On the Future of Congestion Control for the Public Internet

Lloyd Brown[1] Ganesh Ananthanarayanan[2] Ethan Katz-Bassett[3] Arvind Krishnamurthy[4]
Sylvia Ratnasamy[1] Michael Schapira[5] Scott Shenker[1,6]
[1] UC Berkeley [2] Microsoft Research [3] Columbia University [4] University of Washington
[5] Hebrew University of Jerusalem [6] ICSI

Abstract

The conventional wisdom requires that all congestion control algorithms deployed on the public Internet be TCP-friendly. If universally obeyed, this requirement would greatly constrain the future of such congestion control algorithms. If partially ignored, as is increasingly likely, then there could be significant inequities in the bandwidth received by different flows. To avoid this dilemma, we propose an alternative to the TCP-friendly paradigm that can accommodate innovation, is consistent with the Internet's current economic model, and is feasible to deploy given current usage trends.

ACM Reference Format:
Lloyd Brown, Ganesh Ananthanarayanan, Ethan Katz-Bassett, Arvind Krishnamurthy, Sylvia Ratnasamy, Michael Schapira, Scott Shenker. 2020. On the Future of Congestion Control for the Public Internet. In *Proceedings of the 19th ACM Workshop on Hot Topics in Networks (HotNets '20), November 4–6, 2020, Virtual Event, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3422604.3425939

1. INTRODUCTION

The problem of congestion control has probably inspired more papers than any other topic in networking, resulting in a never-ending stream of new proposals [1–11] as well as significant advances in how to create [12, 13] and evaluate them [14, 15]. This paper stands apart from this literature, as it does not present any specific new designs or methodologies. Instead, it discusses an overall framework that would make the Internet more open to these congestion control innovations as they arise.

In doing so, this paper considers the problem of congestion control only in the context of the public Internet. We do not address congestion control solutions that are *only* run in settings where (i) an operator has control over what is deployed, and (ii) flows only interact with other similarly controlled flows. There has been great progress in developing and deploying congestion control algorithms (hereafter, CCAs) in controlled settings such as private datacenters [16–24] precisely because their designs do not affect, nor are affected by, CCAs operated by external entities.

To the contrary, we will focus on congestion control as it plays out on the public Internet, where CCAs adopted by many independent entities freely interact with each other. Note that our purview includes all endpoint CCAs that are deployed on hosts in private networks but which communicate with other endpoints that are reached by crossing the public Internet, because such flows may interact with flows using CCAs chosen by others. In this open setting, where there is no central control over which CCAs are deployed, we consider the question of how to tolerate diversity in CCAs, which is necessary for enabling congestion control innovations to be freely deployed.

To review, the modern era of congestion control started with the seminal works of Jacobson [25, 26] and Ramakrishnan and Jain [27], and has culminated with TCP Cubic [28] (the default in Linux) and other related CCAs [29–32] that follow a simple paradigm. When they detect signals of congestion such as packet drops or increasing delays, they reduce their sending rate. When no such congestion signals are detected, they gradually increase their rate.

The interactions on the public Internet between a set of long-running flows with various CCAs results in an allocation of bandwidth among those flows. For instance, for TCP New Reno flows, the standard TCP equation expresses the bandwidth received by long-running flows as a function of loss rate and RTT [33–35]. We have long known that more aggressive CCAs receive more bandwidth than less aggressive ones; *e.g.,* changing the constants in the TCP AIMD algorithm to increase more quickly or reduce more slowly results in more bandwidth. If individual users could alter their CCAs to be more aggressive, it could lead not only to unfairness but also to a networking version of the Tragedy of the Commons [36] as CCAs get increasingly more aggressive.

The traditional approach to minimizing unfairness and preventing a spiral into ever-more aggressive CCAs has been to demand all newly deployed CCAs be TCP-friendly. The notion of TCP-friendliness arose out of early discussions in the End-to-End Task Force and other venues about the general concept of "Network Friendliness" (*i.e.,* behaviors that did not cause harm to the Internet), and was most articulately formulated and forcefully advocated by Sally Floyd who wrote in [37]: "We say a flow is TCP-friendly if its arrival rate does not exceed the arrival of a conformant TCP connection in the same circumstances."

The TCP-friendliness approach has, for the past twenty-plus years, been widely adhered to on the public Internet. This compliance has been accompanied by significant research on (i) various improvements to TCP (such as in [28, 29, 38]) that are friendly with the default versions, and (ii) developing non-TCP CCAs (for streaming media and other purposes) that are TCP-friendly [39]. However, not all research has resulted in TCP-friendly CCAs. Here we mention two such recent proposals: BBR [1] and PCC [2].

BBR mostly avoids using packet drops as a congestion signal and instead computes a desired rate by using delay measurements to identify the bottleneck bandwidth and the round-trip propagation

HotNets '20, November 4–6, 2020, Virtual Event, USA

ACM ISBN 978-1-4503-8145-1/20/11.
https://doi.org/10.1145/3422604.3425939

time. BBR then tries to sustain this desired rate even in the presence of more conventional loss-based CCAs that would otherwise crowd it out. Through these techniques, BBR virtually eliminates bufferbloat and quickly achieves full bandwidth utilization on uncongested paths. BBR has been deployed by Google on YouTube and many other services, and is used by Netflix and recent versions of the Linux and FreeBSD kernels; thus, it is in active use on the public Internet.

PCC goes even further away from the traditional notion of congestion signals and focuses on optimization by correlating its empirically observed performance (in terms of throughput and packet drops) with its actions (in terms of its sending rate). It then chooses the actions that result in the highest performance. PCC has been shown to achieve better fairness, stability, and performance than traditional TCPs.

While BBR's designers intended for BBR to be TCP-friendly, more recent results have shown otherwise. As noted in [40], several research groups [3, 41, 42] have "observed a single BBR flow consuming a fixed 35-40% of link capacity when competing with as many as 16 Cubic flows." Reference [43] also finds extreme unfairness when BBR competes with Cubic. In contrast, PCC's TCP-friendliness has thus far received very little attention, but the authors of [3] conjecture that "it is fundamentally hard for any loss-based protocol to achieve consistently high performance and at the same time be fair towards TCP."

The lack of TCP-friendliness (verified in BBR and conjectured for the loss-based variants of PCC, Allegro [2] and Vivace-loss [3]) is not because of careless design but results from a fundamental tradeoff. As shown for a simple model of loss-based congestion control [44], the goal of TCP-friendliness is provably incompatible with the goals of achieving certain levels of efficiency (utilizing full path capacity) and rapid ramp-up (not taking too long to achieve high efficiency). A similar result holds when requiring robustness (being able to tolerate a certain level of non-congestion losses without losing too much efficiency).

TCP-friendliness also imposes another kind of constraint: not only do CCAs have to treat traditional TCPs nicely, but they also have to function well in their presence. Notably, delay-based algorithms often suffer from being squeezed out by traditional TCPs that fill the buffer before backing off, causing an increase in delay. Delay-based CCAs tend to interpret these delay increases as congestion, while loss-based TCPs do not detect congestion until the buffer overflows. As an example, the delay-based Vivace-latency suffers from being overly conservative when competing with buffer-filling CCAs like TCP Cubic (see Figure 10 in [3]). Avoiding this phenomenon strongly influenced BBR's design, at the loss of TCP-friendliness. Thus, TCP-friendliness poses two challenges for new CCAs: being friendly to TCPs despite trying to maximize bandwidth while, at the same time, not being overly conservative despite trying to minimize delays. The need to navigate between these two requirements narrows the room for innovation.

These observations make it clear that TCP-friendliness greatly constrains how we handle congestion in the Internet. That is, if we were to remove the constraint of CCAs being TCP-friendly then we could accomplish many more desirable congestion control goals, such as rapidly achieving full path capacity and being impervious to non-congestion losses. In addition, the deployment of BBR by two of the largest Internet actors in the world shows that the commitment to only deploying TCP-friendly CCAs has broken down. Thus, we can view the era of universal TCP-friendliness from a theoretical perspective as overly constraining, or from a practical viewpoint as essentially over. In either case, we should look for an alternative, which is the focus of our paper.

We begin in Section 2 by discussing and ultimately rejecting the two previous alternatives to the TCP-friendly paradigm, and then propose a new alternative. In Section 3 we describe a design implementing our approach and then in Section 4 we discuss the impact of current trends on our design choices.

2. ALTERNATIVE APPROACHES

2.1 Previous Proposals

The literature on congestion control contains two main alternatives to TCP-friendliness. The first, as articulated in the context of Fair Queueing [45, 46] and its many descendants [47–53], is to have every potentially congested router provide isolation between flows; by isolation we mean that no matter how aggressively other flows send, each flow is guaranteed a fair share of the bandwidth. This share could be based on a notion of equality between flows, or there could be a weight associated with each flow that determines its relative share, but the focus of this per-flow-fairness paradigm was not the precise values of these weights but to protect flows from the actions of others. This removed the need for a universally mandated CCA; because of isolation, flows could freely adopt the CCA that best met their needs – for instance trading off bandwidth to ensure fewer losses, or the reverse – without worrying about imposing harm on other flows or being harmed by others.

This approach can be implemented using various forms of packet scheduling [46, 51, 54] and/or selective packet dropping [47, 52]. While the initial proposals [46, 51] required per-flow queues and state, more recent designs [47, 52, 54] do not. In particular, core routers in [47] have no per-flow state and utilize a single FIFO queue. Thus, the isolation approach is implementable and could support ongoing innovation in CCAs, which makes it a potential alternative to TCP-friendliness.

However, the notion of per-flow-fairness is highly problematic. Initially there were concerns about how to define flows (should they be per source, per destination, or per source-destination pairs?), and how to assign weights in the weighted version of these algorithms. But Bob Briscoe, in [55], dismantled the religion of per-flow-fairness by observing that flows were not the economic actors on the Internet, and thus there was no reason to treat them equally (or in some weighted fashion). Briscoe's argument is that congestion control determines, to some extent, the allocation of bandwidth on the public Internet; thus, these allocations should be motivated by some underlying economic model, and per-flow-fairness had no such intellectual foundation. Note that Briscoe's critique also applies to TCP-friendliness since that too produces a form of per-flow-fairness. In replacing TCP-friendliness, we seek a more fundamentally grounded approach.

The other commonly espoused alternative is the Network Utilization Maximization (NUM) approach first articulated by Frank Kelly [56, 57]. In NUM, congestion signals serve as shadow prices

indicating the level of congestion the flow is encountering. In simple network models, if sources use these shadow prices to adjust their rate so as to maximize the utility of the flow minus the shadow cost incurred, then NUM achieves the socially optimal outcome (maximizing the sum of the utilities) at equilibrium.

There are many ways to apply Kelly's insights to the Internet. For example, one could require CCAs to all be self-optimizing based on these shadow prices, or turn these shadow prices into real costs and let CCAs adapt to monetary incentives, or use this NUM formulation to describe the utilities that would produce various currently used CCAs [58].

However, these various applications of the NUM approach all suffer from the fundamental problem that focusing on per-flow utilities, while firmly grounded in theoretical economics, is not consistent with the Internet's current economic model in which larger entities (not individual flows) contract for service from providers. In fact, the NUM approach of maximizing the sum of flow utilities is completely orthogonal to who has paid for service and what their resulting service expectations might be, which would be untenable for providers and users alike. To base congestion control on the NUM model would require replacing the current economic arrangements with ones that charged individual flows, so that payments could be tied to the services being provided.

2.2 Our Approach

In designing an approach that could be a long-lasting framework for congestion control on the public Internet, we begin by embracing Briscoe's point that congestion control should be grounded in some underlying economic foundations, and then extend this by requiring that these foundations be consistent with the Internet's current economic model (which is likely to change far more slowly than congestion control).

Both of these previous approaches failed this latter test, as they embraced flows as the relevant economic actors. Instead, the Internet's economic model revolves around purchasing network access, and then applying those access rights recursively: *i.e.,* a home user's packets are carried by her provider's network because she pays her provider directly, and then the next-hop provider carries her packets because her provider has an economic arrangement with that next-hop (which may be settlement-free peering, or a provider-customer relationship). The actors in these arrangements are the entities that purchase access, not individual flows.

Accordingly, our proposal extends these network access contracts (which currently provide for a certain level of sending and receiving traffic) to also ensure that outgoing traffic has certain relative rights (or shares) when it hits congestion. These *congestion-shares* are enforced by the isolation mechanisms mentioned previously (*i.e.,* packet scheduling or selective dropping) and produce bandwidth allocations as in weighted fair queueing [46]: *i.e.,* under congestion the resulting bandwidths for bottlenecked traffic are proportional to congestion-shares. Moreover, we apply these congestion-shares recursively, similar to how today's access agreements work. That is, when the home user's traffic in the example above hits congestion in her provider's network, the traffic is treated as having some congestion-share dictated by her agreement with her provider; but when her traffic hits congestion in the next-hop provider's network, it is treated as having the same congestion-share as her provider's traffic has in the next-hop domain. We will make this recursive behavior more precise in the next section.

Thus, our approach – which we call Recursive Congestion-Shares (RCS) – is based on the isolation approach, except rather than enforcing shares on the granularity of flows, they are enforced on the granularity of these access agreements, and these access agreements are applied recursively. RCS would let everyone adopt the CCA of their choice – such as BBR, PCC, or future innovations – while the bandwidth allocations under congestion would be determined not by the aggressiveness of their CCA but by their congestion-shares, which reflect the underlying economic arrangements that finance the Internet infrastructure.

The prior work that is most related to our approach is FairCloud [59] which only applies within datacenters, but does assign tenant- or flow-specific shares that dictate how to allocate bandwidth under congestion. FairCloud considers a wider class of policies than we do here (sender and receiver payments are considered, as is proximity). We ignore receiver arrangements since (as we discuss later) the congestion-shares are only applied at egress points (where all aggregates share the same receiver), and do not consider proximity, but do apply these policies recursively (as is necessary in the public Internet setting, but not for internal cloud allocations).

3. THE RCS DESIGN

We present RCS in more detail by addressing a few basic questions, briefly discussing various other technical issues, and then examining how RCS relates to network neutrality. In what follows, we consider a single provider's network, which we refer to as a domain, with a set of neighboring entities (NEs) each with their own attachment point to the network. These NEs can be home users, cellular users, enterprises, or peering domains (which themselves could be customers, providers, or settlement-free peers); our design need not distinguish between these different classes of NEs, as they all have economic arrangements with the domain which result in some congestion-share. Each NE represents both an ingress point and an egress point, and we will refer to the traffic coming from an NE into a domain as an aggregate, since it can be comprised of many flows.

3.1 Where Is Isolation Enforced?

In order for RCS to be effective, it should enforce isolation at the major congestion points. Enforcing isolation requires: (i) identifying which packets belong to which aggregates, and (ii) knowing the congestion-shares of each aggregate. To reduce this information-sharing burden (since it would be difficult for all routers in a domain to have this information), and to cleanly separate how a domain manages its internal routers from how it implements RCS (see Section 3.5), we choose to only enforce isolation at domain egress points, at least in transit domains (we consider originating/terminating domains separately in Section 3.4). That is, when the convergence of traffic from a domain's many ingress points results in congestion at an egress point, the domain handles the various traffic aggregates according to their congestion-shares using the aforementioned isolation mechanisms. To only enforce at domain edges presumes that these are the primary places where congestion occurs. This is already the conventional wisdom among

many we have talked to, but we have found it difficult to verify this through measurement. Accordingly, here we are not considering the statement that congestion mainly occurs at transit domain edges as an empirical fact but as an imposed expectation on domains; to be consistent with RCS, transit domains should be managed in such a way that significant internal drops do not occur.

3.2 How Are Congestion-Shares Computed?

Consider a set of domains (instances denoted by α) and NEs (instances denoted by i). For clarity of notation, each NE is associated with one domain; an entity with access in two different domains is seen as two different NEs. We let $N(\alpha)$ represent the set of NEs of domain α.

Each NE of α has some financial arrangement with the domain that dictates sending and receiving rates, and perhaps SLAs. In our proposal, we extend this arrangement to specify a congestion-share s_i^α; s_i^α need not be tied to any other parameter in the agreement (*i.e.,* it is not necessarily tied to the sending or receiving rates or the SLA, see Section 3.5), but we expect in many cases there will be some correlation: NEs who have contracted for higher bandwidth rates will likely have larger congestion-shares. However, this congestion-share may not be made explicitly visible to the NE since there is no way for an NE to verify (based on external behavior) how it is being treated relative to other NEs.

In what follows we only consider traffic entering and then exiting a domain; later we discuss the case where traffic originates or terminates within a domain. The traffic matrix for the domain α is denoted by $t_{i,j}^\alpha$, which is only defined for $i, j \in N(\alpha)$ and describes the short-term average rate of traffic entering the domain from NE i and leaving the domain towards NE j. Let $T_i^\alpha = \sum_j t_{i,j}^\alpha$ be the short-term average of the total traffic entering at i, so $\frac{t_{i,j}^\alpha}{T_i^\alpha}$ is the fraction of i's traffic exiting at j.

RCS weights the original congestion-shares of each NE i by the fraction of their traffic that is exiting at a particular egress point. Thus, when enforcing isolation at egress j, the isolation mechanism will use the proportional shares $p_{i,j}^\alpha = \frac{t_{i,j}^\alpha}{T_i^\alpha} s_i^\alpha$. Note that $\sum_j p_{i,j}^\alpha = s_i^\alpha$, so we are merely apportioning i's total congestion-share among the various egress points according to the amount exiting there.

Enforcing isolation using these shares results in the desired outcome if all traffic only hits congestion on their first egress point. However, consider the case where two NEs i, j of domain α have very different proportional shares $p_{i,k}^\alpha$ and $p_{j,k}^\alpha$ for some egress k where they encounter no congestion. Assume that egress k leads to domain β, and both these aggregates leave domain β through egress l. If they encounter congestion at egress l then these packets are all treated as part of a single aggregate with congestion-share $p_{k,l}^\beta$, and no distinction is made between NEs i and j. Thus, whatever additional congestion rights NE i had over NE j (or vice versa) have been rendered invisible.

A more theoretically sound way to respect congestion-shares when passing through multiple transit domains would be to employ hierarchical isolation (as defined in [48, 60]). This would require a domain α to know not just the proportional shares of aggregates entering its network, but all the proportional shares at all previous transit domains. This would allow, in the example above, domain β to enforce the relative shares of $p_{i,k}^\alpha$ and $p_{j,k}^\alpha$ within the aggregate of traffic from domain α leaving domain β at egress l.

Providing this level of information about congestion-shares would be difficult. However, as we discuss in Section 4, we think recent trends have made it unnecessary to enforce hierarchical isolation. Note that this design choice (of not using hierarchical isolation) has the side-benefit of making all congestion-shares, and the mechanisms that enforce them, purely internal; each domain can independently make agreements with its NEs about the congestion-shares they are assigned within that domain, and then enforce them upon egress. No broader domain-to-domain agreements or interactions are required.

3.3 How Is RCS Implemented?

There are three challenges to be met in implementing RCS. The first is to minimize dropping within transit domains, so enforcement at egress is the main way in which congestion is resolved; all transit domains already strive towards this goal. The second is to deploy the necessary packet scheduling and/or selective dropping mechanism on egress routers. These mechanisms need only be applied at the fairly rough granularity of aggregates, rather than applied to individual flows; such mechanisms are already available on many commercial routers. Cases that involve many aggregates (such as in a large-scale access network), where having a queue per-aggregate is not feasible, can be handled with approaches such as AFD [52] where a single FIFO queue is used.

The third required element is that the router at egress j in domain α needs to know the proportional shares $p_{i,j}^\alpha$ for all of the domain's ingress points i, and to be able to identify which packets belong to which ingress aggregate. We assume that the congestion-shares are updated perhaps on the order of minutes, so they can easily be shared through some edge-to-edge protocol between routers within a domain (as could be done by using iBGP, or RSVP-TE, or OSPF attributes). If the aggregates belonging to ingress points can easily be identified by a set of source prefixes (as would happen in many provider networks), then this information can be exchanged by the edge-to-edge protocol. However, if (as may be the case in some transit networks) the aggregates cannot be easily described by source prefixes, then one can use ingress-to-egress MPLS or tunneling protocols to attach a label identifying the ingress point to each packet.

Above we implicitly assumed that ingress points knew the egress of each packet, and thus could compute the proportional shares. If the internal routing is such that predicting the egress point cannot be done at ingress, then the ingress points can merely distribute the values T_i^α and s_i^α; each egress point j can measure $t_{i,j}^\alpha$ and compute the proportional congestion shares $p_{i,j}^\alpha$.

3.4 What Happens Within Originating and Terminating Domains?

For transit domains we assume that most congestion happens at egress points, and use the congestion-shares that the aggregates have when entering to resolve congestion when exiting. This approach does not apply to private networks (*e.g.,* homes, enterprises, content providers) where much traffic originates and terminates, since when traffic arises internally there is no service agreement

to guide how it should be treated when exiting. We leave it to private networks to manage this themselves by either (i) requiring the uniform use of a single CCA (resulting in the kinds of allocations that TCP-friendliness produces) or (ii) enforcing isolation across different flows with weights set according to some internal policy (essentially an internal version of congestion-shares for flows), or (iii) a combination of the two (*e.g.,* internal hosts are broken into groups, with each group having compatible CCAs and isolation is enforced between the groups according to some policy-driven weights).

It is useful to separate two cases: private networks with mostly clients (*e.g.,* an enterprise network) and private networks with mostly servers (*e.g.,* a content provider's internal network). In the former case, most traffic is inbound and the inbound congestion is handled by the isolation mechanisms at the egress point of the provider domain. However, in some cases (*e.g.,* provider networks servicing many homes) there can be internal congestion close to where the customer access points are (*e.g.,* at cable modem termination systems or CMTSs in cable networks). To provide isolation at such points, domains could simply use per-flow-fairness; a more sophisticated approach would be to first enforce isolation using the contracted receive rates of the home customers, and then (in a hierarchical fashion within each customer's share) enforce isolation using the congestion-shares of the incoming aggregates. This is not much different from current practice, where CMTSs use token-buckets to enforce receiver contracts, and then employ some form of weighted isolation between homes (with the weights tied to the level of contracted bandwidth).

For the occasional outgoing congestion in these client-heavy networks, some form of per-flow-fairness might be adequate (as the purpose of this occasional isolation enforcement is not to ensure an economically justified fairness, just to make sure that aggressive CCAs do not trample less aggressive ones).

In server-heavy private networks where most traffic is outbound, isolation enforcement may be frequently necessary at the domain's egress point. Here, any of the three options listed above would apply. Since the content provider knows the semantics of their application, as well as the identity of their customers, they can make resource decisions based on application-level and customer-relevant factors (*e.g.,* which flows can tolerate loss and which cannot, who has paid for what level of application service) that lie outside our scope.

3.5 Other Issues

QoS, TE, and SLAs: ISPs use internal traffic management mechanisms, involving both packet forwarding (QoS) and routing (TE), to improve the overall performance of their network and to meet individual customer SLAs. We have purposely designed RCS to be largely orthogonal to those mechanisms, by mainly enforcing congestion-shares at egress points, rather than internally. However, domains must ensure that congestion-shares assigned to their customers are sufficient to meet their SLAs at egress points.

Congestion signals: Our proposal is agnostic to the details of CCAs, but some CCAs require explicit congestion signals from routers, and the question is whether RCS supports such algorithms. There are two challenges here. The first is that CCAs requiring explicit feedback could not be deployed until the necessary algorithms were installed at all enforcement points. Our goal here is to support CCA innovation, but we don't see how to speed up the deployment of such explicit mechanisms. Second, explicit feedback is acted on by each flow's CCA, yet the isolation mechanisms are geared towards controlling the bandwidth shares of aggregates without necessarily knowing about the individual flows. As a result, RCS does not support such explicit signaling (except perhaps ECN [61] which is already a standard) and only relies on the implicit signals of packet delays and loss. However, we do not preclude using explicit signaling if future advances in this area resulted in a widely deployed and general design.

Congestion in low-bandwidth access networks: In access networks with very limited bandwidth, like cellular, we expect that some degree of isolation might be enforced to prevent congestion collapse. In such cases, per-flow fairness might be adequate, rather than worrying about more complicated solutions.

Behavior at IXPs: We treat IXPs as a transparent mesh rather than a separate AS-hop. The only complication is that a domain α can then potentially receive incoming traffic from many NEs at the same ingress point, so that enforcing isolation at egress is not sufficient to prevent congestion at α's ingress. This may require the IXP to enforce isolation upon ingress according to weights assigned by α, which is feasible given the centralized nature of IXPs.

Equity: Just as with any network prioritization proposal, the design should not squeeze out those with fewer financial resources. The dynamic range of congestion-shares could be limited so that congestion would not result in significant harm to any class of users.

3.6 How Does RCS Relate to Network Neutrality?

We end this section by noting that some view our proposal as violating network neutrality and relying too heavily on economics as a rationale for allocating bandwidth. In response to this critique, we first observe that access to Internet bandwidth is already driven by economics: customers purchase access with specific send and receive rates, and their providers enter into economic agreements with their peers. Thus, the bandwidth a customer's traffic receives today already depends on the set of economic agreements along the path. Our proposal applies this information more systematically than today, but our proposal is consistent with current practice (*e.g.,* as mentioned before, under contention CMTSs can allocate bandwidth to customers based on their level of contracted bandwidth).

The issue of network neutrality is complicated by the fact that there is no universally-accepted definition of the term. If network neutrality means that all packets are treated equally, then our proposal certainly violates that standard. However, Misra in [62, 63] discusses a range of possible definitions and proposes one where network neutrality is the proclamation that the "Internet is a platform where ISPs provide no competitive advantage to specific apps/services, either through pricing or QoS." Our proposal is certainly consistent with this formulation of network neutrality which specifically allows non-discriminatory differential QoS and pricing.

This dispute over definitions can only be settled by their implications for the Internet. Our paper addresses the looming reality where the TCP-friendliness no longer holds. If the Internet treated

all packets equally then the bandwidth achieved by flows from different customers would be a function of the aggressiveness of their CCA. This could result in increasingly aggressive CCAs and an overly congested Internet. Thus, in a world where TCP-friendliness has broken down, the Internet must provide some form of explicit bandwidth allocation, which then requires that packets be treated in a nonuniform manner. The question before us is not whether to discriminate between packets, but how bandwidth should be allocated.

Our proposal's core idea is that this allocation be based on a recursive set of bandwidth shares, as it seems closest to current practice on the Internet. This leaves significant room for different policies in how bandwidth shares are computed, which could range from being proportional to the level of contracted bandwidth to all customers receiving the same share. As observed above, equity considerations should play a role in how the shares are determined.

Thus, one should view our contribution as setting up a framework for how bandwidth allocations should be enforced in the Internet in a post-TCP-friendly world. This framework does not dictate a particular policy for computing congestion shares, and allows equity to be a consideration in such policies.

4. EXPLOITING CURRENT TRENDS

Recall from Section 3.2 that RCS would only need to enforce hierarchical isolation (considering congestion-shares from upstream transit domains) when traffic passes through multiple transit domains (*i.e.,* passing through four or more ASes: the originating domain, two or more transit domains, and the terminating domain). Here we argue why we think implementing hierarchical isolation is unnecessary.

The average AS path length between two random destinations on the Internet has remained fairly stable at around four to five hops, with only a slight decrease in recent years [64]. At the same time, the traffic patterns on the Internet have undergone two radical shifts. First, traffic is now dominated by cacheable video content and highly concentrated (with ten ASes being responsible for 70% of Internet traffic [65]). Second, most high-volume providers (such as Netflix, Google, Akamai, Amazon, and Facebook) have invested heavily in placing this content close to users by creating numerous PoPs (with caches) and peering directly with many other domains, causing Geoff Huston to proclaim the "Death of Transit?" [66]. As a result of these trends, some have estimated that roughly 70% of the Internet's traffic goes directly to the requesting client from either a nearby cache or from a neighboring domain [67].

While unfortunately we cannot directly verify this conjecture with publicly available measurements, we can consider a study [68] of routes from various cloud providers (which does not capture routes from caches). For traffic leaving Google Compute Engine towards client networks around the world, weighted by the number of clients in each network, the authors found that: (i) 62% of the requests traversed two ASes (*i.e.,* going straight from originating domain to terminating domain), (ii) 29% of the requests traversed three ASes (*i.e.,* passing through one transit domain), and (iii) 9% of the requests traversed four or more ASes (where knowledge of upstream congestion-shares would be needed to be completely faithful to the relative congestion-shares of aggregates).

The numbers in [68] for other cloud providers (IBM, Amazon, and Microsoft) had significantly higher percentages for this last category (between 39% and 51%). However, the methods in [68] are more conservative (*i.e.,* show longer paths) than those used in a 2015 study [69] which found significantly fewer long paths. Moreover, when one uses the same methodology on the 2015 and 2020 datasets, there is a clear trend towards the shortening of paths over time for this cloud-related traffic that is not being handled by nearby caches. In addition, the rising popularity of cloud/edge computing to support gaming, AR/VR, and the like suggests this trend will continue.

One might argue that we should use upstream congestion-shares in order for RCS to be independent of traffic trends. However, we choose not to because the difficulties – both mechanistic (in terms of algorithmic complexity and bandwidth overhead) and organizational (in terms of requiring interdomain standards) – needed to communicate these congestion-shares across providers are prohibitive. Thus, our proposal is to only use congestion-shares locally within each domain, which is easily implementable.

5. SUMMARY

We considered the problem of congestion control on the public Internet, and observed that the current TCP-friendly paradigm is in trouble. If strictly observed, TCP-friendliness would prevent us from deploying better CCAs; the tradeoff between TCP-friendliness and certain desirable properties has been theoretically established and confirmed by years of design experience. If TCP-friendliness is partially violated, which has already occurred with BBR, then radically unequal bandwidth allocations could result.

The two most promising alternatives to the TCP-friendly paradigm – per-flow-fairness and Network Utility Maximization – focus on flows as the fundamental unit of allocation, which is not consistent with the Internet's current economic model. In response, we have proposed a third alternative, one that involves linking access agreements to congestion-shares, and employing isolation mechanisms to enforce these congestion-shares at the main points of congestion. This would allow new CCAs to be used without requiring them to comply with TCP-friendliness or any other overarching constraints, thereby unleashing the community to develop and deploy a wide range of new congestion control designs.

The benefits of RCS seem clear, and the deployment requirements are modest, as many commercial routers already have the necessary isolation algorithms and disseminating the congestion-shares and aggregate-identification information is easily done. The most pressing question, then, is who would lead the charge to have RCS be adopted? Are there incentives for the relevant actors, such as the ISPs, to adopt these (completely internal) changes? This incentive question is particularly problematic since RCS would yield significant benefits only after being widely adopted. Thus, we end this paper not with a set of future technical challenges but with an open question about the incentives for deployment, where we are at a loss.

Acknowledgements: This work was funded in part by NSF Grants 1619377, 1817115, 1817116, 1704941, 1835253, and by grants from the Israel Science Foundation, Intel, VMware, Ericsson, Futurewei, and Cisco.

References

[1] Neal Cardwell, Yuchung Cheng, C. Stephen Gunn, Soheil Hassas Yeganeh, and Van Jacobson. BBR: Congestion-Based Congestion Control. ACM Queue, 2016.

[2] Mo Dong, Qingxi Li, Doron Zarchy, P. Brighten Godfrey, and Michael Schapira. PCC: Re-architecting Congestion Control for Consistent High Performance. NSDI, 2015.

[3] Mo Dong, Tong Meng, Doron Zarchy, Engin Arslan, Yossi Gilad, Brighten Godfrey, and Michael Schapira. PCC Vivace: Online-Learning Congestion Control. NSDI, 2018.

[4] Tong Meng, Neta Rozen Schiff, P. Brighten Godfrey, and Michael Schapira. PCC Proteus: Scavenger Transport And Beyond. SIGCOMM, 2020.

[5] Soheil Abbasloo, Chen-Yu Yen, and H. Jonathan Chao. Classic Meets Modern: A Pragmatic Learning-Based Congestion Control for the Internet. SIGCOMM, 2020.

[6] Yaxiong Xie, Fan Yi, and Kyle Jamieson. PBE-CC: Congestion Control via Endpoint-Centric, Physical-Layer Bandwidth Measurements. SIGCOMM, 2020.

[7] Keith Winstein, Anirudh Sivaraman, and Hari Balakrishnan. Stochastic Forecasts Achieve High Throughput and Low Delay over Cellular Networks. NSDI, 2013.

[8] Venkat Arun and Hari Balakrishnan. Copa: Practical Delay-Based Congestion Control for the Internet. NSDI, 2018.

[9] Radhika Mittal, Justine Sherry, Sylvia Ratnasamy, and Scott Shenker. Recursively Cautious Congestion Control. NSDI, 2014.

[10] Nandita Dukkipati and Nick McKeown. Why Flow-Completion Time is the Right Metric for Congestion Control. SIGCOMM, 2006.

[11] Dina Katabi, Mark Handley, and Charlie Rohrs. Congestion Control for High Bandwidth-Delay Product Networks. SIGCOMM, 2002.

[12] Keith Winstein and Hari Balakrishnan. TCP Ex Machina: Computer-Generated Congestion Control. SIGCOMM, 2013.

[13] Nathan Jay, Noga Rotman, Brighten Godfrey, Michael Schapira, and Aviv Tamar. A Deep Reinforcement Learning Perspective on Internet Congestion Control. ICML, 2019.

[14] Francis Y. Yan, Jestin Ma, Greg D. Hill, Deepti Raghavan, Riad S. Wahby, Philip Levis, and Keith Winstein. Pantheon: the Training Ground for Internet Congestion-control Research. ATC, 2018.

[15] Ranysha Ware, Matthew K Mukerjee, Srinivasan Seshan, and Justine Sherry. Beyond Jain's Fairness Index: Setting the Bar For The Deployment of Congestion Control Algorithms. HotNets, 2019.

[16] Gautam Kumar, Nandita Dukkipati, Keon Jang, Hassan M. G. Wassel, Xian Wu, Behnam Montazeri, Yaogong Wang, Kevin Springborn, Christopher Alfeld, Michael Ryan, David Wetherall, and Amin Vahdat. Swift: Delay is Simple and Effective for Congestion Control in the Datacenter. SIGCOMM, 2020.

[17] Ahmed Saeed, Varun Gupta, Prateesh Goyal, Milad Sharif, Rong Pan, Mostafa Ammar, Ellen Zegura, Keon Jang, Mohammad Alizadeh, Abdul Kabbani, and Amin Vahdat. Annulus: A Dual Congestion Control Loop for Datacenter and WAN Traffic Aggregates. SIGCOMM, 2020.

[18] Wenxue Cheng, Kun Qian, Wanchun Jiang, Tong Zhang, and Fengyuan Ren. Re-architecting Congestion Management in Lossless Ethernet. NSDI, 2020.

[19] Yuliang Li, Rui Miao, Hongqiang Harry Liu, Yan Zhuang, Fei Feng, Lingbo Tang, Zheng Cao, Ming Zhang, Frank Kelly, Mohammad Alizadeh, and Minlan Yu. HPCC: High Precision Congestion Control. SIGCOMM, 2019.

[20] Radhika Mittal, Vinh The Lam, Nandita Dukkipati, Emily Blem, Hassan Wassel, Monia Ghobadi, Amin Vahdat, Yaogong Wang, David Wetherall, and David Zats. TIMELY: RTT-Based Congestion Control for the Datacenter. SIGCOMM, 2015.

[21] Yibo Zhu, Haggai Eran, Daniel Firestone, Chuanxiong Guo, Marina Lipshteyn, Yehonatan Liron, Jitendra Padhye, Shachar Raindel, Mohamad Haj Yahia, and Ming Zhang. Congestion Control for Large-Scale RDMA Deployments. SIGCOMM, 2015.

[22] Jonathan Perry, Hari Balakrishnan, and Devavrat Shah. Flowtune: Flowlet Control for Datacenter Networks. NSDI, 2017.

[23] Inho Cho, Keon Jang, and Dongsu Han. Credit-scheduled Delay-bounded Congestion Control for Datacenters. SIGCOMM, 2017.

[24] Mohammad Alizadeh, Albert Greenberg, David A. Maltz, Jitendra Padhye, Parveen Patel, Balaji Prabhakar, Sudipta Sengupta, and Murari Sridharan. Data Center TCP (DCTCP). SIGCOMM, 2010.

[25] Van Jacobson and Michael J. Karels. Congestion Avoidance and Control. SIGCOMM, 1988.

[26] Van Jacobson. Modified TCP Congestion Avoidance Algorithm. End2end-interest Mailing List, 1990.

[27] K. K. Ramakrishnan and Raj Jain. A Binary Feedback Scheme for Congestion Avoidance in Computer Networks with a Connectionless Network Layer. SIGCOMM, 1988.

[28] Sangtae Ha, Injong Rhee, and Lisong Xu. CUBIC: A New TCP-Friendly High-Speed TCP Variant. SIGOPS, 2008.

[29] Lawrence S. Brakmo, Sean W. O'Malley, and Larry L. Peterson. TCP Vegas: New Techniques for Congestion Detection and Avoidance. SIGCOMM, 1994.

[30] Joel Sing and Ben Soh. TCP New Vegas: Improving the Performance of TCP Vegas Over High Latency Links. NCA, 2005.

[31] W. Stevens. TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms. RFC 2001, 1997.

[32] S. Floyd and T. Henderson. The NewReno Modification to TCP's Fast Recovery Algorithm. RFC 2582, 1999.

[33] Sally Floyd. Connections with Multiple Congested Gateways in Packet-Switched Networks Part 1: One-Way Traffic. 1991.

[34] Jitendra Padhye, Victor Firoiu, Don Towsley, and Jim Kurose. Modeling TCP Throughput: A Simple Model and Its Empirical Validation. SIGCOMM, 1998.

[35] Matthew Mathis, Jeffrey Semke, Jamshid Mahdavi, and Teunis Ott. The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm. SIGCOMM, 1997.

[36] W. F. Lloyd. Two Lectures on the Checks to Population. Oxford, 1832.

[37] Sally Floyd and Kevin Fall. Promoting the Use of End-to-End Congestion Control in the Internet. IEEE/ACM Trans. Netw., 1999.

[38] S. Floyd. HighSpeed TCP for Large Congestion Windows. RFC 3649, 2003.

[39] J. Widmer, R. Denda, and M. Mauve. A Survey on TCP-Friendly Congestion Control. Netwrk. Mag. of Global Internetwkg., 2001.

[40] Ranysha Ware, Matthew K. Mukerjee, Srinivasan Seshan, and Justine Sherry. Modeling BBR's Interactions with Loss-Based Congestion Control. IMC, 2019.

[41] Dominik Scholz, Benedikt Jaeger, Lukas Schwaighofer, Daniel Raumer, Fabien Geyer, and Georg Carle. Towards a Deeper Understanding of TCP BBR Congestion Control. IFIP, 2018.

[42] R. Ware, M. K. Mukerjee, J. Sherry, and S. Seshan. The Battle for Bandwidth: Fairness and Heterogeneous Congestion Control. NSDI Poster, 2018.

[43] Geoff Huston. "BBR, the new kid on the TCP block". 2017. URL https://blog.apnic.net/2017/05/09/bbr-new-kid-tcp-block/.

[44] Doron Zarchy, Radhika Mittal, Michael Schapira, and Scott Shenker. An Axiomatic Approach to Congestion Control. HotNets, 2017.

[45] J. Nagle. On Packet Switches with Infinite Storage. RFC 970, 1985.

[46] A. Demers, S. Keshav, and S. Shenker. Analysis and Simulation of a Fair Queueing Algorithm. SIGCOMM, 1989.

[47] Ion Stoica, Scott Shenker, and Hui Zhang. Core-Stateless Fair Queueing: Achieving Approximately Fair Bandwidth Allocations in High Speed Networks. SIGCOMM, 1998.

[48] Jon C. R. Bennett and Hui Zhang. Hierarchical Packet Fair Queueing Algorithms. SIGCOMM, 1996.

[49] A. K. Parekh and R. G. Gallager. A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: the Single-node Case. *IEEE/ACM Transactions on Networking*, 1993.

[50] Jon C. R. Bennett and Hui Zhang. WF2Q: Worst-Case Fair Weighted Fair Queueing. INFOCOM, 1996.

[51] M. Shreedhar and George Varghese. Efficient Fair Queueing Using Deficit Round Robin. SIGCOMM, 1995.

[52] Rong Pan, Lee Breslau, Balaji Prabhakar, and Scott Shenker. Approximate Fairness through Differential Dropping. SIGCOMM, 2003.

[53] P. E. McKenney. Stochastic Fairness Queueing. INFOCOM, 1990.

[54] Naveen Kr. Sharma, Ming Liu, Kishore Atreya, and Arvind Krishnamurthy. Approximating Fair Queueing on Reconfigurable Switches. NSDI, 2018.

[55] Bob Briscoe. Flow Rate Fairness: Dismantling a Religion. SIGCOMM, 2007.

[56] Frank Kelly. Charging and Rate Control for Elastic Traffic. European transactions on Telecommunications, 1997.

[57] Frank P Kelly, Aman K Maulloo, and David KH Tan. Rate Control for Communication Networks: Shadow Prices, Proportional Fairness and Stability. Journal of the Operational Research society, 1998.

[58] D. P. Palomar and Mung Chiang. A Tutorial on Decomposition Methods for Network Utility Maximization. IEEE J.Sel. A. Commun., 2006.

[59] Lucian Popa, Gautam Kumar, Mosharaf Chowdhury, Arvind Krishnamurthy, Sylvia Ratnasamy, and Ion Stoica. FairCloud: Sharing the network in cloud computing. SIGCOMM, 2012.

[60] Ion Stoica, Hui Zhang, and TS Eugene Ng. A Hierarchical Fair Service Curve Algorithm for Link-sharing, Real-time and Priority Services. SIGCOMM, 1997.

[61] K. Ramakrishnan, S. Floyd, and D. Black. The Addition of Explicit Congestion Notification (ECN) to IP. RFC 3168, 2001.

[62] Vishal Misra. Half the equation and half the definition. peerunreviewed.blogspot.com, 2015. URL http://peerunreviewed.blogspot.com/2015/12/what-is-definition-of-net-neutrality.html.

[63] Niloofar Bayat, Richard Ma, Vishal Misra, and Dan Rubenstein. Zero-Rating and Network Neutrality: Big Winners and Small Losers. In Proceedings of IFIP WG 7.3 Performance, 2020.

[64] T. Böttger, G. Antichi, E.L. Fernandes, R. Lallo, M. Bruyere, S. Uhlig, and I. Castro. The Elusive Internet Flattening: 10 Years of IXP Growth. RIPE 78, 2018.

[65] Brandon Schlinker, Hyojeong Kim, Timothy Cui, Ethan Katz-Bassett, Harsha V. Madhyastha, Italo Cunha, James Quinn, Saif Hasan, Petr Lapukhov, and Hongyi Zeng. Engineering Egress with Edge Fabric: Steering Oceans of Content to the World. SIGCOMM, 2017.

[66] Geoff Huston. The Death of Transit? APNIC.net, 2016. URL https://blog.apnic.net/2016/10/28/the-death-of-transit/.

[67] Personal and Confidential Communication from an ISP, 2019.

[68] Todd Arnold, Jia He, Weifan Jiang, Matt Calder, Italo Cunha, Vasileios Giotsas, and Ethan Katz-Bassett. Cloud Provider Connectivity in the Flat Internet. IMC, 2020.

[69] Yi-Ching Chiu, Brandon Schlinker, Abhishek Balaji Radhakrishnan, Ethan Katz-Bassett, and Ramesh Govindan. Are We One Hop Away from a Better Internet? IMC, 2015.