexity and power consumption). We argue that even if this was once the case, it no longer is. Today, it is quite feasible, with negligible increase in power or lost capacity, to report the precise queuing delay at the switches, allowing the end hosts to make more accurate decisions when minimizing required buffering. We explore how this might work using modern programmable switches and NICs that stamp each packet with the queue occupancy (or the maximum queue occupancy along the path), which can be thought of as a multi-bit ECN signal. We provide evidence that the resulting signal is a more accurate indication of congestion at the flow's bottleneck and can lead to higher link utilization and shorter flow completion times than RTT-based algorithms. Consequently, it becomes easier to control required buffer sizes. Our goal here is not to argue for a particular multi-bit ECN algorithm, but to point out that in the future, there is no longer a need to rely on noisy RTT measurements at the edges.

1 INTRODUCTION

End-host congestion control algorithms rely on a congestion signal from the network, with three popular signals in use today. TCP Reno uses packet loss, while more modern schemes such as DCTCP [2] and Microsoft's DCQCN [28] use single-bit ECN marked on packets by the routers, and Google's TIMELY [21] uses changes in RTT as a congestion signal. Clearly, even in modern data centers, whose owners have complete control over the network topology and end-hosts, there is not a consensus on the best congestion signal. Our goal in this paper is to point out that (1) all three signals are noisy, imprecise *surrogates*, and (2) modern switches and routers now routinely provide a precise congestion signal: The current occupancy of the buffers where the congestion takes place. It's time to start using this more accurate, direct – and now readily available – congestion signal instead.

Packet loss and RTT are popular because they are measured at the end-host alone, fitting cleanly with the end-to-end argument. Packet loss is now known to be a crude signal, hence early interest in RTT for TCP Vegas [7]. More recently, Timely [21] and BBR [10] adopted RTT as their primary congestion signal. The idea is that variations in RTT measured at the edge indicate variations in queuing delay, hence measure the onset of congestion. But RTT includes both the congestion signal (the queuing delay at the bottleneck) plus the variations in queuing delay at all the other queues, regardless of whether they are congested. As we discuss in section 2, the non-bottleneck queues add noise to the congestion signal, reducing its efficacy, particularly as the number of hops increases. It is relatively easy to see that the noise leads to bigger queues and lost throughput.

It is natural to ask why the Internet does not explicitly feed back precise information about the congestion to the end-host. After all, in order to maintain a packet buffer, all switches and routers must know the precise instantaneous queue occupancy. Why don't they simply tell us? It is a bit hard to disentangle exactly why the early Internet protocols didn't provide queue occupancy as feedback. In part, few people were focusing on congestion until the infamous collapses reported in 1988 [18]. In the same year, Clarke's retrospective on the Internet design [12] doesn't mention congestion, but does point to the benefits of not assuming "internal knowledge of ... delays" which were "explicitly not assumed from the network". One explanation is that the desire to keep the network simple and streamlined, with little attention to congestion, created a blind spot in the early standards. It was not a deliberate decision to prevent congestion from being precisely communicated, or a belief that it was impractical, or that we were better off without it. Rather, it appears not to have been high enough on the priority list. It is interesting to imagine how different the field of congestion control would have evolved if precise, explicit feedback of queue occupancy had been part of the protocols from the outset.

In an attempt to gather more explicit information, Explicit Congestion Notification (ECN) was standardized in 1999 [24].

ECN was proposed so that Active Queue Management (AQM) mechanisms (originally, Random Early Detection (RED) [15]) could mark packets eligible for dropping, not for congestion control. However DCTCP [2] uses ECN to encode current congestion via multiple successive ECN bits. In hindsight, it is worth asking why ECN did not go further and place the bottleneck queue occupancy in the packet header. After all, a probabilistic decision to set an indicator bit in each packet demands more of the router implementation than simply just reporting the current queue occupancy. Then DCTCP could react faster and more accurately if every packet carried the precise bottleneck queue occupancy.

Today, with the advent of programmable switch ASICs (i.e. [3, 8, 11]), network owners can choose to carry new (standard or proprietary) information in packet headers. For example, INT [19] is a popular way to carry switch metadata to the end hosts. Alternatively, a network owner could reuse an existing (or add a new proprietary) header field to carry the queue occupancy. In the extreme, a packet traversing a path with k hops might carry the full queue occupancy vector, $q = (q_1, ..., q_{2k})$ including the reverse path. In section 3 we discuss that carrying vector q would require just a few lines of P4 code and would be the most accurate congestion signal. It might even be more than enough for many (it is up to a network owner to decide). A more bandwidth efficient congestion signal would be to carry a single value function of q along the path. In section 4 we consider three natural functions of q: (1) $f(q) = \sum_{i=1}^{2k} q_i$ (SUM) is the sum of the queue occupancies along the path, and is essentially the same signal as RTT (used by TCP Vegas, Timely and BBR) but without the constant propagation delay. (2) $f(q) = \mathbb{I}\{\exists_i q_i > thresh\}$ (IND), is a single bit indicator equivalent to ECN. (3) $f(q) = \max_i q_i$ (MAX) reports the queue occupancy at the bottleneck. MAX has the appealing property of capturing, in a single value, the degree of congestion at the bottleneck, without noise introduced by other non-bottlenecked links.

In section 4 we also compare the performance of Timely (RTT-based) and DCTCP (ECN-based) with algorithms that use our three occupancy-based congestion signals. Specifically, we simulate the Timely gradient-based approach, replacing RTT measurements with each of our congestion signals. Our main result is that MAX leads to a clean signal of congestion at the bottleneck, because there is no noise introduced by other routers along the path. By definition, the bottleneck is the link that serves the end-host at the smallest rate. Therefore, we can expect MAX (or similar) to be a good congestion signal to determine the end-host's sending rate. Indeed we found that in a network with 7 hops, MAX had 2× higher throughput than Timely. In addition, we simulated MAX with different numbers of bits to understand how many header bits are sufficient for signaling congestion precisely.

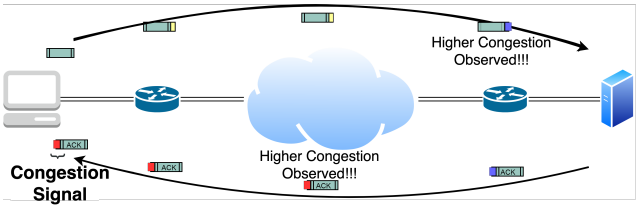

Figure 1: A typical procedure of updating the queue occupancy field on the packet header with the maximum queuing delay encountered along the path. Although the figure shows MAX formulation, any chosen function can be used to update the corresponding field, i.e. SUM.

Our simulations suggest that a single 4-bit value may suffice, with 8 bits plenty.

Our primary goal here is not to argue for a particular function of q. Rather, to point out that it is now possible to carry such a congestion signal, and that simple functions yield promising results compared to single bit ECN, packet loss or RTT. Motivated by the recent analysis in [29], section 5 discusses additional control algorithms that could exploit the information obtained via collected queue occupancy information, so that the required buffering can be minimized. We deliberately do not propose a specific congestion control algorithm, because we believe the correct algorithm depends on the context [26] and it is reasonable to expect that as networks become more programmable, different networks will use different algorithms. However, we do hope to motivate the use of non-surrogate, nanosecond-level accurate values as an improved signal. We predict that future switches and routers will routinely stamp queue occupancy as a fourth, new congestion signal, replacing or enhancing RTT.[1]

2 HANDICAP OF SURROGATE SIGNALS

Loss, ECN, and RTT are widely used signals, and this popularity mainly stems from the fact that they are easily obtained from the network. Given the number of well-defined standards (i.e. [7, 14, 16]), congestion control has evolved around these signals for the last four decades. But all three signals are by-products of the congestion and are not an exact measure of the congestion itself. Let's consider the quality of each signal in turn.

Loss, of course, is the most common congestion signal and indicates that at least one buffer is full along the forward or reverse path,[2] which means the news of congestion arrives

[1] Anecdotally we have already witnessed this happening in a small number of cloud data centers.

[2] We ignore here non-congestion related loss, which is typically rare in a modern wireline network

late, after the congestion is well underway and typically too late for latency sensitive high throughput applications.

RTT is the only continuous, high resolution signal that combines the constant propagation delay and the dynamic queuing delay of switches along the path. Timely [21] uses RTT gradient to infer congestion in the network. The idea is that a change in RTT indicates that the queue occupancy of at least one switch is changing. If the change is at the bottleneck link, we would like the end-host to change its sending rate. One could argue that the primary purpose of a congestion control algorithm is to adjust the sending rate of the end-host to match the fair share rate determined by the bottleneck link. But, of course, RTT does not necessarily tell us about changes at the bottleneck; an increase in RTT does not mean the bottleneck is more congested and therefore does not mean the end-host should reduce its sending rate. Additionally, RTT is the sum of all the delays encountered along the path, so some queue occupancy values may decrease while others increase making RTT a noisy estimate of congestion. If an end-host reacts to changes in the aggregate queue occupancy, it will react to irrelevant changes at non-bottleneck links, even when its fair share rate at the bottleneck has not changed. Section 4 shows some adverse consequences of this phenomenon.

ECN [16] is a single bit marking on packets by switches in the network. Unlike RTT, ECN value does not change depending on the distance to the destination. If the queue occupancy on a link is above a threshold, the packet's ECN field is set with a particular probability distribution. Consequently, congestion is completely ignored until the queue occupancy reaches the threshold value which requires relatively larger buffer sizes. If the ECN bit is set, the current bottleneck queue occupancy is guaranteed to be above the threshold. However, the binary ECN value does not indicate how bad the congestion is. Instead, modern schemes such as [2, 22, 28] use ECN marks from consecutive packets to infer the corresponding distribution and estimate the congestion themselves. Some studies suggest that multiple bits of ECN marking improve these algorithms [23, 25], motivating our study.

3 NON-SURROGATE SIGNAL

At the risk of stating the obvious, in a packet-switched wireline network there is a direct relationship between queue occupancy and congestion: They are *equivalent*; by observing one we are observing the other. By controlling one we are controlling the other. The exact measure of current congestion is determined *precisely* by the current queue occupancy (or queuing delay, which is queue occupancy divided by link rate) at links along the path. There is no other direct measure of congestion in a packet-switched wireline network. An end-host armed with an up-to-date measure of queue occupancy at every hop, has the best possible signal of congestion.

We measure the efficacy of a congestion control algorithm by the buffer occupancy it causes. If the bottleneck link goes empty, we are wasting capacity and hence needlessly prolonging long-lived flows. Then the end-host should increase its sending rate. If buffers are too full, we are unnecessarily delaying packets for short flows which requires end-hosts to back-off. Therefore, it is reasonable to provide queue occupancy as an explicit congestion signal from the network.

Another way to think about our approach, as opposed to RTT measurement, is to consider what happens when a cross flow causes a non-bottleneck queue to temporarily go non-empty, the sender has no need to adjust its window size in order to keep the bottleneck link busy, because the congestion conditions have not changed. As a consequence end-hosts need to be able to ignore irrelevant delay information to make the best decision on sending rate.

3.1 Stamping Queue Occupancy Information

Obtaining the queue occupancy information from a switch is now easier than ever before. Actually, it was never technically difficult or expensive; it just was not supported by fixed function switch ASICs until recently. Nowadays, any new switch ASIC is required to support INT [19], and hence is already capable of placing queue occupancy information into packet headers as they pass through.

We have heard erroneous claims that stamping packets with queue occupancy consumes significant additional power. A modern high-end switch chip consumes very little power reading a queue occupancy, increasing the packet size to hold it, then placing the value into the header. In one estimate, if every packet was stamped with its queue occupancy, it would add less than 0.05% to the overall chip power [6]. This is in part because the majority of the chip power is expended on fixed overhead (leakage current, serial I/O) and per-packet (not per-bit) processing in a pipeline.

In fact, queue occupancy is a simple and free piece of information because of the following reasons.

Switches always know the current queue occupancy value. To maintain one or more FIFO queues, the switch must do internal book-keeping to keep track of their occupancy.

Switches read the queue occupancy value anyway. As soon as a packet arrives, a switch needs to read the current occupancy value in order to decide whether to queue the packet or drop it. Similarly, the queue occupancy value is

updated at the time of departure. There is no need to read the value again.

ECN is already a form of queue occupancy stamping. ECN requires switches to read the current queue occupancy value, compare it to a threshold, and (in case of RED AQM) toss a coin to decide whether to mark the packet. Queue occupancy stamping would be a very similar procedure except using a larger field in the packet header. Later, we show in section 4.2 that half a byte of information is enough to represent the queue occupancy value. Even if we add a whole byte, this adds only 0.4% overhead to a 256B average size packet. Figure 1 illustrates the process of updating an occupancy-based congestion signal in the packet header.

With new programmable switches [3, 5, 8, 11] and programming languages [4], network owners can redefine how their switches process packets. A 700-line P4 program shows how metadata can be stamped to support INT, showing it is trivial to add queue occupancy stamping to deployed, programmable switches. Furthermore, we do not need to wait for new standards to be defined: the network operator need only pick a header location that works in their network and is agreed upon by the end hosts. In a data center, this can be proprietary and chosen to work with existing protocols and boxes. RFC 8592 [9] offers some guidelines to stamp packets with metadata.

Additionally, a network might choose to stamp packets with queuing delay rather than occupancy, allowing links with different line rates to be compared to each other (in nanoseconds instead of bytes).

In summary, as programmable switches become more pervasive, we anticipate that stamping packets will become universal, opening up many new tailored implementations and interesting research directions for congestion control.

4 EVALUATIONS

We use the NS-2 [17] simulator to compare four different congestion signals based on stamping packets with queue occupancy vector q:

- SUM: $f(q) = \sum_{i=1}^{2k} q_i$, the total queuing delay which is equivalent to RTT used in Timely and BBR.
- SUM_SQU: $f(q) = \sqrt{\sum_{i=1}^{2k} q_i^2}$, a function of the queuing delay in which longer queues have an outsized impact without ignoring smaller queues.
- MAX: $f(q) = \max_i q_i$, which only pays attention to the maximum queue size, i.e. the occupancy of the bottleneck queue.
- IND: $f(q) = \mathbb{I}\{\exists_i q_i > thresh\}$, the ECN signal formulation. We only consider this function in our simulations when we consider MAX with a single bit value.

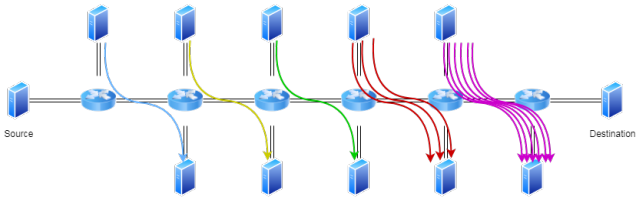

Figure 2: The primary flow (not shown) is from the source to the destination. The cross-cutting flows are shown in different colors matching the queue occupancy of the corresponding link in Figure 3.

Based on the chosen congestion signal, the end-host congestion control algorithm must choose the sending rate. To compare apples with apples, we use Timely's control algorithm for all our simulations. The Timely algorithm updates the sending rate based on the gradient of the congestion control signal (Algorithm 1 in [21]). The intuition is to slow down the sending rate by an amount proportional to increased congestion. In vanilla Timely, if RTT is less than a threshold T_{low}, the rate is increased additively by δ. If, on the other hand, $RTT > T_{\text{high}}$, the rate is decreased multiplicatively by β. If RTT is in between T_{low} and T_{high}, the rate is updated using the gradient, i.e. change in consecutive input signals. We evaluate SUM (an equivalent signal to RTT), SUM_SQU to emphasize congestion from bottleneck queues over non-bottleneck queues, and the MAX function, which eliminates the noisy signal from non-bottleneck queues.

We are aware that a gradient based algorithm may not be optimal for our congestion signals. Our goal here is to show the effect of noise in RTT over the exact congestion information. Section 5 provides more detailed discussion on alternative control algorithms.

4.1 Ablation Topology

We start with a scenario deliberately designed to highlight the problem caused by noise in RTT or the SUM signal. The topology is shown in Figure 2; all links are 10Gb/s. All flows in the topology have the same minimum RTT (their delay without queuing) to ensure that the flows change their rate equally in response to the same amount of change in delay. The primary multi-hop connection from the source to the destination shares queues along the way with several cross-cutting flows. The cross-cutting flows are shown by colored arrows in the figure. The flows are kept active throughout the simulation in order to generate queuing on the links. The last two links of the path are congested the most because more flows share these two queues.

The purpose of the cross-cutting flows is to create queuing delay, and hence fluctuating RTT, in the non-bottleneck queues. They are kept to a low enough rate so that only

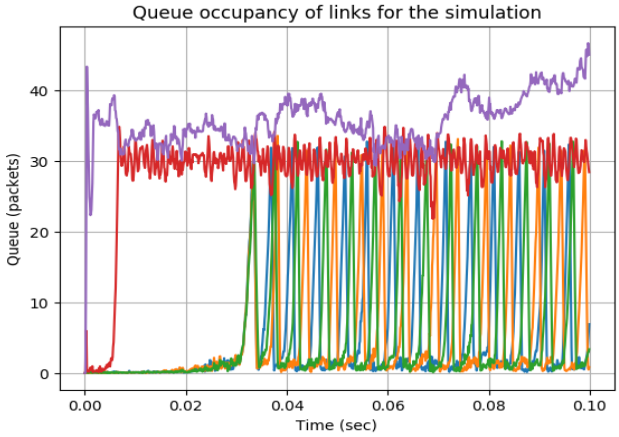

Figure 3: Buffer occupancy. The purple and red curves show the occupancy of the last two queues in the topology. The other curves, which oscillate wildly, are the non-bottleneck links. The oscillations are due to the bursty reactions of high rate cross-flows.

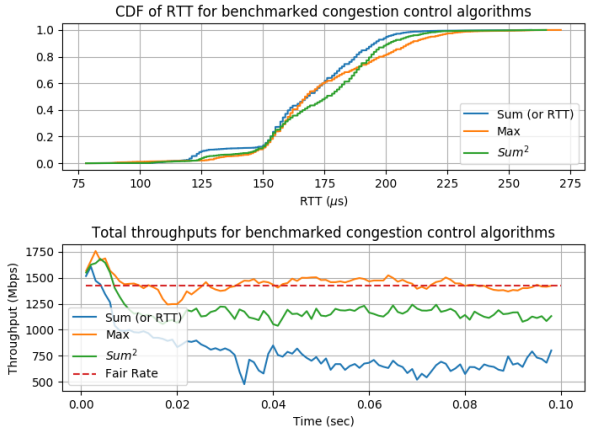

Figure 4: Primary flow RTT distribution and throughput trends for SUM, MAX, and SUM_SQU formulations with oscillating non-bottleneck queues along the path (Figure 2).

the final queue, with seven flows, is the bottleneck for the primary flow. Hence, with RTT or SUM as the congestion signal, the signal carries a noise component that grows with increasing hop count.

The time series of the queue occupancy of each buffer in the network is shown in Figure 3. During the simulation, all flows fill up the buffers until the congestion signal reaches T_{low}. At this point, the cross-cutting flows see a large positive RTT gradient and decrease their rates, causing the queue occupancy in the corresponding links to decrease. Hence these queue occupancies oscillate wildly. On the other hand, the last two links carry more cross-flows (three and six cross flows) and each flow does not affect the total queue occupancy as much as the single cross-flow case, hence the queue occupancy is more stable.

The distribution of round-trip time and throughput for our benchmark flows are given in Figure 4. Notice that in the presence of oscillating non-bottleneck queuing delay, if we use SUM or RTT as our congestion signal the flows receive allocations well below their fair-share rate (the fair-share rate of seven flows sharing a 10Gb/s link is 1.4Gb/s per flow). The fair-share rate of the main flow *should* be dictated by its bottleneck. However, because SUM flows cannot distinguish the oscillation in the non-bottleneck queues (single cross-flow links) from congestion, Figure 4 shows that they unnecessarily reduce their sending rate. SUM may have slightly lower tail round-trip time (not surprising, given this is its congestion signal), but it trades fairness and a big drop in throughput for the improved latency.

MAX and SUM_SQU lead to slightly larger RTT values than SUM, and a big increase in throughput. SUM_SQU pays less attention to non-bottleneck queues and hence is less affected by the noisy signal than SUM; still, the noise causes it to operate below maximum throughput. MAX can sustain much higher throughput because it completely ignores the noise caused by the cross-cutting flows. It converges to the fair-share rate even in the presence of significant noise.

4.2 Overhead in Header Bits

An important aspect of our congestion signal is its precision in terms of the number of bits used to represent the value. The switches must maintain a precise count of the number of packets and their sizes in the queue, likely using 16–32 bits. To minimize the communication overhead, in terms of link capacity needed to carry the extra field, we would like to determine how big the new field needs to be.

To compare the performance of different quantization levels for queue occupancy stamping, we ran simulations on a 3-level Clos topology with 192 hosts, 8 TOR switches, 4 leaf switches, and a single spine switch. Link capacities increases from 1Gb/s to 10Gb/s and 40 Gb/s towards the spine switch and every link has 5μs delay. All data packets are 1500 bytes long. Hosts randomly connect to each other and use the MAX congestion signal. We have tested quantizations with 1, 2, 4, 8, 16, and 32 (default) bits respectively. The quantized value represents which equal sized bin the queue occupancy lies in between zero and the full buffer size of 200 packets. We could alternatively use a non-linear quantization, to give preference to a longer bottleneck queue, but found it did not noticeably improve our results.

Figure 5 shows how the mean and 95$^{\text{th}}$ percentile round-trip time values improve as we increase the number of signal

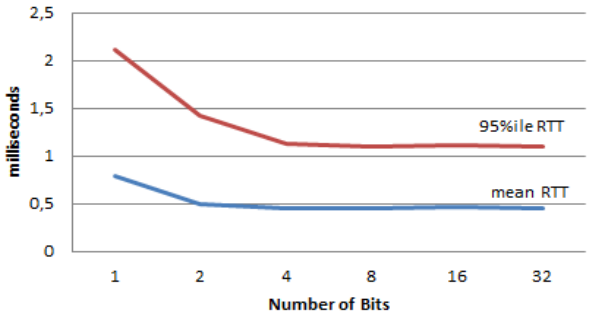

Figure 5: Variation of round-trip time for different number of bits used for the congestion signal.

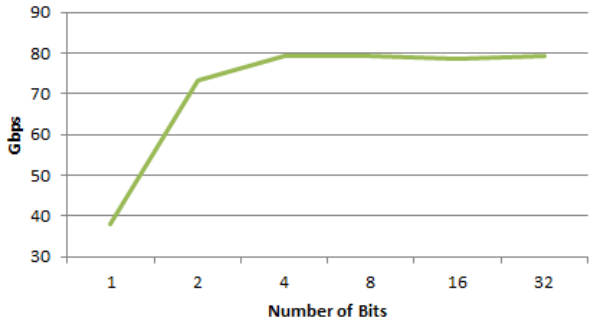

Figure 6: Variation of throughput for different number of bits used for the congestion signal.

bits. Beyond four bits there is a diminishing return which means 16 different quantization levels are likely quite sufficient to minimize congestion for MAX-gradient based algorithm in this scenario. Figure 6 confirms that throughput is stable with four or more bits. A more thorough exploration of different topologies and different queue sizes would help us determine the right number of bits in a particular network. Typically, larger buffers would require more bits for quantization in order to keep the same level of precision. With programmable switches, of course, we do not need all networks to use the same quantization.

5 DISCUSSION

How beneficial is focusing on the bottleneck queue? We found that SUM formulation is too conservative because of noise in the signal. SUM values contain both the signal from the bottleneck (good) and signals from all the non-bottleneck queues (bad). By trying different functions of queue occupancy vector, we tested whether the problem can be mitigated by paying less attention to non-bottleneck signals. Indeed it does: if we give more weight to the larger queue values (e.g. with SUM_SQU and MAX), the client achieves higher throughput. This not only justifies why MAX is a good signal; it also explains why the algorithms in [13] and [1] benefit greatly by directly using maximum queue occupancy information. Similarly, HPCC [20] proposes to exploit INT information for calculating the utilization ratio of the links and chooses the highest utilization, i.e. bottleneck, for deciding on the sending window size.

Focusing on the bottleneck makes intuitive sense. If non-bottleneck queues get longer, it is not because the source is sending too fast; it's because of a cross-flow sharing the queue. Then, the source should not decrease the sending rate and let the cross-flow handle the congestion at that queue (since it will be the bottleneck for the cross-flow). SUM clients react to the (irrelevant) signal from the non-bottleneck queues, whereas the SUM_SQU signal reduces the effect of the erroneous information and MAX ignores it altogether.

What would make a better control algorithm? The goal of this paper is not to propose an ideal congestion control algorithm; our goal is to point out the feasibility and effectiveness of using queue occupancy as the congestion signal. We doubt that Timely's gradient-based algorithm with the MAX signal is the best approach. We believe that considerable amount of work remains to design the ideal congestion control algorithm using this new signal.

Our results question the effectiveness of using RTT as the primary congestion signal. We acknowledge that Timely's focus on delay and DCQCN's focus on ECN are for practical considerations. RTT and ECN are already well measured, widely-used, and standardized signals leading to easier deployments of new algorithms. However, queue occupancy information is now becoming a freely available signal too.

While queue occupancy stamping is quite promising, it does not prevent the use of other signals too. A congestion control algorithm might also measure RTT in addition to queue occupancy stamps, similar to combining RTT and ECN as in [27]. While maximum queue occupancy helps the end-host to converge to max-min fair rate allocation on bottleneck, RTT value could be used to determine proportionally fair resource allocation. A flow with high RTT but low bottleneck queue occupancy could still be throttled to prevent over-utilization of other network resources.

We propose making queue occupancy stamping universal and standardized, so that (over time) it becomes a globally available signal provided by every router and switch in a network. This, would allow us to design, say, a PI controller (e.g. as used in [29]) based on a pre-determined target queue occupancy where all competing flows try to minimize the exact same variable. In return, we may even be able to come up with theoretical bounds on the required buffer sizes, regardless of the path lengths and number of coexisting flows,

allowing us to optimize our resources even for large scale heterogeneous infrastructures. This would not be possible with RTT, because the values vary notably depending on the distance between the end hosts.

6 CONCLUSION

It is natural to be skeptical about new, proposed fundamental changes to how the Internet operates, particularly when it has operated successfully for decades without these changes. But at the risk of being considered unfair and outrageous, this paper argues that by **not** stamping packets with queue occupancy, Internet switches and routers deprive end-hosts of the most precise measure of congestion possible, *even though the switches and routers already know it.* We believe that, with hindsight, its omission was an oversight. Had it been there all along, we would not have needed to rely on noisy, imprecise signals such as loss, RTT or ECN, and today's congestion control algorithms would be quite different. Now that queue occupancy is readily available, and switches are becoming programmable, it is natural to expect that queue occupancy information will become widely used by new congestion control algorithms; to start with, in proprietary data centers, but eventually in other parts of the networks as well. We encourage the community to explore new congestion control algorithms that exploit this new signal.

REFERENCES

[1] Hasnain Ahmed and Muhammad Junaid Arshad. 2019. Buffer Occupancy-Based Transport to Reduce Flow Completion Time of Short Flows in Data Center Networks. *Symmetry* 11 (05 2019), 646. https://doi.org/10.3390/sym11050646

[2] Mohammad Alizadeh, Albert Greenberg, David A. Maltz, Jitendra Padhye, Parveen Patel, Balaji Prabhakar, Sudipta Sengupta, and Murari Sridharan. 2010. Data Center TCP (DCTCP). *SIGCOMM Comput. Commun. Rev.* 40, 4 (Aug. 2010), 63–74. https://doi.org/10.1145/1851275.1851192

[3] BarefootNetworks. 2019. Tofino2: Second-generation of World's fastest P4-programmable Ethernet switch ASICs. (June 2019). https://www.barefootnetworks.com/products/brief-tofino-2/

[4] Pat Bosshart, Dan Daly, Glen Gibb, Martin Izzard, Nick McKeown, Jennifer Rexford, Cole Schlesinger, Dan Talayco, Amin Vahdat, George Varghese, and David Walker. 2014. P4: Programming Protocol-independent Packet Processors. *SIGCOMM Comput. Commun. Rev.* 44, 3 (July 2014), 87–95. https://doi.org/10.1145/2656877.2656890

[5] Pat Bosshart, Glen Gibb, Hun-Seok Kim, George Varghese, Nick McKeown, Martin Izzard, Fernando Mujica, and Mark Horowitz. 2013. Forwarding Metamorphosis: Fast Programmable Match-action Processing in Hardware for SDN. In *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM (SIGCOMM '13)*. ACM, New York, NY, USA, 99–110. https://doi.org/10.1145/2486001.2486011

[6] P. Bosshart, CTO Barefoot Networks. 2019. Personal Communication. personal communication. (2019).

[7] Lawrence S. Brakmo, Sean W. O'Malley, and Larry L. Peterson. 1994. TCP Vegas: New Techniques for Congestion Detection and Avoidance. In *Proceedings of the Conference on Communications Architectures, Protocols and Applications (SIGCOMM '94)*. ACM, New York, NY, USA, 24–35. https://doi.org/10.1145/190314.190317

[8] Broadcom. 2019. BCM56880: High-Capacity StrataXGS Trident 4 Ethernet Switch Series. (June 2019). https://www.broadcom.com/products/ethernet-connectivity/switching/strataxgs/bcm56880-series

[9] Rory Browne, Andrey Chilikin, and Tal Mizrahi. 2019. Key Performance Indicator (KPI) Stamping for the Network Service Header (NSH). RFC 8592. (May 2019). https://doi.org/10.17487/RFC8592

[10] Neal Cardwell, Yuchung Cheng, C. Stephen Gunn, Soheil Hassas Yeganeh, and Van Jacobson. 2017. BBR: Congestion-based Congestion Control. *Commun. ACM* 60, 2 (Jan. 2017), 58–66. https://doi.org/10.1145/3009824

[11] Cisco. 2018. UADP - The Powerhouse of Catalyst 9000 family. (December 2018). https://community.cisco.com/t5/networking-blogs/uadp-the-powerhouse-of-catalyst-9000-family/ba-p/3764605

[12] D. Clark. 1988. The Design Philosophy of the DARPA Internet Protocols. In *Symposium Proceedings on Communications Architectures and Protocols (SIGCOMM '88)*. ACM, New York, NY, USA, 106–114. https://doi.org/10.1145/52324.52336

[13] Nandita Dukkipati. 2008. *Rate Control Protocol (RCP): Congestion Control to Make Flows Complete Quickly.* Ph.D. Dissertation. Stanford University, Stanford, CA, USA. Advisor(s) Mckeown, Nick. AAI3292347.

[14] S. Floyd and T. Henderson. 1999. The NewReno Modification to TCP's Fast Recovery Algorithm. RFC 2582. (1999). https://doi.org/10.17487/RFC2582

[15] Sally Floyd and Van Jacobson. 1993. Random Early Detection Gateways for Congestion Avoidance. *IEEE/ACM Trans. Netw.* 1, 4 (Aug. 1993), 397–413. https://doi.org/10.1109/90.251892

[16] Sally Floyd, Dr. K. K. Ramakrishnan, and David L. Black. 2001. The Addition of Explicit Congestion Notification (ECN) to IP. RFC 3168. (Sept. 2001). https://doi.org/10.17487/RFC3168

[17] Teerawat Issariyakul and Ekram Hossain. 2011. Introduction to Network Simulator NS2. (2011).

[18] V. Jacobson. 1988. Congestion Avoidance and Control. In *Symposium Proceedings on Communications Architectures and Protocols (SIGCOMM '88)*. ACM, New York, NY, USA, 314–329. https://doi.org/10.1145/52324.52356

[19] Changhoon Kim, Parag Bhide, Ed Doe, Hugh Holbrook, Anoop Ghanwani, Dan Daly, Mukesh Hira, and Bruce Davie. 2016. Inband Network Telemetry (INT). (2016). https://p4.org/assets/INT-current-spec.pdf

[20] Yuliang Li, Rui Miao, Hongqiang Harry Liu, Yan Zhuang, Fei Feng, Lingbo Tang, Zheng Cao, Ming Zhang, Frank Kelly, Mohammad Alizadeh, and Minlan Yu. 2019. HPCC: High Precision Congestion Control. In *Proceedings of the ACM Special Interest Group on Data Communication (SIGCOMM '19)*. ACM, New York, NY, USA, 44–58. https://doi.org/10.1145/3341302.3342085

[21] Radhika Mittal, Vinh The Lam, Nandita Dukkipati, Emily Blem, Hassan Wassel, Monia Ghobadi, Amin Vahdat, Yaogong Wang, David Wetherall, and David Zats. 2015. TIMELY: RTT-based Congestion Control for the Datacenter. *SIGCOMM Comput. Commun. Rev.* 45, 4 (Aug. 2015), 537–550. https://doi.org/10.1145/2829988.2787510

[22] A. Munir, I. A. Qazi, Z. A. Uzmi, A. Mushtaq, S. N. Ismail, M. S. Iqbal, and B. Khan. 2013. Minimizing flow completion times in data centers. In *2013 Proceedings IEEE INFOCOM*. 2157–2165. https://doi.org/10.1109/INFCOM.2013.6567018

[23] Pierre-Francois Quet, Sriram Chellappan, A. Durresi, M. Sridharan, Hitay Ozbay, and Raj Jain. 2002. Guidelines for optimizing Multi-Level ECN, using fluid flow based TCP model. In *Proceedings of ITCOMM 2002 Quality of Service over Next Generation Internet.* Boston, MA, USA.

[24] K. K. Ramakrishnan and Sally Floyd. 1999. A Proposal to add Explicit Congestion Notification (ECN) to IP. *RFC* 2481 (1999), 1–25. https://doi.org/10.17487/RFC2481

[25] D. Shan and F. Ren. 2017. Improving ECN marking scheme with micro-burst traffic in data center networks. In *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*. 1–9. https://doi.org/10.1109/INFOCOM.2017.8057181

[26] Keith Winstein and Hari Balakrishnan. 2013. TCP Ex Machina: Computer-generated Congestion Control. In *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM (SIGCOMM '13)*. ACM, New York, NY, USA, 123–134. https://doi.org/10.1145/2486001.2486020

[27] Gaoxiong Zeng, Wei Bai, Ge Chen, Kai Chen, Dongsu Han, and Yibo Zhu. 2017. Combining ECN and RTT for Datacenter Transport. In *Proceedings of the First Asia-Pacific Workshop on Networking*. ACM, 36–42.

[28] Yibo Zhu, Haggai Eran, Daniel Firestone, Chuanxiong Guo, Marina Lipshteyn, Yehonatan Liron, Jitendra Padhye, Shachar Raindel, Mohamad Haj Yahia, and Ming Zhang. 2015. Congestion control for large-scale RDMA deployments. *ACM SIGCOMM Computer Communication Review* 45, 4 (2015), 523–536.

[29] Yibo Zhu, Monia Ghobadi, Vishal Misra, and Jitendra Padhye. 2016. ECN or Delay: Lessons Learnt from Analysis of DCQCN and TIMELY. In *Proceedings of the 12th International on Conference on Emerging Networking EXperiments and Technologies (CoNEXT '16)*. ACM, New York, NY, USA, 313–327. https://doi.org/10.1145/2999572.2999593