# Principles for Internet Congestion Management

Paper #362

## Abstract

Given the technical flaws with, and the increasing nonobservance of, the TCP-friendliness paradigm, we must rethink how the Internet should manage congestion. We explore this question from first principles but 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 that are independent of which congestion control algorithms flows use, but consistent with the Internet's economics. We evaluate how well RCS achieves this goal through game-theoretic calculations and simulations as well as network emulation.

## 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 [1]: "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 congestion control solutions are available in private deployments such as datacenters, enterprises, and private WANs, where 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 [3] TCPF, and that our understanding of the dynamics of CCAs breaks down at scale [4]. In addition, TCPF limits CCAs' ability to ramp up quickly and achieve full efficiency [5] and hinders the emergence of new delay-sensitive CCAs (as shown by Copa [6] and Nimbus [7]). 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 [8–10] has nonetheless 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, and 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 [11]). We thus 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.

## 2. REPLACING TCPF

Before addressing how we might 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 with the total 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 that the network actively enable all reasonable CCAs to achieve the same bandwidth in the same static circumstances. We call this CCA independence (CCAI), and it removes the need for a single standard CCA and instead fosters widespread CCA diversity and innovation. Given our assumption about no

[1] At the time of [1], 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 [2] was to retain TCPF at lower speeds and create new standards for behavior at these higher speeds.

[2] While BBRv2 is less unfair than the original BBR, it is still not TCP-friendly. However, it is not the degree of BBR's violation of TCPF that is our concern, but the lack of resistance to deploying CCAs that do not satisfy TCPF, which applies to both BBR and BBRv2.

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 [12, 13]) and network utility maximization (*i.e.,* as inspired by the work of Kelly [14, 15]) – 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 [16]); flows don't have "rights" to bandwidth or utility, nor are they the units for which users are charged.

Thus, to meet the challenge above, we need a new approach, and this paper proposes "Recursive Congestion Shares" (RCS). RCS is similar to (and inspired by) the work in [17], but we find (as described in §8 and as evaluated in §6.2) that the mechanism that paper proposed does not achieve CCAI. Here, we pursue the quest for CCAI far more deeply by describing a set of principles that should guide our design and then defining a framework based on those principles that does indeed achieve the desired goal of CCAI. However, before turning to those principles, we address three key questions.

**What is a "reasonable" CCA?** Recall that 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. This is compatible with all deployed CCAs we know of. Loss-based CCAs incur persistent low losses (but not high loss), while BBR can incur occasional significant, but not persistent, losses (*e.g.,* if a flow encounters a quick reduction in bandwidth). Indeed, there seems to be no reason to ever incur persistent *and* significant losses, so we assume the existence of a community agreement that such CCAs are not acceptable. In contrast to TCPF, this requirement is relatively easy to enforce and adhere to, and imposes no serious limitations on desirable CCA design properties. Moreover, this requirement prevents flows from needlessly harming other flows, as in the dead-packet phenomena discussed in [18].

**What is the role of CCAs post-CCAI?** In a static scenario (where the rates of other flows and the available bandwidth are fixed), CCAI implies that all reasonable CCAs achieve the same level of bandwidth. There is no reason for CCA innovation in such static scenarios, but Internet conditions are constantly changing, and different CCAs could have very different tradeoffs in terms of how actively they explore network conditions and what levels of loss and delay they incur while doing so [5]. Since applications have different tolerances for loss, delay, and latency, a range of CCAs will remain necessary to meet their needs.

**Is RCS deployable?** Our goal with RCS is to provide a principled conceptual framework for Internet bandwidth allocation rather than design a fully operational mechanism for immediate deployment. However, nothing in our design is inherently impossible. Nonetheless, there are significant barriers to its deployment, as RCS would clearly require changes in network operations and customer expectations, and RCS would need modifications to deal with the complexities of the modern Internet (*e.g.,* rare violations of valley-free routing). We briefly discuss deployment incentives for RCS in §7.2, which might cause carriers to overcome these concerns.

Importantly, as we discuss in §7.3, deploying RCS at Internet scale will likely require approximations relative to our prototype implementation that reduce the amount of state the implementation must maintain. Evaluating the approximations we propose would require a large-scale measurement study that is beyond the scope of this paper. Overall, we argue that as a research community, our focus should first be on providing the intellectual underpinning for essential design questions, such as how to replace TCPF; the detailed and deployable mechanisms can come later, as we as a community gather more measurements and gain more engineering and operational experience with approximating prototypes.

## 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 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 do not 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 only be enforced when the network is congested, and should be described in terms of relative rights.**[3]

This means that when the network is congested, it uses packets' *relative rights* (explained next) to determine which to drop. In contrast, expressing bandwidth rights as guaranteed rates (*i.e.,* specifying the absolute levels of end-to-end

[3] 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.

bandwidth a user can expect) would greatly reduce statistical multiplexing, and therefore be impractical. This condition does not disallow RCP [19] or XCP [20], but does disallow IntServ [21]; 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 [22]) and receivers making decisions about what traffic exits the network (using a mechanism such as Crab [23]). Intermediate cases should be determined by the money flow at the point of congestion. In particular, because of the dangers of "zero-rating" [4] [24–26], it is important that receivers, not senders, make decisions about what traffic they receive.

## 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).

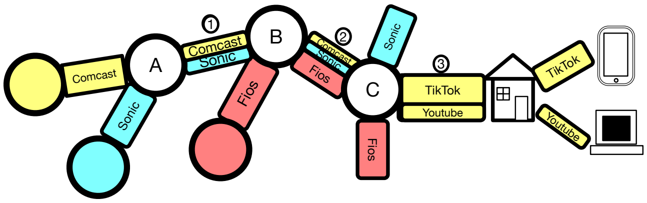

**Figure 1: Illustration of RCS principles: (1) Relative rights. (2) These rights should be applied recursively. This queue first schedules traffic between Fios (red) and the aggregate formed by Sonic (blue) and Comcast (yellow). If it must drop a packet from this aggregate, it considers Sonic and Comcast's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

### 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 being received by another. The term "CCA" refers to how a stream responds to congestion; such a response is in fact is made up of several distinct flow-based congestion control algorithms and application behaviors (such as opening additional connections or throttling streaming data), but for convenience we model it as a single CCA.[5] Hereafter, for cases where we have multiple ingresses and egresses, as in

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

[5] 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 [22, 23] to implement that policy when the congestion occurs elsewhere in the network.

switches and domains, we use the term $066A460C4^{1}8\bullet U^{\circ}$ to refer to traffic entering at ingress $8$ and leaving at egress $U$.

**Relative Rights at a single link.** Consider a single link with several streams sending at rates $A_8$. We denote the resulting egress bandwidths (*i.e.,* the rate leaving the link from each stream) by $O_8$ with $A_8 \quad O_8$: strict inequality represents when the network drops packets from stream $8$, and we call such streams "constrained". The link is work-conserving, so $_8 O_8 = " \ \# » \ _8 A_8 \bullet \ ¼$ where is the bandwidth of the link. In this context, we define *relative rights* in terms of weights $F_8$, and allocate bandwidth to constrained streams proportional to those weights. Specifically, for any two streams $8\bullet 9$ with $O_8 < A_8$ and $O_9 < A_9$, the following holds: $\frac{O_8}{F_8} = \frac{O_9}{F_9} \quad \frac{O_:}{F_:}$ for all other streams $:$. The equality between $8$ and $9$ requires that two constrained streams receive bandwidth proportional to their weights. The inequality between $9$ and $:$ requires that no unconstrained stream is getting more than if it were constrained. This definition of relative rights implies that $O_8 = " \ \# »A_8 \bullet F_8 \_ ¼$ where $\_ \quad 0$ is the smallest value that allows $_8 O_8 = " \ \# » \ _8 A_8 \bullet \ ¼$.

This results in what we call "pipe-like" behavior. As a stream increases its bandwidth demand $A_8$, 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: $O_8 = \min »A_8 \bullet A_8 \frac{\acute{1}}{_9 A_9} ¼$, and the average packet delay (which is the same for all streams) is some function of $_9 A_9$ that increases sharply as the quantity reaches the link capacity. If we fix all other $A_9$, $O_8$ is strictly monotonic in $A_8$ and stream $8$'s packets can experience significant losses and delays even if $A_8$ is very small (*e.g.,* if the remaining load $_{9<8} A_9$ is 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 the weights $F_8$ for each ingress $8$ applied at all egresses $U$. 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 $066A460C4^{1}8\bullet U^{\circ}$ from ingress $8$ to egress $U$ has a weight $F_8^U$. For simplicity, we do not embrace this generalization in our treatment here, but our results apply to this case as well.

However, 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 $= 1$, $_U F_8^U = 1$ for all $8$, and weights $F_8^U$ for a given $8$ 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 there are two ingresses – $8$ and $9$ – sending traffic to two egresses $U\bullet V$. 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 $A_8$ such that $A_8 = O_8$. Assume $8$ sends all of its traffic, of rate $A_8$, to $U$ while $9$ splits its traffic, of total rate $A_9$, between $U$ and $V$ in proportions $G$ and $^{1}1 \quad G^{\circ}$. Then the weight of $066A460C4^{1}8\bullet U^{\circ}$ is $1$, of $066A460C4^{1}9\bullet U^{\circ}$ is $G$, and of $066A460C4^{1}9\bullet V^{\circ}$ is $1 \quad G$. The only allocation choices where ingress $9$ does not incur persistent losses at egress $U$ are (i) $G = 1$ and $A_9 = \frac{1}{2}$ (with $A_8 = \frac{1}{2}$) and (ii) $G = 0$ and $A_9 = 1$ (with $A_8 = 1$). This is because, as soon as $9$ 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 $F_8$ 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 [17], we call these weights $F_8$ congestion shares, and they are determined as part of a user's agreement with their domain. The value of $F_8$ 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 now discuss how to extend that approach across multiple domains using the second principle.

**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,[6] 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 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 "Follow the Money" mean?** In the Internet, money typically flows from end users to domains and onward to other domains which provide broader routing reach and eventually ending in the Tier 1 providers that 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 to transmission against the flow of money. We use this observation to devise 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 exceed 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 does this produce 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 [11], we believe that 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 $U$ is to a customer, then all subsequent hops are to customers (and they determine all the hierarchical weights assigned to the aggregates at that egress). Second, if the egress $U$ is to a peer or provider, then all previous hops are from customers (and they determine all the hierarchical weights assigned to the aggregates at that egress). As a result, all aggregates at a given egress $U$ 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 $F_{8\bullet U}$ as the weight assigned to $066A460C4{}^{1}8\bullet U^{\circ}$. Taking the case where the weights at an egress $U$ were assigned recursively by previous ingresses, the calculation goes as follows: first calculate the allocation to each $066A460C4B{}^{1}8\bullet U^{\circ}$ for all $8$ using the weights assigned by ingresses $8$. That produces a set of allocations $0_8$. Then, for each $066A460C4{}^{1}8\bullet U^{\circ}$, calculate the allocations for all of the aggregates that entered through ingress $8$ (with the weights assigned by their previous hop ingresses), treating the total bandwidth as $0_8$. 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 [27, 28].

**Example.** We use the example in Figure 2 to explain HWFS. In the diagram, packets flow in the direction of the black arrows, and money flows strictly upward from users (senders

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

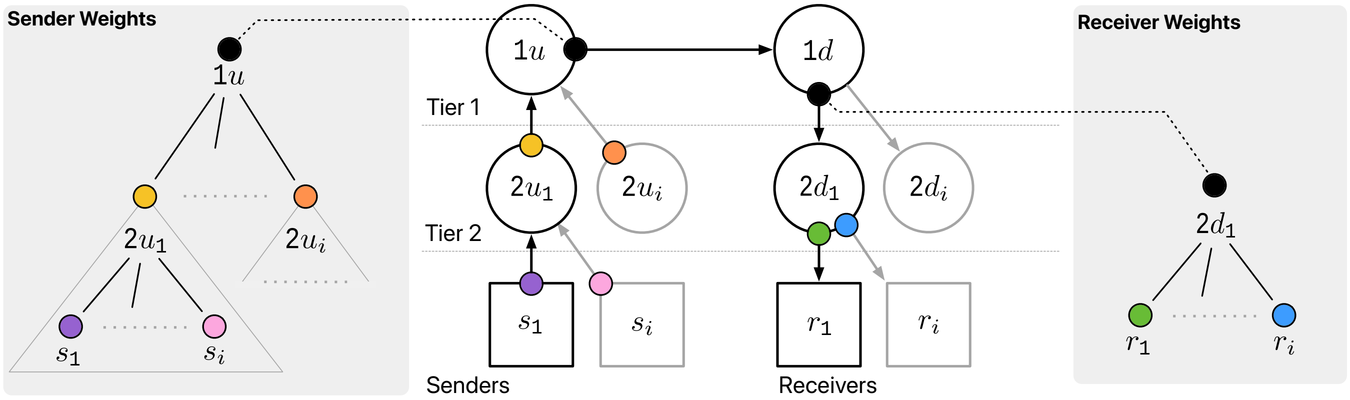

**Figure 2: The center provides an example of how packets and money flow in RCS. By considering the recursive economic relationships that comprise the Internet, we can construct a tree of relative rights (*i.e.,* weights) corresponding to each egress of each domain. For example, the left and right shaded regions show this hierarchical weight-tree for 1u's egress and 1d's egress, respectively.**

and receivers) to customer domains and then to provider domains. In our example, domains $1D$ and $13$ peer with each other. Domains $2D_1 \dots 2D_8$ are $1D$'s customers, and $2D_1$ itself has customers $B_1 \dots B_8$. Similarly, domains $23_1 \dots 23_8$ are $13$'s customers, and $23_1$ itself has customers $A_1 \dots A_8$. In this example, we specify that the link between $1D$ and $13$ and between $13$ and $23_1$ are congested, but the other links are not.

A stream from $B_1$ to $A_1$ will traverse $2D_1$, $1D$, $13$, and $23_1$. At each egress, it is possible to apply HWFS according to the relevant customer agreements at that domain. For example, at $2D_1$'s egress it would apply weights corresponding to its access agreements with $B_1 \dots B_8$. However, $2D_1$'s egress is not congested; as a result, no bandwidth enforcement is needed there. $1D$ carries traffic from $2D_1$ to $1D$'s various egresses. At the egress to $13$, $1D$ first applies the relative rights between $2D_1$ (as a whole) and its other customers ($2D_8$), and then for the traffic belonging to $2D_1$ then applies the weights of $2D_1$'s customers $B_8$. Once packets reach $13$'s egress, they begin flowing against the direction of money; therefore, the receiver's (*i.e.,* $A_1$'s) access agreement determines its relative rights. More specifically, at its congested egress link to $23_1$, $13$ allocates the bandwidth between traffic headed to the various the $A_8$ using the relative rights determined by $23_1$'s commercial agreements with these customers.

Note that in this case the packet travels through two Tier 1 domains, with a peering link between them; some paths (*e.g.,* one between $B_1$ and a customer of $2D_2$) do not traverse a peering link; in this case, $1D$'s egress to $2D_8$ would apply weights corresponding to the receiving domain, following the tie-breaking rule.

To summarize, weights are assigned to ingress aggregates on the way up in a hierarchical fashion (building up a tree of weights): *i.e.,* all packets entering at ingress $8$ are part of the same aggregate with weight $F_8$ throughout the domain they are entering. When receiver weights apply, weights are assigned to egress aggregates in a hierarchical fashion (peeling off the hierarchy of weights as they approach the egress to the network): *i.e.,* all packets exiting at egress $U$ are considered part of the same aggregate with weight $F_U$ throughout the domain they are exiting. No weights are assigned based on top-level peering arrangements.

### 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 [22] and Crab [23] provides a mechanism whereby users can remotely control the internal composition of the stream; we provide more detail in §6. 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 when sender weights apply, and vice versa for the receiving user.

## 5. DOES RCS ACHIEVE CCAI?

We have, thus far, described a scheme, RCS, that is consistent with the Internet's economic model. We now ask the question, does RCS achieve 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". The "game" is defined by the network topology and bandwidth allocation rules (*i.e.,* RCS) that determines, given a set of input rates $A_8$ what are the resulting throughput rates $0_8$ (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 $1$ if their output rate is less than their sending rate (as they are experiencing persistent drops). The question is, what kinds of equilibria do these games converge to? We use two equilibrium concepts from game theory.

The first relevant equilibrium concept is the familiar *Nash equilibrium* [29]. 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 $8, A_8$ is at the maximal value such that $0_8 = A_8$ assuming all players other than $8$ 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* [29]. 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 $8$) occurs when the leader $8$ has chosen its optimal strategy (has chosen the maximial value of $A_8$ such that $0_8 = A_8$), 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 there is little doubt as to what the stable equilibrium of the game is, 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 equilibrium (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. Note that such manipulation would be very hard to achieve in practice 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 equilibrated) and then search for its optimal. Neither seems reasonable in real settings, where users have little knowledge of other users and workloads change rapidly.

### 5.2 Does Greed Pay?

To understand whether RCS can create the possibility of multiple Nash equilibria or non-Nash Stackelberg equilibria, we consider two very different models. The first, which we design to have the possibility of pathological cases, is based on a random topology, with each stream taking a random path, and with weights assigned randomly 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 and there is no need to refer to previous hops. We start with a fully-connected 10-node topology and then generate 40 streams of path length 4 by picking a random sequence of nodes.

The second model uses topologies sub-sampled from the CAIDA AS relationships dataset [30]. 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 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 [31] to evaluate the two models. We disallow streams from incurring persistent losses and we run calculations on several different topologies from each model.

**Random Model.** The MILP starts by defining allocation variables where $0_B$ denotes the allocation that stream $B$ receives in an equilibrium. From a given scenario, we define a set of links $!$ with capacities $2_;$, $($ as the full set of streams, $(_;$ as the set of streams that pass through each link $;$, and a set of weights such that $F_{;\bullet B}$ denotes the weight of stream $B$ at link $;$. Then we create bottleneck indicator variables $_{;\bullet B}$

| Topology | Total | Multiple Nash | Non-Nash Stackelberg |
|---|---|---|---|
| Random | 18•723 | 338 (1"81%) | 598 (3"19%) |
| CAIDA-sampled | 2•897 | 0 | 0 |

**Table 1: Under RCS, both multiple Nash equilibria and non-Nash Stackelberg equilibria are rare.**

such that $_{;\bullet B}$ == 1 if and only if stream $B$ is bottlenecked at link $;$. Lastly, we create an indicator variable for links such that $_{;}$ == 1 if $;$ is at capacity. We then define the following constraints:

$$8_{;2!}8_{B2(4(_{;}} \quad {}_{;\bullet B} == 0 \tag{1}$$

$$8_{;2!}8_{B2(_{;}}8_{B^{0}2(_{;}}\frac{0_B}{F_{;\bullet B}} \quad \frac{0_{B^{0}}}{F_{;\bullet B^{0}}} \qquad '' \quad {}^{1}1 \qquad {}_{;\bullet B}{}^{o} \tag{2}$$

$$\tilde{O}_{8;2!} \quad {}_{;\bullet B} == 1 \tag{3}$$

$$\tilde{O}_{B2(_{;}} \quad 0_B \quad 2_{;} \quad {}_{;} \tag{4}$$

$$\tilde{O}_{8;2!} \quad {}_{;\bullet B} \quad 1 \tag{5}$$

We ensure that no stream is bottlenecked at a link not on its path with constraint (1). Constraint (2) ensures that each stream gets its weighted fair share at each congested link ( $''$ is some large constant). Constraint (3) ensures each stream $B$ is bottlenecked at exactly one link. Finally, constraint (4) ensures no link $;$ is oversubscribed.

To ensure that we characterize the full range of Nash equilibria for a given stream, we take turns optimizing $0_B$ for each stream $B$ to determine the maximum and minimum allocation for that stream under these constraints. Thus, a Nash solution to a topology yields two equilibria for every stream (though in most cases they turn out to be the same). To generate a Stackelberg solution with stream $B$ as a leader, we optimize $0_B$ and allow $B$ 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. Further, out of a total of 748•920 streams in all scenarios, only 68 (0"009%) streams benefitted 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).

**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:

$$8_{;2!}8_{B2(_{;}}8_{320''''\,_{;}}8_{066^{0}2066B_{;\bullet 3}} : 066_{;\bullet B} < 066^{0}$$
$$\frac{\acute{I}_{B\,2*_{066_{;\bullet B}}} 0_B}{F_{066_{;\bullet B}\bullet ;}} \quad \frac{\acute{I}_{B\,2*_{066^{0}}} 0_B}{F_{066^{0}\bullet ;}} \qquad '' \quad {}^{1}1 \qquad {}_{;\bullet B}{}^{o} \tag{6}$$

This additional constraint enforces recursively-weighted allocations. At each depth $3$ of the hierarchy (with $3 = 0$ being the root) for all streams $B$ in $(_{;}$ (borrowed from the flat model) we define $066_{;\bullet B}$ as the aggregate containing stream $B$ at link $;$, and $F_{066\bullet ;}$ to be the weight assigned to aggregate $066$. Lastly we define $_{;}$ to be the maximum depth of the hierarchy at link $;$, $066B_{;\bullet 3}$ to be the set of aggregates at depth $3$ of the hierarchy for link $;$, and $*_{8}$ to be the set of streams comprising aggregate $8$.

We show the results on the second row of Table 1. We found no topologies where multiple Nash or non-Nash Stackelberg equilibria existed, so there were no opportunities for a Stackelberg strategy to return any benefit to the sender.

### 5.3 Even If Greed Paid, is the Payout Collectible?

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 that equilibrium?

To model myopic game theory dynamics, we implement a "best-reply" agent which iterates through several sending rates between a min and max to check their utility. The two sending rates surrounding the best one achieved are used as the min and max for the subsequent iteration. The sender terminates when the difference between the max and min rate is lower than a small $n$.

We used our CAIDA topology generator (described above) and sampled 606 8-stream topologies in which we considered only sender weights, as well as a further 424 sampled topologies which consider both sender and receiver weights. In all of these topologies, the best-reply agents converged to a single equilibrium.

## 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 [32]) which 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 implementing using the deficit round-robin (DRR) algorithm, and (iv) FIFO scheduling.

```
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.**

## 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) [33] implementation of a fair queueing scheduler. Listing 1 shows pseudocode for our implementation. 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 quantum, 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 will assign the child node any remaining deficit, but track the remaining unassigned scheduling deficit for the next scheduling 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 use the same CAIDA-sampling topology generator that we described above in §5.2. We generated 15-stream[7] topologies, and for each topology we further generated 10 random CCA assignments, where we randomly assign one of Reno, Cubic, or BBR as each stream's CCA.

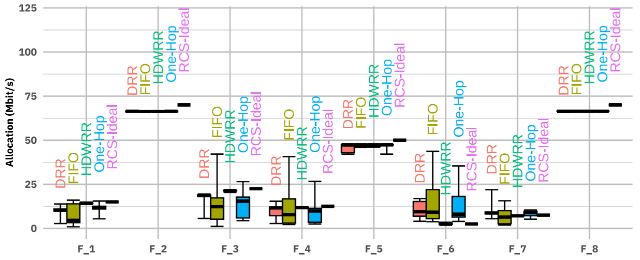

**Figure 3: Achieved allocations on an emulated topology with real CCAs.**

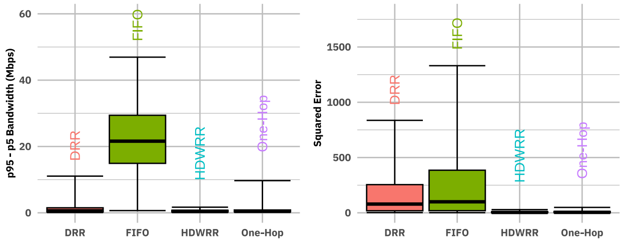

**Figure 4: Results across topologies.**

We consider five experiment configurations: (i) *RCS-Ideal* is the bandwidth allocation our simulator returns, which we include for reference; (ii) *HDWRR*, which is our implementation described above; (iii) *One-Hop*, which uses HDWRR but limits the size of the weight tree to depth 1 (as in [17]); (iv) *DRR*, which allocates per-stream, not per-aggregate, as in fair queueing; and (v) *FIFO*, which represents the status quo.

We evaluate these five configurations on two metrics. First, do they provide CCAI, *i.e.,* do all flows achieve consistent bandwidth regardless of what CCA they (or other flows) use? Second, are the bandwidth allocations consistent with their economic relationships (represented by RCS-Ideal)?

**Visualizing HWFS + CCAI** We show the results for one CAIDA-sampled topology (generated using the second model in 5.2) which has a weight tree depth of 3 in Figure 3. We observe that FIFO's allocations are CCA-dependent and far from RCS-ideal, while DRR's allocations are CCA-independent but far from RCS-ideal. Meanwhile, HDWRR's allocations are both CCA-independent (the variance between CCA assignments and across iterations is low) as well as close to RCS-ideal. Finally, One-Hop provides neither CCA-independence nor RCS-compatible allocations, confirming our refutation of their results.

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

**Results Across Topologies** We extend this to results across multiple emulated topologies in Figure 4. Here, rather than show RCS-Ideal's allocations directly (since they vary by topology), we instead show each configuration's squared-error from RCS-Ideal's allocations. On the left set of axes, we show the distribution of the spread (*i.e.,* the difference between the 95th and 5th percentile achieved bandwidth allocation per flow), CCA assignment, and topology. Overall, HDWRR achieves both the lowest error from the Ideal allocation as well as the lowest amount of spread in that 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 [22] and Crab [23] which shift congestion (and therefore the buildup of packet queues) from an egress in the network to a link at the appropriate endpoint. This requires two pieces of information: (i) what flows are bottlenecked at the same egress, and (ii) what is their aggregate throughput. Given this, the appropriate endpoint can then throttle the stream at slightly less than its allocated bandwidth, causing queues to build, and allowing it to exercise whatever traffic management it chooses. RCS can easily be extended to provide this information (and, in the case of receivers, the throughput rate can be directly measured by the endpoint).

**Complications:** As mentioned earlier, for congestion internal to a domain, the relevant links can use some form of HWFS, using the same signaling mechanisms proposed above. We think that domains will continue to be managed in a way that internal congestion is rare except for specific concentration points, and a domain can arrange to include them 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, which means that the weights are set by the receiving domain, and can be enforced by the IXP at the egress port 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.

**Network Neutrality:** The relationship between RCS and network neutrality is explored in depth in [17], but here we merely observe that there is no widely accepted definition of network neutrality. Our proposal certainly violates the dictate that "all packets should be treated equally" but (i) the existence of routers with thousands of queues and extensive scheduling features shows that this dictate is almost universally violated in practice, and (ii) most definitions of network neutrality are more subtle than this, focusing more on anticompetitive practices than on scheduling algorithms. For example, Misra in [24, 34] proposes this vision of a network neutral Internet: "Internet is a platform where ISPs provide no competitive advantage to specific apps/services, either through pricing or QoS." Our proposal certainly is consistent with this more general definition.

**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 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).

**Privacy:** Of course the incentives above require that domains reveal these congestion shares to their neighboring domains. This reveals something about a domain's set of customer contracts. While the financial details are not known, certainly the existence of various access agreements are already visible (*e.g.,* operators can observe where an enterprise's traffic enters the Internet through BGP advertisements and direct observation). Revealing the congestion shares does reveal more about these agreements than what can be discerned today, but we don't know if this is a major concern, particularly as compared to the benefit of better service under congestion.

### 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) $\$^{1}3^{o}$ operations per packet where $3$ is the depth of the weight tree. Prior work [35] indicates that $3$ 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, and it is clearly unreasonable to expect hardware to, in the worst case, maintain such a large amount of state in the weight tree. We thus 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 Internet scale (nor do we believe it is practical for us to attempt this without data about 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 the scheduling of some subtree of aggregates to upstream/downstream routers (depending on whether traffic is moving with or against 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 [22] and Crab [23]) perform exactly this functionality, by using a CCA on an aggregate to implicitly (or explicitly [19, 20]) 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 bandwith 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 will experience their drops at their bottlnecks), 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 be dynamic in adjusting which state it 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" [36]. Since we expect that the number of constrained aggregates is small compared to the total number of aggregates, we thus conjecture that this approximation would be an effective way of implementing RCS.

## 8. RELATED WORK

We first discuss the most directly relevant work, [17], and then the two leading contenders to replace TCPF, before finishing with a very brief listing of other related work.

**How does our work relate to [17]?** This paper was inspired by [17] where the basic idea of recursive congestion shares was introduced. However, the detailed approach in [17] has two major limitations. First, [17] 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, and we showed this above in §6.2. Thus, the version in [17] does not achieve CCAI even for some relatively short paths.

The second limitation is that the design in [17] ignores the role of receivers, only considering weights assigned by ingress points. This raises issues equity issues (*i.e.,* should content providers determine the priorities on my network

[8] [17] 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)

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.

**What about the alternatives?** As mentioned previously, the literature has suggested two main alternatives to TCPF. The first, as initially articulated by [12] 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 [16] 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 [14, 15], and has generated a 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 these shadow prices (*i.e.,* users must pay for whatever shadow price charges they incur), and then have users selfishly optimize accordingly. For this, the utilities would have to capture the actual utilities of users, which are hopelessly complicated and range far beyond what a transport protocol can monitor. One could then seek a more limited utility, which merely captures the quality of the transport, but then it isn't clear what that utility function would be or what is achieved by maximizing it network-wide. 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 of our scope.

One could instead use congestion signals as a hint, and have CCAs respond to them as if there 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 [37] 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 or enterprises or other providers, pay their provider 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. However, we did want to mention two recent works that have shed new light on these issues: [3] provides a brilliant discussion of TCP friendliness and its discontents, and [38] provides a novel perspective on CCA diversity.

## 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 [12], which is: *how do we reconcile the goal of CCA independence with the Internet's commercial realities?* The former is undeniably desirable, as it would enable much more rapid CCA innovation, while the latter is unlikely to fundamentally change in the foreseeable future. Such a dilemma deserves our intellectual attention even without an imminent crisis. Our approach appears to have resolved this conceptual dilemma. Of course, there is much more that remains to be done to both understand this approach theoretically and engineer it to be more practically deployable, but we hope this is a helpful first step.

## References

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

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

[3] 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. doi: 10.1145/3365609.3365855. URL https://doi.org/10.1145/3365609.3365855.

[4] Adithya Abraham Philip, Ranysha Ware, Rukshani Athapathu, Justine Sherry, and Vyas Sekar. Revisiting TCP Congestion Control Throughput Models & Fairness Properties At Scale. In *Proceedings of the 2021 ACM Internet Measurement Conference (IMC)*, IMC '21, New York, NY, USA, 2021. ACM.

[5] Doron Zarchy, Radhika Mittal, Michael Schapira, and Scott Shenker. Axiomatizing Congestion Control. SIGMETRICS, 2019.

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

[7] 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. doi: 10.1145/3544216.3544221. URL https://doi.org/10.1145/3544216.3544221.

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

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

[10] 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. doi: 10.1145/3355369.3355579. URL https://doi.org/10.1145/3355369.3355579.

[11] Lixin Gao and J. Rexford. Stable Internet Routing Without Global Coordination. *IEEE/ACM Trans. Netw.*, 2001.

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

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

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

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

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

[17] 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. HotNets, 2020.

[18] Barath Raghavan and Alex C. Snoeren. Decongestion Control. HotNets, 2006.

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

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

[21] S. Shenker R. Braden, D. Clark. Integrated Services in the Internet Architecture: an Overview. RFC 1633, 1994.

[22] Frank Cangialosi, Akshay Narayan, Prateesh Goyal, Radhika Mittal, Mohammad Alizadeh, and Hari Balakrishnan. Site-to-Site Internet Traffic Control. EuroSys, 2021.

[23] Ammar Tahir and Radhika Mittal. Enabling Users to Control their Internet. In *NSDI*, 2023.

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

[25] BEREC. Zero-rating. berec.europa.eu, 2015. URL https://berec.europa.eu/eng/open_internet/zero_rating/.

[26] Niloofar Bayat, Richard Ma, Vishal Misra, and Dan Rubenstein. Zero-Rating and Net Neutrality: Who Wins, Who Loses? SIGMETRICS, 2021.

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

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

[29] Drew Fudenberg and Jean Tirole. *Game Theory*. MIT Press, 1991.

[30] CAIDA. AS Relationships. https://www.caida.org/catalog/datasets/as-relationships/, 2022.

[31] Gurobi Optimization. Gurobi optimizer: The world's fastest solver. https://www.gurobi.com/solutions/gurobi-optimizer/.

[32] Nikhil Handigol, Brandon Heller, Vimalkumar Jeyakumar, Bob Lantz, and Nick McKeown. Reproducible Network Experiments Using Container-Based Emulation. CoNEXT, 2012. doi: 10.1145/2413176.2413206. URL https://doi.org/10.1145/2413176.2413206.

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

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

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

[36] 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. doi: 10.1145/3230543.3230549. URL https://doi.org/10.1145/3230543.3230549.

[37] Kanthi Nagaraj, Dinesh Bharadia, Hongzi Mao, Sandeep Chinchali, Mohammad Alizadeh, and Sachin Katti. NUMFabric: Fast and Flexible Bandwidth Allocation in Datacenters. SIGCOMM, 2016. URL https://doi.org/10.1145/2934872.2934890.

[38] Anirudh Sivaraman, Keith Winstein, Pratiksha Thaker, and Hari Balakrishnan. An Experimental Study of the Learnability of Congestion Control. SIGCOMM, 2014.