Principles for Internet Congestion Management

Author(s):
Brown, Lloyd; Gran Alcoz, Albert; Cangialosi, Frank; Narayan, Akshay; Alizadeh, Mohammad; Balakrishnan, Hari; Friedman, Eric Jon; Katz-Bassett, Ethan B.; Krishnamurthy, Arvind; Schapira, Michael; Shenker, Scott J.

Publication date:
2024-08-04

Permanent link:
https://doi.org/10.3929/ethz-b-000691726

Originally published in:
https://doi.org/10.1145/3651890.3672247

Principles for Internet Congestion Management

Lloyd Brown[1], Albert Gran Alcoz[2], Frank Cangialosi[3], Akshay Narayan[4], Mohammad Alizadeh[5], Hari Balakrishnan[5], Eric Friedman[1,9], Ethan Katz-Bassett[6], Arvind Krishnamurthy[7], Michael Schapira[8], Scott Shenker[1,9]

[1] UC Berkeley, [2] ETH Zürich, [3] BreezeML, [4] Brown University, [5] MIT, [6] Columbia University, [7] University of Washington, [8] Hebrew University of Jerusalem, [9] ICSI

Abstract

Given the technical flaws with—and the increasing non-observance of—the TCP-friendliness paradigm, we must rethink how the Internet should manage bandwidth allocation. We explore this question from first principles, but remain within the constraints of the Internet's current architecture and commercial arrangements. We propose a new framework, Recursive Congestion Shares (RCS), that provides bandwidth allocations independent of which congestion control algorithms flows use but consistent with the Internet's economics. We show that RCS achieves this goal using game-theoretic calculations and simulations as well as network emulation.

CCS Concepts

• **Networks** → **Network design principles**.

Keywords

Network Architecture

ACM Reference Format:

Lloyd Brown, Albert Gran Alcoz, Frank Cangialosi, Akshay Narayan, Mohammad Alizadeh, Hari Balakrishnan, Eric Friedman, Ethan Katz-Bassett, Arvind Krishnamurthy, Michael Schapira, Scott Shenker. 2024. Principles for Internet Congestion Management. In *ACM SIGCOMM 2024 Conference (ACM SIGCOMM '24), August 4–8, 2024, Sydney, NSW, Australia.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3651890.3672247

1 Introduction

In addition to being a technological marvel whose architecture has accommodated mind-boggling changes in size, speed, technologies, and uses, the Internet is also a massive experiment in decentralized resource sharing. Because computer communications are bursty, the Internet relies on packet-level statistical multiplexing to achieve reasonable efficiency. To deal with the inevitable overloads, the Internet relies on host-based congestion control algorithms (CCAs).

With this approach, the bandwidth a flow receives can depend heavily on the aggressiveness of its CCA. The Internet community quickly recognized that users would have an incentive to deploy ever more aggressive CCAs, thereby leading to overloads. To prevent this, the Internet community informally required all CCAs to be *TCP-friendly* (hereafter TCPF), as defined by [24]: "a flow is TCP-friendly if its arrival rate does not exceed the arrival of a conformant TCP connection in the same circumstances." [1] TCPF primarily applies to wide-area traffic on the public Internet, and we focus on that case in this paper. Specialized bandwidth allocation solutions are available in private deployments such as datacenters, enterprises, and private wide-area networks (WANs), in which there is a single administrative authority.

There are numerous practical and technical problems with TCPF. Prior work has shown that it is difficult to enforce [46] and that our understanding of the dynamics of CCAs breaks down at scale [38]. In addition, TCPF limits CCAs' ability to ramp up quickly and achieve full efficiency [49] and hinders the emergence of new delay-sensitive CCAs (as shown by Copa [5] and Nimbus [28]). It has also become clear that TCPF is no longer a strict requirement in deploying new CCAs, and that, in practice, non-TCPF CCAs will be deployed widely. For example, the TCP-unfriendly CCA BBR [17, 18, 47] has been widely adopted at Google, Amazon, Akamai, Dropbox, and Spotify for significant portions of their traffic.[2]

Given that TCPF is both deeply flawed and no longer adhered to by the major Internet actors, we should consider whether there are suitable alternatives to the TCPF paradigm. *This simple but central issue is the focus of this paper.* To that end, we explore from first principles what new conceptual framework might replace TCPF. However, while we reason from first principles, we do not start with a clean slate. We assume that, within our design/deployment timeframe, there will be no fundamental changes in the Internet architecture (*e.g.,* IP, BGP, and the best-effort service model) and its commercial arrangements (*e.g.,* how ISPs charge for service and peer with each other, and the widespread adherence to valley-free routing [26]). As such, we seek a conceptual foundation for how the Internet should share bandwidth that (i) can be implemented within the current architecture (though requiring additional protocols and mechanisms) and (ii) provides bandwidth allocations that are consistent with the current commercial arrangements between the parties involved.

This paper makes the following contributions:

- We articulate the goal of CCA independence (CCAI) (§2) as the foundational aim for sharing bandwidth.
- In contrast to the specific claims in prior work [12] and the general assumptions in literature (such as [20] and the

ACM SIGCOMM '24, August 4–8, 2024, Sydney, NSW, Australia

ACM ISBN 979-8-4007-0614-1/24/08
https://doi.org/10.1145/3651890.3672247

[1] At the time of [24], the term "TCP" prescribed a specific CCA: NewReno, as standardized in RFC2582. Also, even the staunchest of the early advocates recognized that TCPF was not tenable at high speeds, but the intent of proposals like High-Speed TCP [23] was to retain TCPF at lower speeds and create new standards for behavior at these higher speeds.

[2] While Google claims recent BBR versions are less unfair than the original [19], researchers dispute this claim [50] and BBRv3 remains TCP-unfriendly. However, our concern is not the degree of BBR's violation of TCPF but rather the lack of resistance to deploying CCAs that do not satisfy TCPF, and this applies to all BBR versions.

many related papers), we show that naively (*i.e.,* setting weights independent of the topology) applying weighted fair queueing or its hierarchical variants does not achieve our goals (§2 and §4.4).
- We derive three principles for achieving CCAI while being consistent with Internet economics (§3).
- Based on these principles, we propose a scheme that sets the weights in each node's hierarchical weighted fair sharing algorithm (§4) that depends on both the user access agreements (which is the traditional approach to setting weights) and the path packets have taken through the network (which is not). This topology-dependent setting of weights is novel and crucial to achieving CCAI.
- We perform simulations and emulations to show that in the overwhelming majority of cases, RCS achieves CCAI (§5, §6).

2 Replacing TCPF

Before addressing how we can replace TCPF, one might ask why we need *any* framework that guides how the Internet shares bandwidth. The reason is simple: without a coherent resource-sharing framework, ever-more aggressive CCAs could be deployed over time, and the resulting increase in overall congestion would be damaging to the Internet.

The key issue with TCPF and the lack of a framework is that the network plays a passive role, so aggressive CCAs receive more bandwidth on congested links. We propose to go to the other extreme by requiring the network to actively enable all reasonable CCAs to achieve the same bandwidth in the same static circumstances. We call this CCA independence (CCAI). CCAI removes the need for a single standard CCA, fostering widespread CCA diversity and innovation. For example, providing CCAI would make it practical to deploy delay-minimizing CCAs, a longstanding challenge. Since we assume no fundamental changes in the architecture or economics of the Internet in the near term, this paper addresses the challenge of achieving CCAI within a framework that is consistent with the current Internet's economic model.

Despite the vast literature on network-assisted congestion control, there is no such proposed framework. For instance, neither of the two leading contenders to replace TCPF – per-flow fairness (*i.e.,* as achieved by fair queueing [20, 36]) and network utility maximization (*i.e.,* as inspired by the work of Kelly [32, 33]) – are consistent with the commercial realities of the current Internet. This is because both of these approaches focus on individual "flows" (*i.e.,* seeking to achieve fairness between flows or to maximize the sum of flow utilities), but flows have no role in the Internet's commercial agreements (see [10]); flows don't have "rights" to bandwidth or utility, nor are they the units for which users are charged.

More recently, an in-progress IETF draft argues for provisioning low-latency, low-loss service (L4S) [11] via a new Internet architecture co-existing with the current best-effort one. L4S proposes to isolate delay-sensitive traffic from classic buffer-filling traffic in the network using traditional fair queueing methods. However, L4S does not provide CCAI; instead, it installs isolation between two classes of traffic and prescribes the CCA behavior (inspired by DCTCP [2]) for one of them. With RCS, we go further: with a comparable amount of deployment effort as L4S, we provide full CCAI, which enables delay-minimizing congestion control. Thus, with RCS, it is possible to satisfy L4S's goals while also providing principled bandwidth allocations.

Thus, to meet the challenge above, we need a new approach but not a new packet scheduling mechanism. That is, while hierarchical fair-queueing is the correct isolation *mechanism*, it requires careful topology-dependent *configuration* of queue weights and assignments to achieve CCAI. Our main contribution is to derive a set of principles that ground a weight and queue assignment framework consistent with the Internet's economic model; we call this "Recursive Congestion Shares" (RCS). RCS takes inspiration from the approach introduced in [12]; unfortunately, as shown in §6.2, this prior approach does not achieve CCAI.

However, before turning to those principles, we address three key questions.

What is a "reasonable" CCA? Our goal of CCAI requires that a flow's bandwidth should be independent of its or other flows' choices of CCA, as long as the CCAs are reasonable. This requires a definition of "reasonable": we say a CCA is reasonable if it effectively uses the available bandwidth in static settings while avoiding *persistent and significant* loss. We require this condition because it prevents flows from needlessly harming other flows and causing congestion collapse, as in the dead-packet phenomena discussed in [40]. Avoiding persistent and significant loss is compatible with all proposed CCAs we know of (unlike TCPF, which is far more constraining). Loss-based CCAs incur persistent low losses (but not significant loss), while BBR can incur occasional significant, but not persistent, losses (*e.g.,* if a flow encounters a quick reduction in bandwidth). Indeed, the only realistic scenario with persistent *and* significant losses is a DDoS attack, and operators already deploy significant resources to prevent and mitigate such traffic.

What is the role of CCAs post-CCAI? A key property of CCAI is that since a CCA's bandwidth rights are guaranteed, CCAs become free to optimize on other metrics such as loss, delay, and jitter. While today, designing a delay-minimizing CCA is considered impractical due to fairness concerns when competing against not only buffer-filling traffic but also other delay-minimizing flows [4], adopting RCS would enable the adoption of these and other optimized algorithms. Further, while RCS guarantees bandwidth *rights*, it does not dictate bandwidth *allocations*; this leaves the traditional problem of searching for available bandwidth while minimizing self-inflicted delay for CCA designers to solve while removing fairness concerns.

Is RCS deployable? While our goal with RCS is to provide a principled conceptual framework for Internet bandwidth allocation rather than an immediately deployable mechanism, we argue in §7.3 that our prototype design for RCS is not incompatible with wide-scale deployment. More importantly, both network operators and their customers will have incentives to adopt RCS's framework; whereas today, an operator cannot offer any guarantees about end-to-end bandwidth rights to customers who pay more for higher tiers of service, with RCS, such guarantees become feasible. Similarly, while a customer desiring better end-to-end performance might today deploy a private WAN with reserved capacity, with RCS, they could purchase the bandwidth rights their application needs at a lower cost. We discuss these economic incentives for adopting RCS further in §7.2.

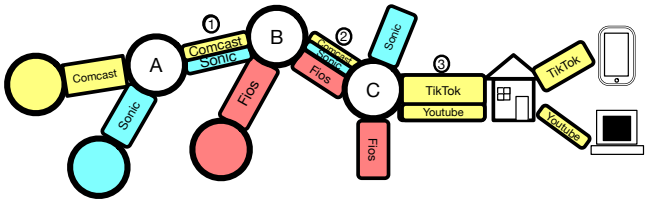


Figure 1: **Illustration of RCS principles: (1) Relative rights. (2) These rights should be applied recursively: the queue first considers Fios (red) and the aggregate formed by Sonic (blue) and Comcast (yellow). If it must drop a packet from the aggregate, it considers Sonic and Comcast's relative rights at the upstream domain A. (3) Endpoints manage traffic control at finer granularities. The endpoint prioritizes TikTok traffic over Youtube traffic within the Comcast aggregate.**

3 Principles for "Consistent" CCAI

Our goal is to achieve CCAI in a manner that is consistent with the Internet's commercial arrangements, but it is far from obvious what it means to be "consistent with the Internet's commercial arrangements." Here, we state three principles[3] that describe what this entails. We discuss how to achieve these principles in §4. Since these principles should guide congestion management both now and in the future, we neither tie them to the characteristics of today's traffic or network technologies nor make assumptions about what applications are dominant. We illustrate these principles in Figure 1.

Principle #1: Bandwidth allocations should be enforced only under network congestion and should be described in terms of relative rights.[4] This means that, when the network is congested, it uses packets' *relative rights* (explained next) to determine which ones to drop. In contrast, expressing bandwidth rights as guaranteed rates (*i.e.,* the absolute levels of end-to-end bandwidth a user can expect) would greatly reduce statistical multiplexing and thus be impractical. This condition does not disallow RCP [22] or XCP [31], but does disallow IntServ [39]; the former two only give ephemeral estimates (based on some notion of relative rights) while IntServ makes persistent guarantees (not based on relative rights).

Principle #2: These relative rights should be tied to current commercial arrangements, respecting their granularity, recursive nature, and flow of money. Users pay for access at the edge, so the prevailing commercial arrangements are at the granularity of these access agreements, not at the level of individual flows. In addition, these access agreements are recursive (*i.e.,* packets are delivered end-to-end because the sender's carrier pays the next-hop carrier, which pays the subsequent-hop carrier, etc.). Money similarly flows from a receiver's domain to that domain's provider, and so forth. These inter-domain arrangements are crucial to how networks carry traffic, and we thus argue that their recursive nature must play a role in how the network manages congestion.

Principle #3: While the network determines bandwidth allocations between two endpoints, the endpoints should determine the composition of traffic that flows between them. This basic requirement is clear, but the question is which endpoint should have control. The guidance from Principle #2 should determine whether the sender or receiver is responsible for controlling this traffic. Decisions should follow the flow of money, with senders making decisions about what traffic enters the network (using a mechanism such as Bundler [16]) and receivers making decisions about what traffic exits the network (using a mechanism such as Crab [45]). Intermediate cases should be determined by the money flow at the point of congestion. In particular, because of the dangers of "zero-rating" [5] [6, 7, 9], it is important that receivers, not senders, make decisions about what traffic they receive.

[3]Ours are not the only possible principles; we offer them as a starting point. A different set of principles might result in both different bandwidth allocations and different competition dynamics, and we leave the analysis of such frameworks to future work.
[4]Edge ISPs typically throttle a user's bandwidth on their access line to their contractual rate, regardless of congestion. Our focus here is on congestion internal to the Internet.

4 From Principles to Practice

We next turn these principles into an algorithm for calculating bandwidth allocations. For ease of exposition, we first consider allocations in static settings where users send at fixed rates over links with fixed capacities. The mechanisms we propose for RCS can handle realistic dynamic settings, but it is hard to reason about such settings in a principled manner. After developing this algorithm for bandwidth allocations, we then ask (in §5) whether the RCS allocations achieve CCAI. *Answering this question in the affirmative is the central contribution of this paper.* We further evaluate how our approach interacts with real CCAs (§6). We then discuss (i) some other practical concerns (§7), (ii) how RCS compares to related work (§8), and (iii) finish with some concluding remarks (§9).

4.1 Principle #1: Relative Rights

We explore the implications of relative rights in three scopes: a link (by which we mean a technology that has multiple ingresses and one egress), a switch (which has multiple ingresses and multiple egresses), and a domain (a network of switches). We use these three tractable cases as "building blocks" to reason more generally about managing congestion in the Internet. In the first two cases, the congestion point is at the egresses because (i) we assume that one switch's egress is another switch's ingress and that the paired egress and ingress have the same capacities (so any congestion would be handled by the previous egress, not the ingress) and (ii) we assume the switch has full internal bandwidth. In the third case, a domain, the location of congestion will depend on the specific scenario. In this section, we will assume that the relative rights are derived from the sender's access agreement (not the receiver's), but we will generalize this in §4.2.

We use the term "user" to refer to the entity entering into an access agreement (*i.e.,* an agreement that provides it some level of Internet service) at the network edge and "stream" to refer to an aggregate of traffic entering the network at the same ingress point and exiting the network at the same egress point. Thus, a stream is traffic being sent by one user and received by another. The term "CCA" refers to how a stream responds to congestion; such a response is, in fact, made up of several distinct flow-based congestion control algorithms and application behaviors (such as opening additional connections or throttling streaming data), but

[5]With zero-rating, content providers pay networks to deliver their traffic, but not traffic from other content providers, to users.

for convenience, we model it as a single CCA.[6] Hereafter, for cases where we have multiple ingresses and egresses, as in switches and domains, we use the term $aggregate(i, \alpha)$ to refer to traffic entering at ingress i and leaving at egress α.

Relative Rights at a single link. Consider a single link with several streams sending at rates r_i. We denote the resulting egress bandwidths (*i.e.,* the rate leaving the link from each stream) by a_i with $r_i \geq a_i$: strict inequality represents when the network drops packets from stream i, and we call such streams "constrained". The link is work-conserving, so $\sum_i a_i = MIN[\sum_i r_i, C]$ where C is the bandwidth of the link. In this context, we define *relative rights* in terms of weights w_i, and allocate bandwidth to constrained streams proportional to those weights. Specifically, for any two streams i, j with $a_i < r_i$ and $a_j < r_j$, the following holds: $\frac{a_i}{w_i} = \frac{a_j}{w_j} \geq \frac{a_k}{w_k}$ for all other streams k. The equality between i and j requires that two constrained streams receive bandwidth proportional to their weights. The inequality between j and k requires that no unconstrained stream is getting more than if it were constrained. This definition implies that $a_i = MIN[r_i, w_i\lambda]$ where $\lambda \geq 0$ is the smallest value that allows $\sum_i a_i = MIN[\sum_i r_i, C]$.

This results in what we call "pipe-like" behavior. As a stream increases its bandwidth demand r_i, at first, its demand is entirely satisfied, and then it is capped at some maximal bandwidth. This sharp "knee" in the curve makes it easy for a CCA to find the maximal allowed bandwidth and doesn't reward streams for creating persistent drops (*i.e.,* they get no additional bandwidth by sending at a rate past the knee).

As a comparison, if a link does not actively manage congestion and just uses FIFO packet scheduling, then the bandwidth allocations are given by: $a_i = \min[r_i, r_i \frac{C}{\sum_j r_j}]$, and the average packet delay (which is the same for all streams) is some function of $\sum_j r_j$ that increases sharply as the quantity reaches the link capacity. If we fix all other $r_j > 0$, a_i is strictly monotonic in r_i and stream i's packets can experience significant losses and delays even if r_i is very small (*e.g.,* if the remaining load $\sum_{j \neq i} r_j$ is much larger than the link capacity).

Relative Rights at a single switch. The minimal generalization of the single-link approach is to have each egress apply the single-link definition with weights w_i for each ingress i applied at all egresses α. This is the approach we use in RCS. We could also consider the case where the weights are static but depend on each egress: *i.e.,* the $aggregate(i, \alpha)$ from ingress i to egress α has a weight w_i^α. For simplicity, we do not embrace this generalization in our treatment here, but our results apply to this case as well.

One might argue that the weights should depend on the current traffic matrix, with the total weight assigned at ingress split across the weights applied at egress proportional to the current traffic split. For instance, assume that all ingresses and egresses have capacity $C = 1$, $\sum_\alpha w_i^\alpha = 1$ for all i, and weights w_i^α for a given i are proportional to the relative flow rate (*i.e.,* if two-thirds of an ingress's traffic goes to one egress, then that aggregate gets two-thirds of the ingress's weight). Consider the case where two ingresses i, j send traffic to two egresses α, β. Recall that reasonable CCAs maximize bandwidth subject to the constraint that they do not experience significant and persistent loss; in the context of this model, this means they select the maximal value for r_i such that $r_i = a_i$. Assume i sends all of its traffic, of rate r_i, to α while j splits its traffic, of total rate r_j, between α and β in proportions x and $(1 - x)$. Then the weight of $aggregate(i, \alpha)$ is 1, of $aggregate(j, \alpha)$ is x, and of $aggregate(j, \beta)$ is $1 - x$. The only allocation choices where ingress j does not incur persistent losses at egress α are (i) $x = 1$ and $r_j = \frac{1}{2}$ (with $r_i = \frac{1}{2}$) and (ii) $x = 0$ and $r_j = 1$ (with $r_i = 1$). This is because, as soon as j dilutes its weight by sending traffic to both egresses, some of its packets are dropped. Thus, splitting weights proportional to traffic can lead to pathological allocations, so we eliminate it as a viable way of setting weights.

Relative Rights at a domain. We do not assume congestion only happens at the edges of a domain, but we do assume that the relative rights are determined by access agreements between users and their domains as well as those between domains. If there is no internal congestion, then a domain behaves analogously to a switch. However, if the domain does suffer internal congestion, it should enforce the relative rights (as defined by the weights w_i assigned upon egress) on all internal links or switches where congestion occurs. We believe most domains are, and will continue to be, managed to avoid internal congestion except at particular hotspots – such as cable modem termination systems (CMTSs) and transoceanic links – so this enforcement need not be widely deployed inside a domain. For convenience, in what follows, we assume there is no internal congestion.

Following [12], we call these weights w_i congestion shares, and they are determined as part of a user's agreement with their domain. The value of w_i is not directly related to the access bandwidth, but presumably, access agreements for higher speeds will typically have higher congestion shares.

4.2 Principle #2: Recursion and Following The Money

We take the approach discussed above for allocating bandwidth in a single domain as a basic building block and discuss how to extend that approach across multiple domains using the second principle. This extension across domains and our demonstration that it is sufficient to achieve CCAI in the vast majority of cases is the main contribution of our work.

What does "Recursion" mean? Still focusing on the case where weights are assigned on ingress to a domain when a user's packets enter their ingress network (call it domain A), their relative rights when leaving A (via one of A's egress links) should be determined by the user's access agreement with domain A. When those packets travel from domain A to domain B over some link L,[7] to first order the relative rights of those packets when leaving domain B should be determined by the access agreement between A and B on that link L. While B's decision about how many of A's packets to drop is driven by the access agreement between A and B, when domain B is deciding *which* of A's packets to drop, the decision should be driven by the relative rights derived from A's various access agreements with the users associated with that traffic (as in Figure 1). This is

[6] We later return to the question of how to enforce CCAI within a stream in §7.1. For now, we merely note that this (i) is a matter internal to a single organization and, as such, does not have to be consistent with any economic agreements and is merely a matter of internal policy, and (ii) requires mechanisms such as [16, 45] to implement that policy when the congestion occurs elsewhere in the network.

[7] Our approach also applies to peering via IXPs; we briefly explain in §6.

what we mean by recursion of access agreements: (i) we recursively assign relative rights as packets travel through the network based on the agreements with the domain in which they are currently (since we are assuming no internal congestion in this example, these rights are only relevant at the egress of a domain), and (ii) we apply these rights in a hierarchical fashion, first applying their current rights to decide the total bandwidth, and then turning to their previous rights. Thus, RCS associates each stream with a hierarchy of access rights. The way we enforce relative rights is to schedule packets (see §6), so if a stream exceeds its bandwidth allocation, it will eventually overflow its queue, and its packets will be dropped.

What does "Following the Money" mean? In the Internet, money typically flows from end users to domains and onward to other domains that provide broader routing reach. This flow of money eventually ends with the Tier 1 providers which freely peer with each other. The typical routing path goes up this hierarchy of domains (going from customer to provider) and then down (going from provider to customer). Thus, for a flow between two end users, there will exist a point at which packets switch from following the flow of money (up the hierarchy) to flowing against the flow of money (down the hierarchy). We use this observation to define two rules that control which relative weights guide the sharing of bandwidth. We will first consider one domain's relationship with end users, then explain how the same two rules also apply to traffic between domains.

The first rule is that, consistent with Principle #3, at the egress from a domain to its user (*i.e.,* an aggregate's destination), the domain should derive the user's relative rights from its commercial agreement with that user. As an example, consider two media streams coming from two different content providers with a total bandwidth that is larger than the domain's egress to the user; the user should have the power to assign weights expressing the relative rights of those streams.

Analogously, the second rule is that when an end user's traffic enters a domain (*i.e.,* an aggregate's source), its commercial agreement with the domain should determine its relative rights at the domain's egress. That is, when the traffic from two users exceeds the capacity of one of the domain's egress links, the relative weights are determined by the weights of the sending users. Note that this rule conflicts with the first rule in the case that an aggregate both enters a domain from an end user and exits the same domain to another end user. In this case, the first rule applies; *i.e.,* the receiver determines the aggregate's relative rights. This receiver-preference tiebreak is important because it prevents zero-rating, where a company can pay provider domains to deliver only their traffic to users at the exclusion of other traffic.

What allocations are produced when applied recursively? Given these two specific rules for packets entering and exiting the Internet, which respect the flow of money, we now seek to apply this approach recursively. While today's interdomain agreements are more complicated than the Gao-Rexford model [26], we believe the following two statements hold for the vast majority of the cases: (i) for a specific logical link between domains A and B, either A pays B, or B pays A, or neither pays, and (ii) the payment structure along Internet paths are "valley-free" in the sense Gao and Rexford describe. Thus, using the term customer/peer/provider to refer to the flow of money on a given link, we know that, given valley-free routes, two facts hold. First, if the egress α is to a customer, then all subsequent hops are to customers (and they determine all the hierarchical weights assigned to aggregates at that egress). Second, if the egress α is to a peer or provider, then all previous hops are from customers (and they determine all the hierarchical weights assigned to aggregates at that egress). As a result, all aggregates at an egress α have their weights determined in the same direction (either previous hops or future hops). The resulting bandwidth allocations result from the recursive application of relative weights.

Define $w_{i,\alpha}$ as the weight assigned to $aggregate(i, \alpha)$. Taking the case where the weights at an egress α were assigned recursively by previous ingresses, the calculation goes as follows: first calculate the allocation to each $aggregates(i, \alpha)$ for all i using the weights assigned by ingresses i. That produces a set of allocations a_i. Then, for each $aggregate(i, \alpha)$, calculate the allocations for all of the aggregates that entered through ingress i (with the weights assigned by their previous hop ingresses), treating the total bandwidth as a_i. This process recurses all the way down the hierarchy of domains until it reaches end users.

We call this allocation Hierarchical Weighted Fair Sharing (HWFS), which is identical to the static allocations achieved by the various hierarchical weighted fair queueing algorithms in the literature [8, 44] with the same given set of weights. This work's novelty is in the *configuration* of the weights; we propose computing these weights from the rest of the topology and from access agreements.

4.3 Principle #3: Endpoint Control

This principle states that while the network determines bandwidth allocations to each stream, endpoints (both ingress and egress) should determine the composition of that stream. For example, while relative rights should determine the aggregate bandwidth allocation between two university networks based on their access agreements, those networks may want to prioritize bulk transfers of research data over video streaming traffic. RCS makes no statements about this prioritization other than to note that endpoint domains should control it. Of course, an endpoint could decide to use a default FIFO policy, in which case the most aggressive CCAs within the endpoint's streams would take more of its bandwidth allocation. However, RCS would prevent those CCAs from affecting the bandwidth available to *other* aggregates.

To understand how to achieve this, it is useful to distinguish between three cases where a stream might encounter congestion. The first is on a user's ingress into the network, where the user can use its own internal mechanisms to control the composition of the stream. The second is somewhere internal to the network, such as on an egress link between two domains. Recent work on Bundler [16] and Crab [45] provides a mechanism whereby users can remotely control the internal composition of the stream (we provide more details about how Bundler and Crab work to achieve this in Appendix C). The third case is at the endpoint domain's egress to the destination user; this is a special case of the prior one, and the same mechanisms can be applied. Note that in keeping with the previous principle, the sending user determines the composition if congestion occurs when sender weights apply, and vice versa for the receiving user.

4.4 How Does Our Work Relate to [12]?

This paper was inspired by [12], which introduced the basic idea of recursive congestion shares. However, the detailed approach in [12] has two major limitations. First, [12] only applies one level of weights: those being assigned by the most recent ingress the traffic has passed through. For user traffic that is going directly from one domain to another (precisely, the case where traffic goes from user X, to domain A, to domain B), this is sufficient.[8] However, this does not cover the case where traffic goes from users X and Y, to domain A, to domain B, to user Z. If the congestion is at the egress of domain B, there is no way for the transit provider to distinguish between the streams belonging to the two customers X and Y, and the customer with the more aggressive CCA will get more bandwidth. Thus, the version in [12] does not achieve CCAI even for some relatively short paths.

The second limitation is that the design in [12] ignores the role of receivers, only considering weights assigned by ingress points. This raises equity issues (*i.e.,* should content providers determine the priorities on my network access line?), and also means that the allocations do not follow the flow of money on the downward part of the path. These two deficiencies mean that the more complicated but more functional proposal presented here is required.

5 Does RCS Achieve CCAI?

The principles above ensure that RCS is consistent with the Internet's economic model. We now ask whether RCS achieves our goal of CCAI. In this section, we address that question from a game-theoretic perspective where users are trying to individually optimize their throughput, subject to the reasonability constraint that they do not incur persistent losses. We analyze this game in two ways: (1) a mixed-integer linear program (MILP) formulation that determines whether multiple game-theoretic equilibria exist and (2) simulations of game theory dynamics to determine if reasonable CCAs would converge to those equilibria.

5.1 Just Enough Game Theory

In our model, the individual CCAs that control streams are the game's "players," which we assume act rationally (*i.e.,* maximizing utility). The "game" is defined by the network topology and bandwidth allocation rules (*i.e.,* RCS) that determines, given a set of input rates r_i, what are the resulting throughput rates a_i (using the previous notation). A player's strategy is their sending rate, and their payoff is either their sending rate (if their output rate is the same as their sending rate) or $-\infty$ if their output rate is less than their sending rate: this condition merely ensures that no equilibrium results in persistent loss, and is not intended to reflect a realistic payoff. We thus ask: what kinds of equilibria do these games converge to? To answer this question, we use two concepts from game theory.

The first concept is the familiar *Nash equilibrium* [25]. A Nash equilibrium occurs when all players (*i.e.,* CCAs) are playing their optimal strategy (*i.e.,* sending rate), assuming all the other players have already chosen their strategies and they remain fixed. That is, for each i, r_i is at the maximal value such that $a_i = r_i$, assuming all players other than i keep their sending rate fixed. This is an equilibrium where no player can gain by unilaterally deviating from the equilibrium.

The second concept is the less familiar *Stackelberg equilibrium* [25]. In this model, a "leader" (*i.e.,* an individual CCA) commits to some strategy (*i.e.,* a sending rate) first. Other "follower" CCAs in the game observe this leader's action and react to it, reaching a Nash equilibrium in the resulting sub-game with the leader's strategy remaining fixed. A Stackelberg equilibrium (with a single leader i) occurs when the leader i has chosen its optimal strategy (has chosen the maximal value of r_i such that $a_i = r_i$), assuming that in each case, the set of other players will reach a Nash equilibrium in their subgame in response to the leader's strategy. Informally, we think of the leader as testing each of their strategies, observing where the others converge to, and picking the strategy that leads to the best outcome.

For a given game, if there is a single Nash equilibrium and it is also the only Stackelberg equilibrium, then the game has exactly one stable equilibrium, regardless of how "aggressive" each player is. However, if there is a Stackelberg equilibrium that differs from the Nash, or if there are multiple Nash equilibria[9] (which implies there must be multiple Stackelberg equilibria), then aggressive players (*i.e.,* the Stackelberg leaders) can try to manipulate the game to achieve the outcome that maximizes their throughput (we show example topologies where this is true in Appendix B). We show in §5.3 that such manipulation rarely results in a better equilibria for the leader. Moreover, such manipulation would be infeasible to achieve since the Stackelberg leader would either have to be omniscient (so it could calculate its optimal strategy) or sample the response to its behaviors over long time periods (so that the other players would have converged to an equilibrium) and then search for its optimum.

5.2 Does Greed Pay?

To understand whether multiple Nash equilibria or non-Nash Stackelberg equilibria are common, we consider two models. The first is based on a random topology, which we designed to increase the possibility of pathological cases: each stream takes a random path and uses random weights at each ingress along the path. We do not have aggregation in this model (each stream is treated separately at each router), so each router only applies its own weights to each stream. There is no need to refer to previous hops because flows never aggregate. We start with a fully-connected 10-node topology and then generate 40 4-hop streams by picking a random sequence of nodes.

The second model uses topologies sub-sampled from the CAIDA AS relationships dataset [15]. We pick a random AS in the dataset to start an initial stream. Then, with probability 0.7 we set this stream's next-hop (or previous-hop – we grow the stream in both directions) to one of that AS's neighbors while maintaining a Gao-Rexford compliant path, or else terminate the stream at a neighbor. When we add a new AS to a stream's path, with probability 0.5 we also generate a new stream passing through at that AS. We grow this stream using the same rules. We stop growing the topology

[8] [12] incorrectly claims that their version of RCS is sufficient for traveling through three domains, but this ignores traffic that is either generated and consumed by end users (as opposed to being generated or consumed internal to a domain, as it would be by Facebook or Google)

[9] §9 shows an example topology with multiple Nash equilibria.

once we have generated 40 streams. Since the CAIDA AS relationships dataset does not contain capacity or weight information, we generate these randomly per link.

We use a commercial MILP solver [37] to calculate allocations under the two models. We disallow streams from incurring persistent losses.

Random model. The MILP starts by defining allocation variables where a_s denotes the allocation that stream s receives in an equilibrium. From a given scenario, we define a set of links L with capacities c_l, S as the full set of streams, S_l as the set of streams that pass through each link l, and a set of weights such that $w_{l,s}$ denotes the weight of stream s at link l. Then we create bottleneck indicator variables $B_{l,s}$ such that $B_{l,s} == 1$ if and only if stream s is bottlenecked at link l. Lastly, we create an indicator variable for links such that $C_l == 1$ if l is at capacity. We then define the following constraints:

$$\forall_{l \in L} \forall_{s \in S \triangle S_l} B_{l,s} == 0 \tag{1}$$

$$\forall_{l \in L} \forall_{s \in S_l} \forall_{s' \in S_l} \frac{a_s}{w_{l,s}} - \frac{a_{s'}}{w_{l,s'}} \geq -M * (1 - B_{l,s}) \tag{2}$$

$$\sum_{\forall l \in L} B_{l,s} == 1 \tag{3}$$

$$\sum_{s \in S_l} a_s \leq c_l \tag{4}$$

$$\sum_{\forall l \in L} B_{l,s} \leq 1 \tag{5}$$

Constraint (1) ensures that no stream is bottlenecked at a link not on its path. Constraint (2) ensures that each stream gets its weighted fair share at each congested link (M is some large constant). Constraint (3) ensures each stream s is bottlenecked at exactly one link. Finally, constraint (4) ensures no link l is oversubscribed. It is important to note that we use this MILP formulation to model bandwidth allocations *under RCS*, rather than to model the status quo. Specifically, due to the specification of RCS's "pipe-like" behavior (§4.1), transient interactions don't determine bandwidth allocations in RCS; we thus do not model them in this analysis. We consider transient interactions when evaluating partial deployment scenarios in our emulation experiments (§6.2).

To ensure that we characterize the full range of Nash equilibria for a given stream, we take turns optimizing a_s for each stream s to determine the maximum and minimum allocation for that stream under these constraints. Thus, a Nash solution to a topology yields two equilibria (corresponding to the maximum and the minimum) for every stream, though in most cases, they turn out to be the same. To generate a Stackelberg solution with stream s as a leader, we optimize a_s and allow s to be bottlenecked at no links or a single link by replacing constraint (3) with constraint (5).

We show the results on the first row of Table 1, where very few topologies exhibit problematic equilibria. When looking at individual streams, the results are even stronger. Out of a total of 748, 920 streams in all scenarios, only 68 (0.009%) streams benefited from an aggressive Stackelberg leader strategy. The average percent gain for any stream attempting to improve itself by adopting a Stackelberg strategy was only 0.011%. Thus, in this model, greed does not pay (very much).

Topology	Total	Multi. Nash	Stackelberg
Random	18, 723	338 (1.81%)	598 (3.19%)
CAIDA-sampled	2, 897	0	0

Table 1: **The types of equilibria under different topologies. We derive CAIDA-sampled topologies from published data on AS relationships [15] and derive Random topologies from our heuristics for generating AS graphs (§5.2). The results show that with RCS, both multiple Nash and non-Nash Stackelberg equilibria are rare for random topologies, and both are even more rare for CAIDA-sampled topologies.**

CAIDA-sampled model. For the CAIDA-sampled model, in our MILP calculations, domains assign weights to aggregates on ingress. Egress links between two domains mirror these agreements. Thus, in this MILP calculation, we only model weights assuming all traffic follows the flow of money (but consider the more general case in §5.3 and §6.2). To implement the CAIDA model, we add the following constraint to what we described above:

$$\forall_{l \in L} \forall_{s \in S_l} \forall_{d \in 0..D_l} \forall_{agg' \in aggs_{l,d}} : agg_{l,s} \neq agg'$$
$$\frac{\sum_{s* \in U_{agg_{l,s}}} a_s*}{w_{agg_{l,s},l}} - \frac{\sum_{s* \in U_{agg'}} a_s*}{w_{agg',l}} \geq -M * (1 - B_{l,s}) \tag{6}$$

This additional constraint enforces recursively-weighted allocations. At each depth d of the hierarchy (with $d = 0$ being the root) for all streams s in S_l (borrowed from the flat model), we define $agg_{l,s}$ as the aggregate containing stream s at link l, and $w_{agg,l}$ to be the weight assigned to aggregate agg. Lastly, we define D_l to be the maximum depth of the hierarchy at link l, $aggs_{l,d}$ to be the set of aggregates at depth d of the hierarchy for link l, and U_i to be the set of streams comprising aggregate i.

We show the results on the second row of Table 1. We found no CAIDA-sampled topologies where multiple Nash or non-Nash Stackelberg equilibria existed, which means adopting Stackelberg strategies would not benefit any senders.

5.3 Will Reasonable Flows Benefit?

The MILP formulations show that in the vast majority of cases, there is a single Nash equilibrium. The question now is, will reasonable CCAs–which myopically adjust their behavior–reach this single Nash equilibrium?

To model myopic game theory dynamics, we implement a "best-reply" agent, which iterates through several sending rates between a min and max and computes their utility at each one. The two sending rates surrounding the best one achieved are used as the min and max for the subsequent iteration: note that the best sending rate is the highest rate at which it does not experience drops. The sender terminates when the difference between the max and min rate is lower than a small ϵ.

We used our CAIDA topology generator (described above) and sampled 1,030 8-stream topologies in which we considered sender-only weights (606) and sender and receiver weights (424). We assigned CCAs to hosts in the topology (such that each CCA is represented at least once) and ran multiple iterations rotating the CCA assignments. In all of these topologies, the agents converged to a single equilibrium.

We also investigated whether this held with more realistic dynamics. We implemented simplified versions of several conventional CCAs (CBR, AIMD, BBR, Decongestion Control, Latency-sensitive) and said a flow was in equilibrium if it reached a repetitive cycle in terms of its sending rate (that cycle differs depending on the CCA). Again, in 1,052 topologies, the agents converged to a single equilibrium.

5.4 Summary

These results suggest that RCS is indeed sufficient to achieve CCAI in most settings and that there is little payoff in trying to exploit RCS (*i.e.,* the average payoff is extremely low, even if the CCAs were omniscient). What leads to this degree of effectiveness? The use of hierarchical weighted fair queueing – whether applied to individual flows or to aggregates – is necessary to achieve CCAI at a single node, but it is clearly not sufficient, as we demonstrated in the simple example in §4.4. The key is that the weights in the hierarchical tree in a given egress are based on the set of peering agreements upstream or downstream of that egress.

6 Implementation and Evaluation

We now turn to packet emulations to determine whether an implementation of RCS's bandwidth allocations provides CCAI for real-world CCA implementations. We first describe an implementation of a scheduling algorithm that can implement HWFS, Hierarchical Deficit Weighted Round Robin (HDWRR). We then show results from packet emulations (using Mininet [29]) that use three CCAs with varying levels of aggressiveness: Reno, Cubic, and BBR. We compare results across (i) the idealized best-reply allocations described above in §5.3, (ii) HDWRR, (iii) per-flow fairness using the deficit round-robin (DRR) algorithm, and (iv) FIFO scheduling.

6.1 Hierarchical Deficit Weighted Round Robin

To evaluate RCS's allocations on real CCA implementations, we must first implement a mechanism that can achieve those allocations. For this, we use a scheduling algorithm, HDWRR, that is based on the Deficit Round Robin (DRR) [41] implementation of a fair queueing scheduler. We show pseudocode for our implementation of HDWRR in Listing 1 (in Appendix A). Unlike DRR, which is constant-time, HDWRR's complexity scales with the depth of the weight tree. Additionally, while DRR can always assign a given queue its full quanta, HDWRR must track the case where a non-leaf node in the tree does not have enough remaining deficit to accommodate a full quanta for its next child. In this case, our implementation assigns the child node any remaining deficit but tracks the remaining scheduling deficit for the next round.

6.2 Mininet Emulations

To evaluate our HDWRR implementation as well as its interactions with real CCAs, we implement a prototype HDWRR in ~150 lines of Rust as a user-space TUN/TAP device. The full implementation, including alternate DRR and FIFO schedulers, is ~1500 lines of Rust.

We consider four experiment configurations: (1) *FIFO*, which represents the status quo, (2) *DRR*, which allocates per-stream, not per-aggregate, as in fair queueing; (3) *One-Hop*, which uses HDWRR but limits the weight tree to depth 1 (as in [12]); and (4) *HDWRR*, which is our implementation described above.

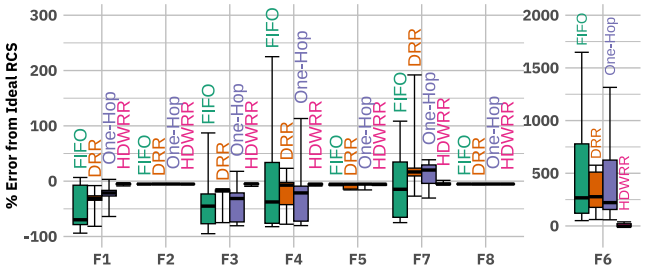
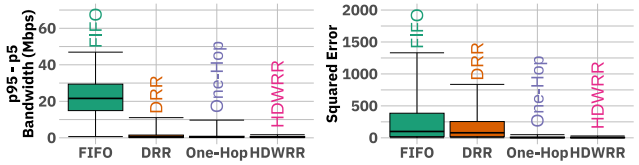


Figure 2: **Achieved allocations on an emulated topology with real CCAs. A perfect scheme has 0% error for all flows. All boxplots in this paper show percentiles (p5, p25, p50, p75, p95). Note the different y-axis for F6.**



a: **CCAI** b: **Allocations**

Figure 3: **(a) shows difference between bandwidth received in 5th and 95th percentile for each scheme. Smaller difference means CCA choice impacts received bandwidth less. (b) shows squared error between allocations flows receive under the scheme and expected allocations under RCS.**

We evaluate these four configurations in two ways. First, do they provide CCAI, *i.e.,* do all streams achieve consistent bandwidth regardless of what CCA they (or other flows) use? Second, are the bandwidth allocations consistent with their economic relationships? We represent whether allocations are consistent with economic relationships by normalizing the streams' achieved bandwidth to the shares our simulator (§5.3) calculates. To evaluate these questions, we use the same CAIDA-sampling topology generator that we described above in §5.2. We generate 8 and 15-stream[10] topologies, and for each topology, we further generate 10 random CCA assignments, where we randomly assign Reno, Cubic, or BBR as each stream's CCA. We show results both for all streams in a single topology (Figure 2), as well as a distribution across the 10 CAIDA-sampled topologies we generate. While we summarize statistics across topologies, we note that CCAI and bandwidth misallocation matter most in the worst case, since if any part of a topology does not support CCAI or economically consistent bandwidth allocations then CCAs must be pessimistically designed for this worst case.

FIFO. As expected, using FIFO scheduling provides neither CCAI nor RCS-compliant bandwidth allocations. Consider flows F1, F3, F4, F7, and F6 in Figure 2: these flows receive up to 100% below the RCS allocation (*i.e.,* starvation), and F6 receives up to 15x over-allocation. Similarly, across the topologies (Figure 3), FIFO provides neither consistent bandwidth allocations, with a 28 Mbps spread between p5 and p95 bandwidth (left) nor economically consistent allocations with a large squared-error relative to the RCS allocation.

DRR and One-Hop. Traditional fair queueing, as well as one-hop (which adds a single layer of hierarchy), can provide CCAI,

[10]Recall that a stream can have multiple component TCP flows, and endpoint domains control those flows' relative allocations.

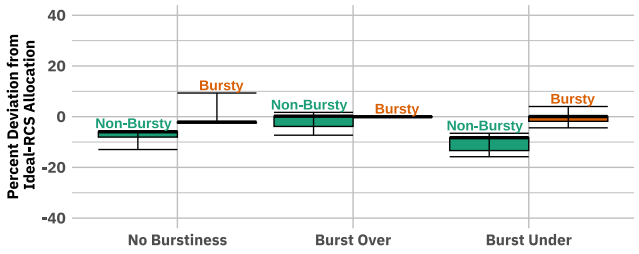


Figure 4: **In RCS, bursty aggregates receive similar bandwidth allocations compared to non-bursty ones.**

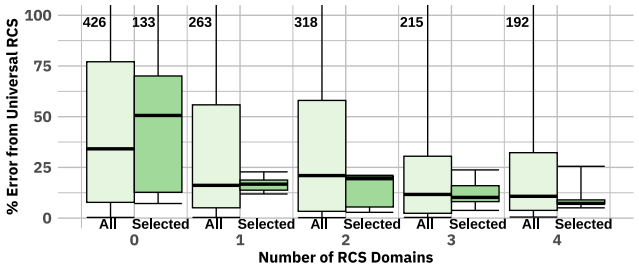


Figure 5: **The x-axis shows the number of egresses running RCS along a given stream's path. We display the error rate in achieved flow allocations compared to those of a network-wide RCS deployment. The numbers at the top are outlier p95 errors.**

but both result in non-RCS-compliant bandwidth allocations. Indeed, Figure 3 shows that DRR and One-Hop achieve a much lower bandwidth spread than FIFO; their bandwidth allocations remained consistent across choices of CCAs. Unfortunately, DRR and One-Hop's allocations can be far from RCS's; in Figure 2, flows F1, F3, F4, F6, and F7 under- or over-allocate significantly.

HDWRR. HDWRR is the only evaluated scheduling algorithm that achieves both CCAI and allocations consistent with RCS. We can see in Figure 2 that all flows achieve allocations nearly identical to the ideal RCS allocations, and similarly, across topologies, both the bandwidth spread and squared error are low.

6.3 Bursty Traffic

Thus far, we have focused on evaluating scenarios using traditional CCA implementations. In this subsection, we consider whether aggregates can influence the allocation they receive by varying their traffic's burstiness. We consider both "over" and "under" forms of burstiness: "over" bursty aggregates send traffic at more than the expected RCS allocation for short periods of time, and "under" bursty aggregates send traffic at less than the expected allocation. We run an experiment in which "over-" and "under-" bursty aggregates compete with a control aggregate that uses TCP cubic.

We ran RCS on a simple topology where two aggregates (with the other aggregate having half the weight of the TCP aggregate) feed into a single shared bottleneck. We fix one aggregate to use Cubic as its CCA and consider three configurations of the other aggregate: "Regular," *i.e.,* another aggregate using Cubic, and Burst Over and Burst Under as described above. We assign the fixed aggregate a weight twice that of the bursty aggregate. We generate bursty traffic by generating short flows with sizes drawn by analyzing packet traces from an Internet core router [15], with a target average sending rate. We configure this traffic generator with an average sending rate 50% above the Ideal-RCS allocation for Burst Over, and 50% below the Ideal-RCS allocation for Burst Under. In both cases, there are both periods of time for which the instantaneous network load is above the Ideal-RCS allocation as well as below it. For each burstiness configuration, we measure the percentage deviation of each aggregate's allocation compared to the Ideal-RCS allocation across 20 experiment iterations.

Of course, an aggregate can at any time achieve less than its Ideal-RCS allocation by sending less data, and our work-conserving scheduling implementation will allocate this bandwidth to other traffic. Rather, in Figure 4 we evaluate two factors: first, whether the bursty aggregate can gain more bandwidth than its RCS allocation by bursting traffic, and second, whether the bursty aggregate can force other aggregates to achieve less than their RCS allocations by bursting traffic. We find that the answer to both these questions is no. In the Burst Over case, the percent deviation from Ideal-RCS remains below 2%, which is also lower than the p95 deviation we observe in the case with no burstiness. Similarly, burstiness does not significantly affect the allocation of other aggregates' bandwidth; the p5 deviation we observed for the non-bursty aggregate in the Burst Under case was 16% below the Ideal-RCS allocation, which is comparable to the p5 deviation in the case without burstiness, 12% below Ideal-RCS.

6.4 Incremental Adoption

Can RCS provide its benefits without requiring a universal adoption? In the most general case, the answer is no: arbitrary competing traffic could out-compete a single domain's RCS traffic elsewhere in the network. However, we identify one common case where incrementally adopting RCS is both useful and effective: bilateral commercial agreements along a single path. For example, if a domain's provider has upstream congestion, the two domains could mutually use RCS to inform each other about their relative rights. In this case, although both domains' traffic would still be susceptible to interference from other non-participating domains' traffic, they would respect the relative rights of each others' traffic.

In Figure 5, we show an evaluation of this type of incremental deployment. We generate a 12-stream topology and first evaluate the streams' allocations with zero participating RCS domains (the left-most set of boxplots). We then choose a stream with an inter-domain path-length of at least 4 at random; note that enforcing a lower-bound on the path-length represents the worst case for RCS's incremental deployment prospects, since with longer paths there is more opportunity for contention. Along the chosen stream's path, we incrementally switch each domain's egress along that stream's path to observe RCS's bandwidth allocation rules and measure each resulting allocation. In each case, we measure the deviation in allocation from a universal deployment of RCS across five 30-second iterations for both (a) the stream under test as well as (b) all flows in the topology; in both cases, a lower deviation indicates that RCS is resilient to incremental deployment scenarios. We define and report the allocation percent error for a given stream and topology (parameterized by the depth) as the absolute percent difference between the allocation the stream receives in the given topology and the allocation the stream receives in a topology where every egress is running RCS. In the topology we sampled for this experiment, the stream's path length is 4, but we note that using 4 RCS domains

is not a universal deployment: there are 17 other non-RCS domains whose traffic can interfere with the stream under test.

We find that enforcing RCS at even one domain significantly reduces the deviation in allocation for the stream under test from a median deviation of 51% to 17%. When all four domains enforce RCS, the deviation further drops to 7%, but other streams benefit as well, with a median deviation of 11% compared to the original 34%. When we consider all streams, and not only the stream under test, the deviation in allocations from a universal RCS deployment is much higher—the p95 deviation across the 5 iterations is 192% even with four domains enforcing RCS allocations. Of course, many of the streams contributing to the deviation don't encounter RCS-enforcing domains in their path, so we do not expect these streams to receive RCS-compliant allocations. However, for the stream under test, our experiment shows that incremental adoption does converge to RCS's allocation.

7 Practical Concerns

7.1 Additional Mechanisms

RCS Signalling. To implement HWFS, each egress needs to know the hierarchy of aggregates to which a packet is assigned and the weights associated with those aggregates. Providing this state requires (i) protocols carrying information between ingresses and egresses within the same domain, and (ii) passing between one domain's egress and the connected domain's ingress. There are many possible implementations for this first task, but perhaps the most straightforward is standing signalling connections between each ingress and each egress in a domain, and they periodically exchange (in both directions) information about prefixes and their associated weight hierarchies. For the second, all that is needed is to forward a summary of the information received from the intradomain signalling to the attached domain.

RCS Endpoint control. Recall RCS's third principle: while the network should determine a stream's total bandwidth allocation, that stream's endpoints should determine which individual flows use that bandwidth. There is an implementation challenge in supporting more complex flow allocation policies: the stream's bottleneck (*i.e.,* the egress link where it encounters congestion) will likely not be at a link within its control, which is where such control must be exercised.

To address this challenge, we adopt the approaches described in Bundler [16] and Crab [45] that shift congestion (and therefore the buildup of packet queues) from an egress in the network to a link at the appropriate endpoint (fully described in Appendix C).

Complications. As mentioned earlier, for congestion internal to a domain, congested links can use some form of HWFS, using the signaling mechanisms proposed above. We think domains will continue to be managed such that internal congestion is rare except for specific concentration points that domains can arrange to include in their internal signalling.

Our description so far assumes direct peering relationships over dedicated bilateral links. Many domains peer via IXPs over a common substrate, where the access line to a domain is shared among several peering relationships. However, such IXP peering arrangements are almost always settlement-free, meaning the weights are set by the receiving domain, and can be enforced by the IXP at the egress to the receiving domain using the same signalling information as described above.

7.2 Policy and Incentives

While this work does not raise any ethical issues, it does raise questions about policies and incentives.

Incentives. RCS is designed to have bandwidth allocations follow the money. This is precisely what gives ISPs an incentive to deploy RCS. Purely locally, a domain can start by implementing RCS internally, treating congestion at their egress points based on the agreements at their ingress points; when congestion occurs, those with greater congestion shares receive a larger share of bandwidth. This directly provides additional value to these access agreements and can lead to greater revenue and a competitive advantage over other providers. Then, on a bilateral basis, two connected domains can agree to respect each other's congestion shares; this, again, is a value-added service to each other. A domain having such arrangements can then say that purchasing higher congestion shares not only benefits them within that domain but also in the next downstream domain. Applying this reason recursively, there are significant incentives that could drive RCS deployment. This is in stark contrast to today's Internet, where domains only control their customers' traffic locally, so customers are often forced to pursue entirely private wide-area networks at extreme cost (or, barring this option, simply coping with the congestion).

Network neutrality. There are many definitions of network neutrality, and it is important to distinguish among them. Below are three different ways network neutrality has been characterized, where we have invented our own terms to label the various lines of thought. Our definitions are, of course, overly simplified, but they capture the core differences between them. We list them in descending order of strictness:

- Mechanistic Neutrality: This definition focuses on the mechanisms for packet handling, ruling out various forms of discrimination in these mechanisms.
- Payment Neutrality: This definition also focuses on limiting discrimination in the packet handling mechanisms but specifically allows discrimination based on how much customers pay for service. Included in this are priority schemes where you can pay to obtain higher priorities.
- Competitive Neutrality: Misra's vision of a network-neutral Internet: "a platform where ISPs provide no competitive advantage to specific apps/services, either through pricing or QoS" [7, 34]. Here the focus is not on the mechanism but on the impact of discrimination on the ecosystem. Zero-rating, as discussed in §3, is an example of a violation of this form (and the previous two forms) of network neutrality, as is blocking traffic from a competitor.

RCS clearly satisfies the second and third definitions of network neutrality, but not the first, in that while it allows the level of payment to dictate the handling of the packets, no other factors influence packet handling (and thus, there are no competitive advantages to specific apps/services). We think this is a reasonable compromise, given that network access bandwidths already depend on levels of payment, and we are merely allowing that information

to inform how congestion is handled within the network. We acknowledge that there would be significant equity issues if networks made congestion shares proportional to fees paid. However, RCS as a mechanism does not require such strict proportionality as a policy. Instead, RCS gives operators the flexibility to limit how small and how large congestion shares could be, to ensure both a basic level of access as well as a cap on resources allocated to customers with more commercial power. We note that [1] broadens the notion of network neutrality to capture various forms of fairness, which we think is an interesting topic for future work but lies outside the scope of this paper.

7.3 Deployment and Scalability

Our prototype HDWRR implementation (§6.2) is in software, which is, of course, unsuitable for Internet-scale deployment. There are two relevant technical considerations for deploying HWFS: the number of operations needed per packet, and the number of required router queues. Implementing HWFS requires (from Listing 1) $O(d)$ operations per packet where d is the depth of the weight tree. Prior work [3] indicates that d is typically ≤ 3. Thus, the computational complexity required is low and compatible with modern router hardware.

In terms of queues, in its purest form, RCS dictates that the weight tree include weights for every upstream (or downstream, for receiver-dictated weights) entity with a commercial relationship to the Internet. Of course, there are already easily millions of such entities today.

This is not within reach of current hardware but may be in the near future. We have been told confidentially by one router vendor that they will soon have products that support close to half a million queues and with approximately 7-8 levels of hierarchy. Given the current estimated average AS path length between two random destinations as 4 hops [14] and the trend of the flattening of the Internet [30], we suspect 7-8 levels of hierarchy is more than sufficient.

However, it would be better if we could reduce the state requirements of RCS to enable deployment on current hardware. To that end, we propose two potential approximations to ease the amount of state necessary for any individual router to maintain. We note that we have not evaluated these approximations at the Internet scale (nor do we believe it is practical for us to attempt this without data on current traffic patterns); we thus leave the design of an Internet-scale RCS implementation to future work.

Move scheduling to the state. Our prototype enforces bandwidth allocation at each congested link and does so on the entire tree of congestion shares. An alternative is to offload scheduling of some subtree of aggregates to upstream/downstream routers (depending on the flow of money). Of course, this upstream/downstream router is not the natural bottleneck for these aggregates, so it must be informed of its aggregate bandwidth allocation before it can enforce that allocation on its subtree. We observe that the systems we employ for endpoint control (*i.e.,* Bundler [16] and Crab [45]) perform exactly this functionality, by using a CCA on an aggregate to implicitly (or explicitly [22, 31]) signal this rate. Because the bandwidth allocations vary in time and the CCAs used on the aggregate only discover a delayed approximation of this bandwidth, the bandwidth allocations would only approximate the ideal RCS calculation.

Dynamic state assignment. Since only constrained aggregates need to have packets dropped when they exceed their limits (unconstrained aggregates experience drops at their bottlenecks), one can implement an approximation to HWFS using an amount of state that is proportional to the number of constrained aggregates, not the total number of aggregates. Such an algorithm would need to dynamically adjust the state kept as aggregate rates changed and thus would occasionally not implement HWFS precisely as defined as it adjusted its state to reflect current usage. How much savings does this offer? Our own experience attempting to measure congestion in the Internet indicates that congestion is relatively rare, and prior work from Dhamdhere et al. corroborates this: "we did not find evidence of widespread endemic congestion of interdomain links between U.S. access ISPs and directly connected transit and content providers" [21]. Since we expect that the number of constrained aggregates is small compared to the total number of aggregates, we conjecture this approximation would be an effective way of implementing RCS.

8 Related Work

We first discuss the two leading contenders to replace TCPF, and end with a very brief listing of other related work.

What about the two leading alternatives? As mentioned previously, the literature has suggested two main alternatives to TCPF. The first, as initially articulated by [36] and further explored by many, is per-flow fairness. However, the real benefit of such approaches is not that they provide a morally superior resource allocation; instead, the true benefit is that these approaches provide isolation between flows so they achieve CCAI. This approach was "dismantled" by Briscoe in [10] where he observed that the resulting allocations made no economic sense. Flows, no matter their definition (*e.g.,* per source, per destination, per source-destination pairs, per five-tuple), have no relation to the commercial arrangements of the current Internet. Of course, as Briscoe observed, TCPF makes no economic sense either; thus, per-flow-fairness – which achieves CCAI, but does not make economic sense – is strictly an improvement over TCPF. In contrast, with RCS we are searching to achieve both CCAI and economic sanity, which per-flow-fairness most definitely does not.

The other alternative to TCPF is network utility maximization (NUM), where each flow has a utility function and the goal is to maximize this utility. This approach was introduced by Kelly in [32, 33], and has generated significant literature. NUM's fundamental idea is that congestion signals can serve as shadow prices, providing a measure of how much congestion a particular flow is causing other flows. If individual CCAs optimize their own utility minus this shadow price, the system at equilibrium will maximize the sum of utilities, which is the socially optimal outcome.

This core idea could be employed in two ways. The most straightforward is to actually charge shadow prices (*i.e.,* users must pay whatever shadow price charges they incur), and then have users selfishly optimize accordingly. For this, users would be required to define their (typically unknown) utility functions in the CCA. However, for our purposes, the most relevant objection is that this

approach requires a massive change in how users are charged for Internet access and usage, which renders it explicitly outside our scope.

One could instead use congestion signals as a hint, and have CCAs respond to them as if they were self-optimizing. This approach essentially mandates a universal CCA and a universal congestion signalling mechanism at routers. Thus, this would replace the voluntary TCP-friendly paradigm with a voluntary NUM paradigm. This would not solve the incentive problem, as CCAs that ignored these congestion signals would get better service. In more limited deployments, like datacenters, this is more feasible because the network is serving the needs of the operator, not individual users. See [35] for such an example.

A far more profound difficulty with NUM, in our setting, is that it is not consistent with the granularity of the Internet's current commercial model in which entities purchase service from providers at relatively stable prices. The entities, which could be home users, enterprises, or other providers, pay their providers for being able to send and receive packets. There is some degree of utility maximization in this process, but it is at the level of these entities that purchase access, not at the level of individual network flows. RCS addresses this level of utility by ensuring that the treatment their traffic receives as it flows through the network reflects, to some degree, the level of access they purchased (as measured by their congestion shares, and the congestion shares their provider receives from its peers, etc.).

What about other related work? There is a vast literature on congestion control, which provides the context for this work and which we cannot possibly acknowledge in full. Nevertheless, three recent position papers have shed new light on issues with TCPF: [46] provides a brilliant discussion of TCP friendliness and its discontents, [42] provides a novel perspective on CCA diversity, and [13] raises questions about the role of TCPF in determining bandwidth contention. Indeed, even if TCPF does not determine bandwidth allocations today, RCS provides a principled framework rather than an arbitrary or obfuscated one.

Meanwhile, two works present alternatives to the HDWRR mechanism we described in §6. HCSFQ [48] extends classic CSFQ [43] to approximate HWFQ. We chose not to pursue this as our mechanism because it would require core switches to estimate the arrival rates of all aggregates in the hierarchy. Gearbox [27] follows a line of work approximating WFQ and achieves hierarchical calendar queues with a small number of physical FIFO queues. Extending Gearbox to HWFQ approximation would exacerbate the problem of calendar range overflow, which is why we did not explore this direction.

9 Conclusion

One might dismiss this paper as being *unnecessary* (today's Internet works reasonably well), *untested* (its proposed mechanism has not yet been validated at scale), *impractical* (its mechanisms cannot be deployed in the near term), and *hysterical* (seeing the adoption of BBR as an apocalypse rather than a mere blip in the long history of rough adherence to TCP-friendliness). We willingly plead guilty on all counts, but see these objections as largely missing our point.

This paper is definitely not claiming to solve an urgent practical problem with a well-tested and easily-deployable solution. Instead, our goal is to address a fundamental conceptual problem that has remained unresolved since Nagle's 1985 paper [36], which is: *how do we reconcile the goal of CCA independence with the Internet's commercial realities?* This is an important question since the former concern (CCAI) is undeniably desirable, as it would enable much more rapid CCA innovation, while the latter concern (commercial realities) is unlikely to fundamentally change in the foreseeable future.

We think that such a dilemma deserves our intellectual attention even without an imminent crisis. Our approach appears to have resolved this conceptual dilemma. The solution is not simple – conceptually or practically – and there is much more that remains to be done to both understand this approach theoretically and engineer it to be more practically deployable. Nonetheless, we believe this paper represents a helpful first step.

Acknowledgements

We thank Aurojit Panda, Tejas Narechania, our shepherd Sachin Katti, and the anonymous reviewers for their comments. This work was supported by NSF grants 2201489, 2212102, 2213387, and 2242502, and by funding from IBM, Intel, and Broadcom.

References

[1] Muhammad Abdullah, Pavlos Nikolopoulos, and Katerina Argyraki. Caching and Neutrality. In *HotNets*, 2023. Cited on page 11.

[2] Mohammad Alizadeh, Albert Greenberg, David A. Maltz, Jitendra Padhye, Parveen Patel, Balaji Prabhakar, Sudipta Sengupta, and Murari Sridharan. Data Center TCP (DCTCP). In *SIGCOMM*, 2010. Cited on page 2.

[3] Todd Arnold, Jia He, Weifan Jiang, Matt Calder, Italo Cunha, Vasileios Giotsas, and Ethan Katz-Bassett. Cloud Provider Connectivity in the Flat Internet. In *IMC*, 2020. Cited on page 11.

[4] Venkat Arun, Mohammad Alizadeh, and Hari Balakrishnan. Starvation in End-to-End Congestion Control. In *SIGCOMM*, 2022. Cited on page 2.

[5] Venkat Arun and Hari Balakrishnan. Copa: Practical Delay-Based Congestion Control for the Internet. In *NSDI*, 2018. Cited on page 1.

[6] Niloofar Bayat, Richard Ma, Vishal Misra, and Dan Rubenstein. Zero-Rating and Net Neutrality: Who Wins, Who Loses? In *SIGMETRICS*, 2021. Cited on page 3.

[7] Niloofar Bayat, Richard T. B. Ma, Vishal Misra, and Dan Rubenstein. Big Winners and Small Losers of Zero-rating. *ACM Trans. Model. Perform. Eval. Comput. Syst.*, 7(1), Sep 2022. Cited on pages 3 and 10.

[8] Jon C. R. Bennett and Hui Zhang. Hierarchical Packet Fair Queueing Algorithms. In *SIGCOMM*, 1996. Cited on page 5.

[9] BEREC. Zero-Rating. https://berec.europa.eu/eng/open_internet/zero_rating/, 2015. Cited on page 3.

[10] Bob Briscoe. Flow Rate Fairness: Dismantling a Religion. In *SIGCOMM*, 2007. Cited on pages 2 and 11.

[11] Bob Briscoe, Koen De Schepper, Marcelo Bagnulo, and Greg White. Low Latency, Low Loss, Scalable Throughput (L4S) Internet Service: Architecture. IETF Internet Draft: draft-ietf-tsvwg-l4s-arch-11, 2021. Cited on page 2.

[12] Lloyd Brown, Ganesh Ananthanarayanan, Ethan Katz-Bassett, Arvind Krishnamurthy, Sylvia Ratnasamy, Michael Schapira, and Scott Shenker. On the Future of Congestion Control for the Public Internet. In *HotNets*, 2020. Cited on pages 1, 2, 4, 6, and 8.

[13] Lloyd Brown, Yash Kothari, Akshay Narayan, Arvind Krishnamurthy, Aurojit Panda, Justine Sherry, and Scott Shenker. How I Learned to Stop Worrying About CCA Contention. In *HotNets*, 2023. Cited on page 12.

[14] 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. In *RIPE 78*, 2018. Cited on page 11.

[15] CAIDA. AS Relationships. https://www.caida.org/catalog/datasets/as-relationships/, 2022. Cited on pages 6, 7, and 9.

[16] Frank Cangialosi, Akshay Narayan, Prateesh Goyal, Radhika Mittal, Mohammad Alizadeh, and Hari Balakrishnan. Site-to-Site Internet Traffic Control. In *EuroSys*, 2021. Cited on pages 3, 4, 5, 10, 11, and 14.

[17] Yi Cao, Arpit Jain, Kriti Sharma, Aruna Balasubramanian, and Anshul Gandhi. When to Use and When Not to Use BBR: An Empirical Analysis and Evaluation Study. In *IMC*, 2019. Cited on page 1.

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

[19] Neal Cardwell, Yuchung Cheng, Kevin Yang, David Morley, Soheil Hassas Yeganeh, Priyaranjan Jha, Yousuk Seung, and Van Jacobson. BBRv3: Algorithm Bug Fixes and Public Internet Deployment. In *IETF*, 2023. Cited on page 1.

[20] A. Demers, S. Keshav, and S. Shenker. Analysis and Simulation of a Fair Queueing Algorithm. In *SIGCOMM*, 1989. Cited on pages 1 and 2.

[21] Amogh Dhamdhere, David D. Clark, Alexander Gamero-Garrido, Matthew Luckie, Ricky K. P. Mok, Gautam Akiwate, Kabir Gogia, Vaibhav Bajpai, Alex C. Snoeren, and Kc Claffy. Inferring Persistent Interdomain Congestion. In *SIGCOMM*, 2018. Cited on page 11.

[22] Nandita Dukkipati and Nick McKeown. Why Flow-Completion Time is the Right Metric for Congestion Control. In *SIGCOMM*, 2006. Cited on pages 3 and 11.

[23] Sally Floyd. HighSpeed TCP for Large Congestion Windows. RFC 3649, IETF, 2003. Cited on page 1.

[24] Sally Floyd and Kevin Fall. Promoting the Use of End-to-End Congestion Control in the Internet. In *IEEE/ACM Trans. Netw.*, 1999. Cited on page 1.

[25] Drew Fudenberg and Jean Tirole. *Game Theory*. MIT Press, 1991. Cited on page 6.

[26] Lixin Gao and J. Rexford. Stable Internet Routing Without Global Coordination. *IEEE/ACM Trans. Netw.*, 2001. Cited on pages 1 and 5.

[27] Peixuan Gao, Anthony Dalleggio, Yang Xu, and H. Jonathan Chao. Gearbox: A Hierarchical Packet Scheduler for Approximate Weighted Fair Queuing. In *NSDI*, 2022. Cited on page 12.

[28] Prateesh Goyal, Akshay Narayan, Frank Cangialosi, Srinivas Narayana, Mohammad Alizadeh, and Hari Balakrishnan. Elasticity Detection: A Building Block for Internet Congestion Control. In *SIGCOMM*, 2022. Cited on page 1.

[29] Nikhil Handigol, Brandon Heller, Vimalkumar Jeyakumar, Bob Lantz, and Nick McKeown. Reproducible Network Experiments Using Container-Based Emulation. In *CoNEXT*, 2012. Cited on page 8.

[30] Geoff Huston. BBR, the new kid on the TCP block. https://blog.apnic.net/2017/05/09/bbr-new-kid-tcp-block/, 2017. Cited on page 11.

[31] Dina Katabi, Mark Handley, and Charlie Rohrs. Congestion Control for High Bandwidth-Delay Product Networks. In *SIGCOMM*, 2002. Cited on pages 3 and 11.

[32] Frank Kelly. Charging and Rate Control for Elastic Traffic. In *European transactions on Telecommunications*, 1997. Cited on pages 2 and 11.

[33] Frank P Kelly, Aman K Maulloo, and David KH Tan. Rate Control for Communication Networks: Shadow Prices, Proportional Fairness and Stability. In *Journal of the Operational Research Society*, 1998. Cited on pages 2 and 11.

[34] Vishal Misra. Half the Equation and Half the Definition. http://peerunreviewed.blogspot.com/2015/12/what-is-definition-of-net-neutrality.html,, 2015. Cited on page 10.

[35] Kanthi Nagaraj, Dinesh Bharadia, Hongzi Mao, Sandeep Chinchali, Mohammad Alizadeh, and Sachin Katti. NUMFabric: Fast and Flexible Bandwidth Allocation in Datacenters. In *SIGCOMM*, 2016. Cited on page 12.

[36] J. Nagle. On Packet Switches with Infinite Storage. RFC 970, IETF, 1985. Cited on pages 2, 11, and 12.

[37] Gurobi Optimization. Gurobi optimizer: The world's fastest solver. https://www.gurobi.com/solutions/gurobi-optimizer/. Cited on page 7.

[38] Adithya Abraham Philip, Ranysha Ware, Rukshani Athapathu, Justine Sherry, and Vyas Sekar. Revisiting TCP Congestion Control Throughput Models & Fairness Properties At Scale. In *IMC*, 2021. Cited on page 1.

[39] S. Shenker R. Braden, D. Clark. Integrated Services in the Internet Architecture: an Overview. RFC 1633, IETF, 1994. Cited on page 3.

[40] Barath Raghavan and Alex C. Snoeren. Decongestion Control. In *HotNets*, 2006. Cited on page 2.

[41] M. Shreedhar and George Varghese. Efficient Fair Queueing Using Deficit Round Robin. In *SIGCOMM*, 1995. Cited on page 8.

[42] Anirudh Sivaraman, Keith Winstein, Pratiksha Thaker, and Hari Balakrishnan. An Experimental Study of the Learnability of Congestion Control. In *SIGCOMM*, 2014. Cited on page 12.

[43] Ion Stoica, Scott Shenker, and Hui Zhang. Core-Stateless Fair Queueing: Achieving Approximately Fair Bandwidth Allocations in High Speed Networks. In *SIGCOMM*, 1998. Cited on page 12.

[44] Ion Stoica, Hui Zhang, and TS Eugene Ng. A Hierarchical Fair Service Curve Algorithm for Link-sharing, Real-time and Priority Services. In *SIGCOMM*, 1997. Cited on page 5.

[45] Ammar Tahir and Radhika Mittal. Enabling Users to Control their Internet. In *NSDI*, 2023. Cited on pages 3, 4, 5, 10, 11, 14, and 15.

[46] 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. In *HotNets*, 2019. Cited on pages 1 and 12.

[47] Ranysha Ware, Matthew K. Mukerjee, Srinivasan Seshan, and Justine Sherry. Modeling BBR's Interactions with Loss-Based Congestion Control. In *IMC*, 2019. Cited on page 1.

[48] Zhuolong Yu, Jingfeng Wu, Vladimir Braverman, Ion Stoica, and Xin Jin. Twenty Years After: Hierarchical Core-Stateless Fair Queueing. In *NSDI*, 2021. Cited on page 12.

[49] Doron Zarchy, Radhika Mittal, Michael Schapira, and Scott Shenker. Axiomatizing Congestion Control. In *SIGMETRICS*, 2019. Cited on page 1.

[50] Danesh Zeynali, Emilia N. Weyulu, Seifeddine Fathalli, Balakrishnan Chandrasekaran, and Anja Feldmann. Promises and Potential of BBRv3. In *PAM*, 2024. Cited on page 1.

Appendices

Appendices are supporting material that has not been peer-reviewed.

Appendix A HDWRR Implementation

```
Schedule(n) for root node:
  while True:
    n.deficit += n.quantum
    for c in n.children:
      n.deficit -= c.quantum
      Schedule(n, c.quantum)
Schedule(n, credits) for non-leaf nodes:
  n.deficit += credits
  while n.deficit > 0:
    c = n.children.head
    q = min(n.deficit, c.quantum + c.leftover)
    n.deficit -= q
    Schedule(c, q)
    if c is inactive:
      n.children.remove(c)
      if n.children is empty:
        set n as inactive
        n.deficit = 0
    if n.deficit == 0:
      c.leftover += c.quantum - q
      n.children.rotate() // move c to tail
Schedule(n, credits) for leaf node:
  n.deficit += credits
  while n.queue not empty and
            n.deficit > n.queue.head.length:
    pkt = n.queue.dequeue()
    n.deficit -= pkt.length
    transmit(pkt)
  if n.queue is empty:
    n.deficit = 0 and set n as inactive
```

Listing 1: HDWRR implementation.

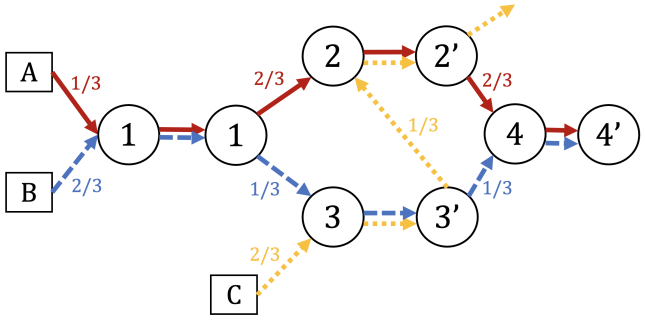
Appendix B Game Theory Examples

B.1 Multiple Nash

We now illustrate that topologies can have multiple equilibria, even with RCS. Figure 6a depicts an example with two Nash equilibria, as both $(\frac{2}{3}, \frac{1}{3}, \frac{1}{3})$ and $(\frac{1}{2}, \frac{1}{2}, \frac{1}{2})$ are Nash equilibria. If any flow attempts to increase they immediately incur persistent losses at one of the egress points. Recall that when we modelled realistic topologies from CAIDA data (§5.2), we found zero instances of this behavior with RCS.

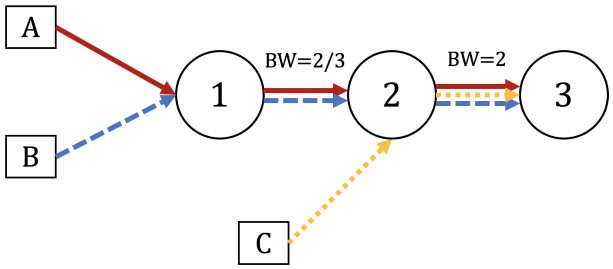
B.2 Non-Nash Stackelberg

The example in Figure 6b depicts a scenario with no Nash but a single Stackelberg equilibrium. One can verify that the only solution that satisfies the necessary conditions for a Nash equilibrium as outlined in §5.1 is $(\frac{1}{3}, \frac{1}{3}, \frac{4}{3})$, which is not a Nash equilibrium. This is because A could increase its rate by a small ϵ, which reduces B's throughput while C's is unchanged, thereby increasing its own total throughput by ϵ. While not a Nash equilibrium, this solution



a: **Example with 2 Nash equilibria $(\frac{2}{3}, \frac{1}{3}, \frac{1}{3})$ and $(\frac{1}{2}, \frac{1}{2}, \frac{1}{2})$ with no intermediate equilibria. All capacities are 1.**

Weights	1	2
A	2	1
B	1	3
C	0	4



b: **Example with no Nash equilibria but a Stackelberg equilibria.**

Figure 6: **Example topologies with undesirable convergence properties.**

is a Stackelberg equilibrium since if A increases its rate then (because Stackelberg considers how streams respond to changes by the leader) in addition to B reducing its rate, C will slightly increase its rate leading to losses for A; this causes A's throughput to be less than its sending rate, which eliminates its utility entirely. Thus, A is best off at its current rate in the Stackelberg sense. Thus, this is an example of a scenario that has no Nash equilibrium, but has a Stackelberg equilibrium.

Appendix C Endpoint Control

In section 4.3 we introduced the principle of endpoint control which dictates that endpoints should be responsible for determining the composition of their streams rather than the network. However, enforcing such control requires enacting policies on the handling of packets at the bottleneck links where queues build up, and these bottleneck links are often not under the control of the endpoint. Here we turn to two recent works that enable endpoint control in the face of remote bottlenecks, Bundler [16] and CRAB [45] that edge domains can use alongside RCS to enforce their scheduling policies.

The core idea of Bundler [16] is to move a stream's queue buildup from the bottleneck point (outside the endpoint's domain) to the sender to enable the sender to enforce its policies. Migrating the queue requires dynamically and accurately estimating the rate at

which the stream is sending as well as rate limiting traffic at the sender side. These two tasks are accomplished with a sendbox and receivebox that interpose between the sender and receiver. The sendbox and receivebox measure congestion signals and utilize a delay-based congestion control algorithm at the sendbox to determine the sending rate of the stream. The sendbox then rate limits outgoing traffic according to the determined rate which builds a standing local queue. The choice of delay-based congestion control for rate estimation is both key to the mechanism (a loss-based control loop would fill the queue at the bottleneck link) and enables constituent flows that are delay-sensitive to achieve low delay.

CRAB [45] instead focuses on enabling users to weight incoming traffic bottlenecked at their access link and receive weighted max-min fair share rates. Computing these rates requires the weights, bottleneck capacity, and the demands. Determining flow's demands directly from perceived rates after the bottleneck link is difficult because it is unclear whether the flow is limited by its own demand or by competition at the bottleneck. To combat this CRAB throttles flows at the receiver slightly under their perceived rate and determines what the flow does when offered more bandwidth to understand whether the flow's demand has been met or not, making sure that a flow never receives more than it is entitled to under weighted max-min fairness.

More specifically, CRAB estimates the total capacity of the receiver's access link as well as the perceived rate of each flow in rounds. After a round of rate estimation flows are analyzed to determine if they belong to one or more of: growing (using the extra bandwidth it was lent), saturating (either growing or at their assigned bandwidth), or non-saturating (under their assigned bandwidth). If there is at least one non-saturating flow and one saturating flow then CRAB starts the reallocation process. During re-allocation, CRAB first determines the total amount of bandwidth each flow can lend. Then it runs a water-filling algorithm to determine the new rates for each flow, which apportions this excess capacity across each of the saturating flows. If lending flows are later classified as growing, then CRAB returns all of its lent bandwidth. Lastly, CRAB must maintain a current understanding of the capacity of the bottleneck link to accurately determine rates. Ideally, CRAB only drops the observed capacity when the available capacity has actually changed, not when flow demands happened to decrease suddenly. Therefore, CRAB reacts to drops in total observed capacity by first attempting to reallocate bandwidth if there is any saturating flow and otherwise dropping the observed capacity. CRAB also probes for higher total throughput by increasing rates for a measurement period. These changes in observed capacity result in further recalculations of the weighted max-min fair rate for each flow.