Principles for Internet Congestion Management

Paper #245: 12 pages body, 1 page references, 6 pages appendix

Abstract

Given the flaws with, and the increasing non-observance 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 using Recursive Congestion Shares (RCS), a substantial revision of the design in [1], to provide bandwidth allocations that are largely independent of which congestion control algorithms flows use. We evaluate how well RCS achieves this goal through analysis, calculations, and 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*, as defined by [2]: "a flow is TCP-friendly if its arrival rate does not exceed the arrival of a conformant TCP connection in the same circumstances."[1] The scope of TCP-friendliness (hereafter, TCPF), and of our paper, is limited to congestion management in the context of best-effort wide-area traffic on the public Internet. Specialized solutions are available in private deployments such as datacenters, enterprises, and private WANs, where there is a single administrative authority.

The problems with TCPF are well known and would mostly apply even if we replaced TCP with some other standard CCA. TCPF is both hard to *enforce*, so relies on broad cooperation within the ecosystem, and difficult to *achieve*, as anyone who has tried to design an algorithm to be TCP-friendly across a wide range of scenarios can attest. TCPF is also fundamentally *limiting*, restricted to the particular properties of TCP (*e.g.,* as shown in [5], TCP-friendly algorithms cannot ramp up quickly and cannot achieve full efficiency under certain conditions). Lastly, TCPF *prevents the emergence* of new delay-sensitive CCAs (*e.g.,* TCP Vegas[6], Copa [7], PCC Vivace[8]) that strive to achieve low-latency, because they cannot compete with TCP and its queue-filling behavior.

Given these flaws, current beliefs about TCPF range widely in the field (and among the authors of this paper). At one extreme are those who think that TCPF, despite its flaws, is a crucial restraining force on the ecosystem, preventing ecosystem actors from deploying ever-more-aggressive CCAs. However, this faith in TCPF has been shaken by the deployment of the BBR congestion control algorithm [9]. BBR is not TCP-friendly [10] but is nonetheless currently used by Google, Netflix, Dropbox, and Spotify for significant portions of their traffic.[2]

On the other extreme, there are those who think that while TCPF was a guiding concept twenty years ago, it has been largely disregarded starting with TCP Cubic, which was adopted as the default in Linux in 2006. To them, the deployment of BBR is merely a confirmation of this trend.

No matter where along this spectrum of opinion one belongs, it seems clear that TCPF is: (i) deeply flawed, as it is impossible to enforce, difficult to obey, limiting in what it can achieve and what it allows to emerge; and (ii) no longer adhered to by the major Internet actors. Thus, at least as an intellectual matter, we should consider whether there are suitable alternatives to the TCPF paradigm. This simple but central issue is the sole focus of this paper.

Our goal is to explore from first principles what new conceptual framework might replace TCPF. 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). We thus seek a conceptual foundation for how the Internet should share bandwidth that (i) can be implemented within the current architecture (though likely requiring additional protocols) and (ii) provides bandwidth

[1] Note that even the staunchest of the early advocates recognized that the TCP-friendly paradigm was not tenable at high-speeds, but the intent of proposals like [3, 4] was to retain TCP-friendliness 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. More importantly, it is not the degree of BBR's violation of TCPF that is our concern, but the lack of resistance to its deployment, which clearly indicates that TCPF is no longer a strict requirement for the major Internet actors.

allocations that are consistent with the current commercial arrangements.

2. REPLACING TCPF

Despite the vast literature on congestion control, there is no good replacement for TCPF that satisfies these constraints. As we discuss in Section 7, neither of the two leading contenders to replace TCPF – per-flow fairness (as in fair queueing [11, 12]) and network utility maximization (as in the work of Kelly [13, 14]) are consistent with the commercial realities of the current Internet. In particular, these two approaches focus on "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. Without an adequate replacement for TCPF in sight, the Internet's grand experiment in resource sharing is now intellectually rudderless. While the current situation is suboptimal (*e.g.,* not achieving low latency, and precluding various CCA innovations), it is hardly an urgent crisis as the Internet functions reasonably well with its current CCAs. However, the lack of a TCPF alternative creates an unfortunate intellectual vacuum, which this paper hopes to remedy.

Our goal is *not* to design a fully operational mechanism that can be immediately deployed. There is nothing in our design that is inherently infeasible, but it does require changes in network operations and customer expectations which make near-term deployment impossible. Nor do we expect our conceptual design to explicitly deal with all of the complexities of the modern Internet (*e.g.,* the rare violations of valley-free routing or other special cases). As a research community, our focus should first be on providing the intellectual underpinning for essential design questions; the detailed and deployable mechanisms can come later. These mechanisms may fall short of what the conceptual foundations require, but they should be at least guided by the North Star of first-principled reasoning.

The key issue with TCPF is that the network plays a passive role, so aggressive CCAs receive more bandwidth on congested links. We propose requiring that the network more 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. However, this goal raises three key questions.

What is a reasonable CCA? We assume a reasonable CCA can effectively use the available bandwidth in static settings. In addition, we assume reasonable CCAs do not incur persistent and significant losses. TCP incurs persistent low losses, and BBR can incur occasional significant losses (*e.g.,* if a flow encounters a quick reduction in bandwidth), but there seems no reason to ever incur persistent *and* significant losses. We thus assume that there is a community agreement that such CCAs are not acceptable, which only requires that all CCAs reduce their sending rate in response to significant losses. Note, in contrast to TCPF, this requirement is relatively easy to enforce and adhere to, and imposes no serious limitations on what can be achieved and what it allows to emerge.

What is the role of congestion control if we achieve CCAI? In a static scenario (where the rates of other flows are fixed), CCAI demands that all reasonable CCAIs achieve roughly 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 of loss, delay, and latency, we will still need to develop a range of CCAs to meet their needs.

What is new here? Our treatment is inspired by the work in [1]; our paper adopts their general approach but refutes their conclusions about achieving CCAI (Section 7), and also significantly extends their analysis and mechanisms (Sections 5 and 6). The per-flow fairness paradigm achieves CCAI and is consistent with the Internet's current architecture (though it requires the widespread deployment of new router algorithms). However, as previously mentioned, per-flow-fairness is not consistent with the Internet's commercial arrangements. Thus, the main novelty of this paper is in achieving CCAI within this constraint. This work does not raise any ethical issues, but see [1] for a discussion of how this proposal relates to network neutrality.

3. PRINCIPLES FOR "CONSISTENT" CCAI

It is far from clear what it means to be "consistent with the Internet's commercial arrangements." Here we state three principles that describe what this entails. 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.

Principle #1: Bandwidth allocations should only be enforced when the network is congested, and should be described in terms of relative rights, not guaranteed rates. This means that when the network is congested, which packets get dropped are determined by their relative rights. In contrast, expressing bandwidth rights as guaranteed rates (by which we mean specifying the absolute levels of end-to-end bandwidth a user can expect over a sizable time period) would greatly reduce statistical multiplexing, and therefore be impractical. Note that this condition does not disallow RCP [15] or XCP [16] but does disallow IntServ [17], as the former two only give ephemeral estimates, while the latter makes persistent guarantees.

Principle #2: These relative rights should be tied to current commercial arrangements, respecting their granularity, recursive nature, and flow of money. Currently 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 invoked recursively (*i.e.,* packets are delivered end-to-end because the sender's carrier has arrangements with the next-hop carrier, which has arrangements with the subsequent-hop carrier, and so forth). These interdomain arrangements are crucial to how traffic is carried, and we thus argue (following [1]) that their recursive nature must play a role in how congestion is managed. In terms of the flow of money, the vast majority [18] of Internet routes are "valley-free" in that money flows up from senders to some domain that is not paying the next hop to carry the packet; similarly, money flows up from receivers to that same domain or to a freely-peering neighboring domain.
Principle #3: While the network determines bandwidth allocations between two endpoints, the endpoints should determine the composition of traffic that flows between them. This point was raised in the Bundler work [19], and we embrace it here. As to whether the sender or the receiver makes these decisions, we invoke the reasoning in principle #2; these decisions should follow the flow of money, with senders making decisions about what traffic enters the network, receivers making decisions about what traffic exits the network, and intermediate cases determined by the money flow (as we discuss in the next section). In particular, because of the dangers of "zero-rating" [20] – where content providers pay providers to deliver their traffic, but not traffic from other content providers, to users – it is important that receivers make decisions about what traffic they receive, not the senders of that traffic.[21, 22]

4. FROM PRINCIPLES TO PRACTICE

Here we reduce these general principles to practice; the term "practice" does not refer to a specific implementation, merely to an algorithm for calculating bandwidth allocations that is consistent with these principles. In defining this algorithm, we only consider allocations in static settings, where a fixed set of users are sending at a fixed rate, and the links have fixed capacities that can handle, without loss, any total sending rate under that capacity. Of course, the real world is far more dynamic and complicated, and our mechanisms will handle such settings, but we reduce the guiding principles to practice in static settings because they are much easier to reason about.

After developing this algorithm, the rest of the paper is structured as follows. In the following section we then ask whether the allocations produced by our algorithm achieve our goal of CCAI. *The answer to this question is the central focus and contribution of this paper, and our answer can be summarized as a "cautiously qualified yes".* We then discuss how our approach (i) might be implemented and deployed (Section 6) and (ii) how it compares to related work (Section 7), before finishing some concluding remarks (Section 8).

4.1 Principle #1: Relative Rights

To explore what relative rights mean, we consider three examples: a link (by which we mean a technology that has multiple ingresses and one egress), a switch (which has multiple ingresses and multiple egress), and an entire domain. 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 all three cases, we will assume that the relative rights are derived from the sender's access agreement, not the receiver's, but will generalize this when we discuss Principle #2.

To clarify our terminology, the term "user" refers to the entity entering into an access agreement (*i.e.,* an agreement that provides it some level of Internet service) at the network edge. The term "stream" refers 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, but for convenience we model it as a single CCA. Hereafter, for cases where we have multiple ingresses and egresses, as in switches and domains, we use the term $aggregate(i, \alpha)$ refers to the traffic entering at ingress i and leaving at egress α.
What does "Relative Rights" mean at a single link? Consider a single link with several streams sending at rates r_i. We denote the resulting 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. More 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 of relative rights 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 it gets what it asks for, 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). In our quest for CCAI, we want to allow all reasonable CCAs to utilize the available bandwidth. There are a wide variety of congestion signals that CCAs might use, such as increasing delays (*e.g.,* TCP Vegas), onset of drops (*e.g.,* TCP Cubic), or resulting throughput (*e.g.,* BBR). CCAI requires that all of these signals lead to roughly the same operating point. We should thus think about CCAI as having two different qualitative dimensions: how aggressive is a CCA, and what congestion signals does it use. Then, in addition to allocating bandwidth according to relative rights (which limits the impact of being more aggressive through pipe-like behavior), streams should only experience substantial increases in delay at or near the same point they start experiencing drops. This requirement will become relevant when implementing our bandwidth allocations (Section 6).

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, 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 larger than the link capacity).

What does "Relative Rights" mean 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 w_i for each ingress i applied at all egresses α. One could generalize this in two ways. First, the weights could be static but depend on each egress: *i.e.,* the $aggregate(i, \alpha)$ from ingress i to egress α has a weight w_i^α. To keep things simple, we do not embrace this generalization in our treatment here, but our results apply to this case as well.

Second, 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 there are two ingresses – i and j – sending traffic to two egresses α, β. Recall that CCAs send at the maximal rate where they do not experience loss. 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 do not consider it here.

What does "Relative Rights" mean at a domain? When considering this question, 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 and access agreements between domains. If there is no internal congestion, then a domain looks like a switch, with its multiple ingresses and multiple egresses. However, if the domain does suffer internal congestion, the relative rights (as defined by the weights w_i assigned upon egress) should be enforced on all internal links or switches where congestion occurs. We think 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.

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,[3] to first order the relative rights

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

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. 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 in (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 to determine which packets are dropped. This process recurses, building up a hierarchy of access rights for each stream. As in [1], we call these *recursive congestion shares* and refer to the general approach as RCS.

What does "Follow the Money" mean? So far we have been assuming, for simplicity, that these access agreements assign relative rights to the *sending* of packets. We now examine this assumption more carefully with an eye towards how money flows through the network. We start with the general observation that payments typically flow from end users to domains at network ingresses and egresses. Thus, consistent with Principle #3, at an egress to a user, the weights for the various aggregates leaving the network via that egress are set by the agreement that user has with that domain. 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.

Similarly, as argued before, when an end user's traffic enters a domain, the relative rights (expressed as a weight) for that traffic when leaving that domain should be set by the agreement between the sending user and its first-hop domain. This rule should apply *except* when it conflicts with the first rule (*i.e.,* when the packet enters the domain from a user, and exits the domain to a user, the weight at egress is set by the receiving user). The reason for this exception is that (i) we have two conflicting rules, and we must choose between them, (ii) in general, we want to give users control over what traffic they receive, and (iii) we want to avoid zero-rating, where companies pay to have their own traffic delivered to users, to the exclusion of other traffic.

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 [23], 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 described earlier.

To make our notion of "follow the money" excrutiatingly concrete, consider a packet following a path between domains where, for convenience, we assign the domains to two tiers and assume domains on the lower tier are customers of domains on higher tier, and domains in the top tier peer freely. In addition, we assume that the senders and receivers of packets are connected to domains in the lower tier.

This is all depicted in Figure 1, where packets flow in the direction of the black arrows. Money flows strictly upward, from users (senders and receivers) to customer domains to provider domains. The colored dots represent scheduling on egress links.

Consider the path taken by packets from s_1 to r_1. On the way up, packets are carried by $2u_1$ from s_1 to $1u$ because of the customer access agreement the source s_1 has with that domain (with money going from s_1 to $2u_1$). At the egress from $2u_1$ to $1u$, HWFS is applied with the weights derived from the customer agreements between $2u_1$ and the various s_i. Packets are carried by $1u$ from $2u_1$ to $1d$ because of the customer interdomain agreements between $2u_1$ and $1u$ (with the money going from $2u_1$ to $1u$). At the egress of $1u$ to $1d$, HWFS is applied with the weights derived from the customer agreements between $1u$ and the various $2u_i$.

On the way down, packets are carried by $1d$ from $1u$ to $2d_1$ because of the interdomain agreements between $2d_1$ and $1d$ (with the money going from $2d_1$ to $1d$). At the egress from $1u$ to $2d_1$, HWFS is applied with the weights derived from the customer agreements between $2u_1$ and the various r_i. Packets are carried by $2u_1$ from $1d$ to r_1 because of the customer access agreement the receiver r_1 has with that domain (with money going from r_1 to $2d_1$).

The hierarchical set of weights on packets at the egress from $1u$ to $1d$ is depicted on the left. The first layer of allocation occurs between the aggregates from the $2u_i$. The next layer of scheduling occurs between the aggregates s_i within $2u_1$. The hierarchical set of weights on packets at the egress from $1d$ to $2d_1$ is depicted on the right. The first (and only) layer of allocation occurs between the aggregates r_i.

Note that in this case the packet travels through two Tier 1 domains, with a peering link between them; some paths do not traverse a peering link (*e.g.,* when the sender and receiver share the same parent in their money flow); the example could be suitably modified to this case, with the weights in the single top tier domain being driven by the receiving customer domain (based on the tie-breaking rule).

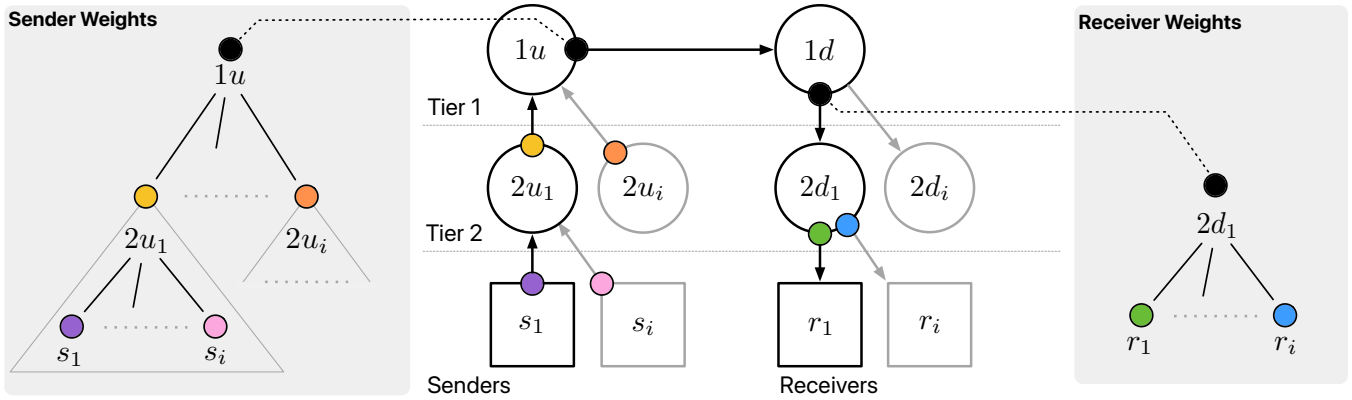

Figure 1: The center provides an example of how packets and money flow in RCS. Left and right show the tree of hierarchical weights on the sender and receiver side, respectively.

More generally, 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 i are considered part of the same aggregate with weight w_i throughout the domain they are entering. On the way down, 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 α are considered part of the same aggregate with weight w_α throughout the domain they are exiting. No weights are assigned based on the top-level peering arrangements.

What allocations does this produce? 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 the 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 the aggregates at that egress). Thus, all aggregates at a given 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 recurs all the way through the hierarchies. We will call this Hierarchical Weighted Fair Sharing (HWFS), which is identical to the static allocations achieved by the various hierarchical weighted fair queueing algorithms in the literature [24, 25].

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 [19] provides a mechanism whereby users can remotely control the internal composition of the stream; we provide more detail in Section 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 occurs on the way up, and the receiving user determines the composition if the congestion occurs on the way down.

There is a subtle distinction to make for traffic on the way down. The receiving user can assign weights to its various incoming aggregates, and the network will enforce bandwidth allocations to the aggregates according to those weights. However, *within* those aggregates, we need additional measures so that the receiver can control the composition of that aggregate (*i.e.,* the relative scheduling of the aggregate's flows), which is what we address in Section 6.

5. DOES RCS ACHIEVE CCAI?

5.1 Context

The RCS allocation scheme is derived directly from our three basic principles. While there may be other ways to reduce these principles to practice, we think they will differ only in details, not in the overall structure of having recursive congestion shares dictate the relative rights on congested links. The basic question we must address is: *Does RCS achieve CCAI?* If the answer is yes, then we have a conceptual answer to how we can achieve our goal of CCA independence in a way that is consistent with the Internet's commercial realities. If the answer is no, and that negative answer does not depend on any minor detail in our scheme but is due to the general approach, then we must conclude that achieving CCAI involves rejecting one of our principles, and thus would not be consistent with the Internet's current commercial arrangements.

As mentioned previously, CCAI involves two issues, the use of differing congestion signals (which we return later in this section) and differing degrees of aggressiveness, which is what we focus here. We thus ask, do more aggressive (but still reasonable) CCAs receive more bandwidth under RCS?

You might think that the answer to this question is trivial. We know that a network of weighted-fair-queueing (WFQ) routers [12, 26] does not give more bandwidth to more aggressive CCAs, but instead provides isolation between CCAs. RCS involves a hierarchical version of fair sharing, so why doesn't CCA independence follow immediately for RCS? The reason is that the weights get assigned differently at each domain in the network (based on the local agreements), so the clean WFQ results no longer hold (and we later provide an example of this in Figure 5 in Appendix A). There are other differences between RCS and per-flow-fairness, such as (i) the hierarchical nature of weights as agreements recurse, (ii) the weights being per-aggregate instead of per-flow, and (iii) the weights being determined by ingress or egress. All of these contribute to making RCS's behavior much harder to understand, but we need only the fact that the weights change arbitrarily from egress to egress to undermine the CCAI properties of WFQ. In fact, in our exploration, we did not find any anomalies (described below) in RCS behavior that we could not reproduce only using what we call a non-hierarchical and non-aggregated model of the network, which we refer to as NHM.

We should make clear that in what follows we are focusing on the incentive aspects of RCS (*i.e.,* does a more aggressive CCA get more bandwidth?), not on the equity aspects on RCS (*i.e.,* do the allocations reflect the underlying economic relationships?). The hierarchical and aggregated nature of RCS is designed to satisfy the latter but, at least in our experience, does not impact the former. Thus, we focused our attention on this simplified NHM model of RCS for our analysis.

5.2 Network Model and Equilibria

We now analyze RCS's bandwidth allocations in a theoretical NHM network model, and compare them to what would happen with FIFO at egresses. In this model, we consider a set of streams in the network, each controlled by their own CCA. We keep only the first level in the weight hierarchy, but allow these first-level weights to be set arbitrarily. These CCAs maximize their resulting throughput under the constraint of not incurring persistent loss (*i.e.,* the utility function is their throughput rate as long as they have no loss, and the utility function is negative infinity if they do incur losses). The network is a set of interconnected domains that only experience congestion at their egress links, so the domains effectively operate as switches.

The interactions of CCAs in the network can be seen as a game between self-optimizing users where "strategies" are the rates at which each user sends. The standard solution concept in such games – *i.e.,* the assumption about which set of strategies the players eventually converge to – is the Nash equilibrium, where no user can make a unilateral change (*i.e.,* holding all other sending rates fixed) to improve their throughput without incurring persistent losses.

However, there is another form of equilibrium that we must consider. A Stackelberg equilibrium, in our setting, is where one player (the leader) fixes its sending rate, and then the other players adapt to a Nash equilibria in the subgame with the leader's sending rate fixed. The Stackelberg leader can do at least as well in Stackelberg equilibria as they can in Nash, so there is an incentive to take the lead.

To make this more precise, consider a function $\vec{F}(\vec{r})$ where r_i is the sending rate of stream i and $F_i(\vec{r})$ is the throughput rate of the i'th stream (*i.e.,* the rate at which its packets arrive at the user after passing through the final egress). At Nash equilibrium we must have, for all i, $r_i = F_i(\vec{r})$ and r_i is the largest value (given fixed values for all r_j) where this is true. Let $A_i(r_i)$ be the value of F_i in the Nash equilibrium in the subgame defined by fixing r_i and letting all r_j's vary (if there are multiple such Nash equilibria, choose the one that maximizes F_i). Define S_i to be the largest value of r_i such that $A_i(r_i) = r_i$. This represents the maximal throughput without loss stream i can achieve by becoming the Stackelberg leader,

and thus represents the best outcome an aggressive CCA can achieve for stream i.

For FIFO on this network model, the results are simple and self-evident:

THEOREM 5.1. *For FIFO on a single link, the set of Nash equilibria is the set of $\vec{r}$ with $\sum r_i = C$, and the only Stackelberg equilibrium with i leading is $r_i = C$ and all other $r_j = 0$.*

This shows that there is no CCA independence for FIFO, and that aggressiveness pays to an extreme degree. The situation with RCS is far more complicated, as we now explore through calculations and emulation. We explain the basic results here, but please see Appendix A for more details.

5.3 Results for NHM

To understand the equilibria of the NHM model, we introduce a modified model (MM) that we analyze numerically. MM slightly alters the way the throughput is calculated to avoid various discontinuities in best-replies in the full-game $\vec{F}$. In MM the flow rates entering at each link are the same as what enters the network, and the output is the minimum over what emerges from each link. To express our results, we need a few definitions. A scenario is a set of users, domains, interconnecting links with given bandwidths, and traffic streams with given weight assignments. A non-degenerate scenario is one in which, if you randomly vary the various link capacities slightly, the set of equilibria in MM remain the same. As we show in Appendix A, almost all (in a measure-theoretic sense) scenarios in MM are non-degenerate. Thus we can focus only on non-degenerate scenarios because those are indicative of the behaviors you will find in real networks (as opposed to special degenerate cases that we can artificially create). An equilibrium is robust if it continues to exist (albeit slightly perturbed numerically) under all small perturbations of link capacities.

Based on our understanding of both models (see Appendix A), NHM and MM, we have the following conjecture (in which the set of equilibria includes both Nash and Stackelberg) which suggests that the simplified model MM allows us to bound the set of equilibria in NHM.

CONJECTURE 5.2. *The set of equilibria in any NHM is nonempty and is a subset of the equilibria in MM. For non-degenerate scenarios the set of equilibria in NHM is finite and robust.*

Assuming this conjecture holds, if a scenario has a single Nash equilibrium in MM then it has a single equilibrium in NHM. *In such cases, there is an unambiguous outcome of the game. No matter how aggressive the various reasonable CCAs are, there is only one equilibrium outcome.* This is the epitome of CCAI.

However, we know there are cases where NHM has multiple Nash equilibria (see Figure 5 in Appendix A), but the question is: how common are they in practice? We analyzed the MM model using a mixed integer program. We applied these numerical calculations to a set of scenarios where the underlying topologies were derived from a CAIDA [27] dataset that contains AS relationships inferred using the techniques described in [28]. We then randomly chose the locations for the sources and destinations for 100 streams, and used a publicly available route simulator [29] to handle the routing between them on this topology. For the interdomain links, we randomly chose bandwidths of the form $10B$ for integers between 1 and 20. To assign weights in our topology we randomly chose integer values between 1 and 100.

We found that out of 6042 scenarios, 172 (2.84%) had multiple Nash, but in many of those multiple Nash only a few streams had differing outcomes. Looking at individual streams, out of a total of 604200 streams in all scenarios, only 3696 (0.61%) had cases where being more aggressive would have paid off, and the average percent gain was 4.93%.

To confirm that these results were not merely artifacts of our particular topology modeling, we then considered a completely random topology generator that starts with a set of 10 nodes fully connected by links, and then generated 40 streams of length 4 by merely picking the sequence of nodes randomly. We found that out of 111862 scenarios, 1302 (1.16%) had multiple Nash. Out of a total of 4474480 streams in all scenarios, only 14118 (0.32%) had cases where being more aggressive would have paid off, and the average percent gain was 8.59%. To confirm that things did not get worse with longer flows, we repeated the experiment with streams of length 6, and found that out of 48618 scenarios, 251 (0.52%) had multiple Nash. Out of a total of 1944720 streams in all scenarios, only 5138 (0.26%) had cases where being more aggressive would have paid off, and the average percent gain was 6.69%.

Thus, for a random stream, the average expected gain (*i.e.,* the probability of having a better Nash times the possible gain) from being aggressive in our various experiments was less than 0.03%. Moreover, to gain this advantage, a CCA wouldn't just need to be aggressive, but would also need to be both clairvoyant and persistent. Because CCAs have no idea of the network topology nor the set of streams on the network, it cannot calculate the where the preferable equilibrium would be, so to force the system into that equilibrium would require (i) guessing what value of rate to send at in order to induce the new equilibrium and (ii) sending at that rate long enough to let all the other CCAs adapt which, if the CCA guessed wrong, would lead to persistent losses.

These results strongly suggest that we have achieved a reasonable degree of CCA independence. However, it is impossible to validate how RCS would operate in the current

Internet: accurate or even suggestive data about the distribution, duration, and dynamics of flows is not available to us; we would have to guess about how weights would be set in an RCS-based Internet; and simulations at that scale would be infeasible even if we had the requisite data. The two questions we would want these simulations to answer are: (i) *Would the system of CCAs converge to an equilibrium?* (so our results about equilibria are relevant) and (ii) *Would our results about their typically only being one natural equilibrium hold?* (so that there is no incentive to be more aggressive). For the first, we know from previous work [30] that convergence properties with fair-queueing-like scheduling is strictly better than with FIFO, so there seems little reason to worry about RCS destabilizing the Internet's congestion control dynamics. In addition, from close inspection of examples, it seems clear that the cases where there were multiple equilibria all involved very specific interactions between a few streams that would be much harder to realize if there were many more streams on each egress. Moreover, the rarity of multiple Nash seem completely unconnected with the simplifying assumptions in NHM, since these specific forms of interaction are made even rarer with aggregation and hierarchical weights. The reason Conjecture 5.2 is not a theorem is because we have not yet been able to prove that these specific arrangements, which we now understand, are the *only* problematic configurations. Lastly, the chance that a single stream could lead the network into a significantly different equilibrium becomes smaller as there are more streams in the network.

5.4 Packet Emulation

Thus far, we have discussed the possible outcomes for hypothetical CCAs in a simplified theoretical network. We now use packet emulations to explore RCS's effects on real CCAs. We emulate a topology, weights, and a traffic matrix, and run persistently backlogged data transfers among nodes in the topology. In this setting, we can calculate each flow's "HWFS rate," *i.e.,* the steady-state rate it should achieve given the weights of the domains its path passes through. Consider the topology in Figure 2. We set this topology's bottleneck link to be at the egress of domain B. Given the weights the topology uses, we expect a flow from domain S1 to domain D1 to achieve twice the throughput of one from domain S2 to D1. Similarly, the cumulative throughput of these two flows should be twice the throughput of a flow from S3 to D1.

How well do various bandwidth allocation mechanisms perform in assigning bandwidth to these flows? We consider three schemes: FIFO (*i.e.,* the status quo), fair queueing (we use DRR [31]), and HWFS (configured with the topology's weights). We ran 100 60 second `iperf` transfers, randomly assigning each flow (choosing with replacement) from a selection of CCAs: NewReno[32], Cubic[4], Vegas[6], and BBR[9]. In Figure 2, we find that when the egresses use: (1) FIFO, throughput depends on the sender's CCA, the CCA of the competing flows, and variations in starting time; (2) DRR, senders get CCAI, but their throughputs are not consistent with their access agreements; and (3) HWFS, senders get CCAI *and* achieve throughput consistent with the economics.

We also considered a larger topology (Figure 2 right) with 8 domains, 4 senders, 4 receivers, depth-2 sender and receiver weight trees, and weights randomly selected for each ingress in the range [1,10]. We ran 10 experiments with randomly chosen CCAs for each sender (each sender used each CCA at least once). The bottleneck link rate was 80 Mbit/s. We calculated the standard deviation among the iterations for each sender-receiver pair; a high standard deviation indicates that bandwidth allocations were CCA-dependent and a low one would indicate CCAI. While FIFO configuration had a mean standard deviation (across the senders) of 20.0 Mbit/s, HWFS had a mean standard deviation of 0.3 Mbit/s.

6. IMPLEMENTATION

We now describe a set of mechanisms implementing RCS. We start this discussion by assuming bilateral interconnects and no intradomain congestion, and then later address the cases of IXP-mediated interconnects and internal congestion. There are three categories of mechanism needed for RCS: packet scheduling at the egresses to implement HWFS, signalling within and between domains to ensure that these egresses have the appropriate set of weights, and additional mechanisms to handle endpoint control.

Implementing HWFS: We must achieve HWFS at every egress link. Appendix B contains the pseudocode for our implementation. Our approach is based on the Deficit Round Robin (DRR) [31] implementation of fair queuing, but with hierarchical extensions. Consider an egress α which is on the way up. We first schedule within each $aggregate(i, \alpha)$ for each ingress i. Within that aggregate, we then schedule with the children of that aggregate, and so on. While the scheduling complexity of normal DRR is $O(1)$, as it simply tries to transmit a packet from the next flow in DRR's linked list, here the scheduling complexity is $O(d)$, where d is the depth of the hierarchy (which is typically no more than 3 [33] for most paths taken). Today's routers already support hierarchical scheduling with a depth of 3,[4] so we believe that there is sufficient computing headroom for our HWFS implementation.

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.

[4]The hierarchical scheduling in many current routers implements different algorithms at each level; *e.g.,* fairness through DRR, rate limiting, and timed releases through calendar queues, as opposed to providing hierarchical DRR.

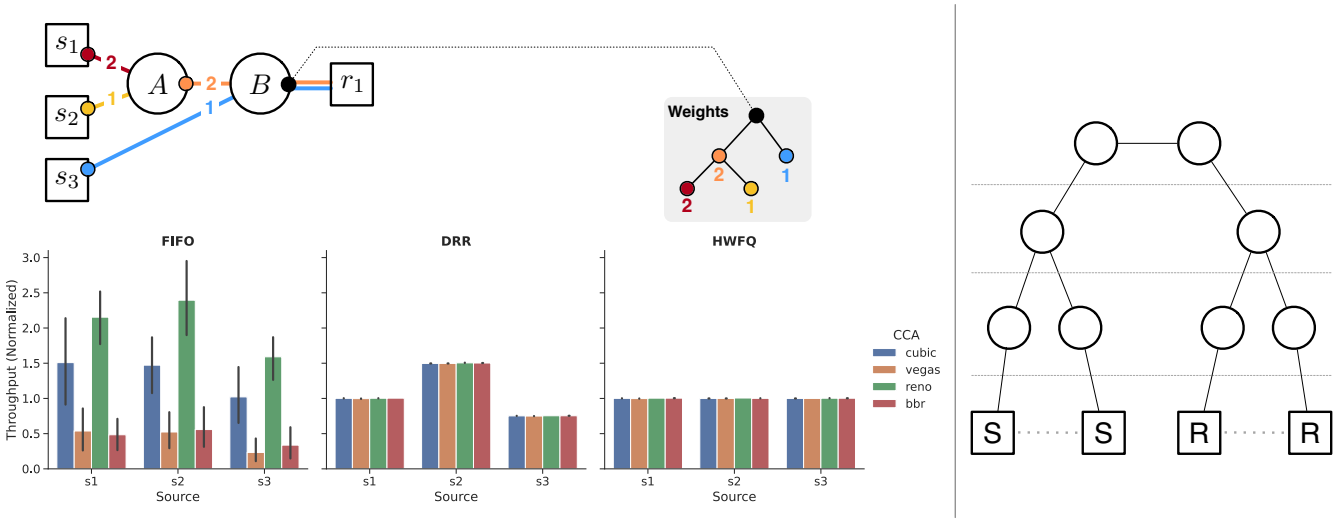

Figure 2: (a) Simple topology for packet emulation experiments. Squares represent hosts, circles represent domains. The numbers next to each link indicate their relative weights. The box shows the tree of HWFS weights used at the egress $B \rightarrow r_1$ which is the only congested link. (b) The y-axis plots throughput by a when using a given CCA, normalized to each host's share of the bandwidth, as dictated by their weights. (c) Larger topology.

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 an aggregate's total bandwidth allocation, that aggregate's endpoints should determine which individual flows use that bandwidth. There is an implementation challenge in supporting more complex flow allocation policies: the aggregate'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 approach described in Bundler [19] which shifts 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 aggregate (bundle in the terminology of [19]) 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.

Prospects for deployment: The mechanisms described above are entirely feasible. While there are likely no commercial routers that implement HWFS, the complexity of HWFS is no more than what such routers currently implement for traffic shaping and the like, and is easily within range of today's hardware capabilities. The signalling mechanisms

could probably be piggy-backed on current protocols, but even a from-scratch implementation would not be difficult.

Thus, while domains certainly *could* deploy RCS (once such routers became available), the question is *will they?* One could imagine domains starting to offer congestion shares as part of their service offering, providing an assurance about how a customer's traffic would be treated under congestion. Then pairs of domains could decide to start exchanging such information, so that these agreements help further upstream. This process could continue, but this is hardly a compelling business case.

7. RELATED WORK

We first discuss the most directly relevant work, [1], 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 [1]? This paper was inspired by [1] where the basic idea of recursive congestion shares was introduced. However, the detailed approach in [1] has two major limitations. First, and most fundamentally, [1] 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, this is sufficient.[5] However, this is insufficient if one or more transit providers are traversed. For example, consider two customers of an ISP whose traffic must go through a transit domain before reaching their common destination. Then, if the congestion is at the egress of the transit domain to the destination, there is no way for the transit provider to distinguish between the streams belonging to the two customers, and the customer with the more aggressive CCA will get more bandwidth. How common are such cases? Even in the most favorable study cited in [1], roughly 40% of traffic that left the originating domain traveling to a GCP server traveled through at least one transit domain (the numbers for other major destinations were higher).[6] For these cases, the more fully hierarchical form of congestion shares advocated here would produce different bandwidth allocations. The question is, do these differences matter? And this is where we think [1] is fundamentally mistaken.

From the point of view of the "fairness" of the bandwidth allocations, it probably makes little difference. In quantitative terms, the allocations produced by our hierarchical approach would, in most cases, only vary moderately from those in [1]. Besides, users would not know enough about the overall network setting to know if their allocations followed those in [1] or what we propose here. However, from the perspective of incentives, these differences may matter a great deal. This is because users will adopt a more aggressive CCA if it provides even moderate improvements in a significant number of cases. We think these improvements will indeed be significant with the flat version of RCS in [1], since such gains are possible in any situation where traffic is going through one or more transit hops.[7] With the fully hierarchical of RCS presented here, as we showed in the previous section, they would have little incentive to use more aggressive CCAs.

The second limitation is that the flat version of RCS ignores the role of receivers, only considering weights assigned by ingress points. This raises issues equity issues (*i.e.,* should any content provider be able to buy priority over other providers 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.

The two deficiencies – that flat version of RCS in [1] does not solve the incentive problems for CCAs and does not follow the path of money – means that we must look at the more complicated but more functional proposal presented here. However, when we look at the additional complications, the signaling protocols between the two approaches are essentially identical, and the only significant change lies in the scheduling algorithm, which is hardly a showstopper.

What about the alternatives? As mentioned previously, the literature has suggested two main alternatives to TCPF. The first, as initially articulated by [11] and further explored by many, is to provide 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 [34] where he observed that the resulting allocations made no economic sense. Flows, no matter how you define them (*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. However, here 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

[5] [1] 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).

[6] We have no definitive answer to the question of how much traffic actually travels over one or more transit domains on either the upwards or downwards paths. For instance, as noted in [1], some informal estimates suggest that roughly 70% of traffic stays either within the domain or the next hop. However, we think there is some nontrivial fraction – whether it is 10% or 20% or 30% doesn't matter – that travels over one or more transit domains.

[7] Note that there is a subtle distinction between the flat RCS model, which does not enforce CCAI and the NHM model which does; neither carries a hierarchical set of weights, but the flat RCS model aggregates streams that share the same ingress, while NHM preserves their separate weights.

goal is to maximize this utility. This approach was introduced by Kelly in [13, 14], and has generated a significant literature. The fundamental core of NUM is that congestion signals can serve as shadow prices, providing a measure of how much congestion a particular flow is causing others. 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 different ways. The most straightforward is to actually charge these shadow prices, 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 the 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 [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 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 on which we heavily rely; what follows is an illustrative but hardly exhaustive list of our indebtedness. For instance, the basic ideas about per-flow fairness were explored in [11, 12] and many other papers. Efficient and alternative implementations were explored in [15, 16, 31, 36–41], and hierarchical versions of this were explored in [24, 25, 42], More recently, [10] was a brilliant discussion of TCP friendliness and its discontents, and [43] provided a novel perspective on CCA diversity.

8. DISCUSSION

One might dismiss this paper as being *unnecessary* (the Internet of today works reasonably well), *premature* (the technical results depend on a conjecture that is not yet settled), *untested* (its proposed mechanism has not yet been validated at scale), *impractical* (the necessary 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 [11], 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. The problematic behavior we have identified in small examples seems very unlikely to create problems in the Internet at large. Moreover, the mechanisms proposed here, while not simple, are hardly complicated compared to the extensive traffic engineering and other measures that ISPs regularly deploy. The degree of coordination between domains, while new to congestion control, does not does not involve a tussle where their interests may differ; it is only domains reciprocally agreeing that when they have to drop another domain's traffic, they respect the weights assigned by that domain. As for scalability, we think HWFS could easily be implemented at the speeds necessary for interdomain links. Thus, there are no fundamentally insurmountable barriers to adopting a design resembling our approach, and we expect that further analysis and engineering could produce far better implementations than we discuss here.

However, we think our primary contribution lies not in the details of our design but in the first conceptual reconciliation of two ecosystem imperatives – CCA independence and Internet economics – that have long been at odds. Of course, there is much more to be done – theoretically and practically – but we hope to have provided the basis for further progress.

References

[1] 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.

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

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

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

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

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

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

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

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

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

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

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

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

[14] 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.

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

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

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

[18] Phillipa Gill, Michael Schapira, and Sharon Goldberg. A Survey of Interdomain Routing Policies. SIGCOMM, 2014.

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

[20] 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.

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

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

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

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

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

[26] Scott Shenker. Making Greed Work in Networks: A Game-Theoretic Analysis of Switch Service Disciplines. SIGCOMM, 1994.

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

[28] Matthew Luckie, Bradley Huffaker, Amogh Dhamdhere, Vasileios Giotsas, and kc claffy. AS Relationships, Customer Cones, and Validation. IMC, 2013.

[29] Italo Cunha. Bgpsim. 2022. URL https://github.com/TopologyMapping/bgpsim.

[30] Scott Shenker. A Theoretical Analysis of Feedback Flow Control. SIGCOMM, 1990.

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

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

[33] 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.

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

[35] 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.

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

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

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

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

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

[41] Hongyuan Shi and H. Sethu. Greedy Fair Queueing: A Goal-oriented Strategy for Fair Real-time Packet Scheduling. IEEE Real-Time Sys. Symp., 2003.

[42] Zhuolong Yu, Jingfeng Wu, Vladimir Braverman, Ion Stoica, and Xin Jin. Twenty Years After: Hierarchical Core-Stateless Fair Queueing. NSDI, 2021.

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

APPENDIX

A. THEORY

In the following we will consider four theoretical different models. There is the real model (RM) which is the model discussed in the main text with arbitrary recursive hierarchical weights and stream aggregation, and which was emulated in Section 5.4. In this section we begin our treatment by focusing on the non-hierarchical model (NHM) introduced in the body of the paper which removes the hierarchical weights. Both of these models pose a technical problem (as we explain below, this is a lack of continuity in the best-reply function), so we consider modified versions that avoid this issue. We call these the modified model (MM), which is based on NHM, and the hierarchical modified model (HMM) that is based on the real model. These modified models do not capture some important aspects of reality but we conjecture they bound what goes on in the more complicated models.

A.1 Notation

In NHM, the weights assigned to streams at an egress link are arbitrary but non-hierarchical (each stream has a single weight, not a hierarchy of weights) and non-aggregated (each stream has its own weight; they are not combined into aggregates that share a weight). We discuss a generalization to the hierarchical and aggregated case RM in Section A.7. Let N be the set of links and M the set of streams in a network (where we treat domains as switches, assuming no internal congestion). Each link j has a capacity c_j and each stream i follows a path p_i which is a sequence of links from source to destination. At every link there a set of weights for all streams that pass through that link, w_i^j; these are chosen arbitrarily. Each link allocates bandwidth according to HWFS (which, because the NHM has no hierarchy, just uses the definition of relative rights defined in the body of the paper). We define $\vec{F}(\vec{r})$ to be the throughput attained when the sending rates are $\vec{r}$. We assume that the utility of streams are r_i if $r_i = F_i(\vec{r})$ and $-\infty$ if $r_i > F_i(\vec{r})$. We say that $\vec{r}$ is non-wasteful if $\vec{r} = \vec{F}(\vec{r})$. Define the fairness index for a stream i at link j as $\vec{r}_i/w_i^j$.

A.2 Necessary Conditions for Nash Equilibria in the NHM

Now consider some Nash equilibrium $\vec{r}$. In any Nash equilibrium there are no persistent losses so the arrival rate throughput for any stream must be the same at all links on its path and the total sending rate from all streams at any link is less than or equal to its capacity. The key idea in our analysis is the bottleneck.

Definition A.1. Assume that $r_i = F_i(\vec{r})$. Then stream i is bottlenecked at link j for $\vec{r}$ if an increase in r_i would not increase $F_i(\vec{r})$ and j is the earliest link in p_i that prevents such an increase.

For example, consider a stream that only uses two links and there are no other streams. It goes through link 1 then link 2, both with capacity 1. If $r_i = 1$ then link 1 is the bottleneck link.

We then have the following result:

Lemma A.2. *If $\vec{r}$ is a Nash equilibrium then every stream i must be bottlenecked at some link j for $\vec{r}$.*

Proof: If i is not bottlenecked at any link then it can increase r_i to increase its throughput $F_i(\vec{r})$. □

We say that a link is a bottleneck if any stream is bottlenecked at it.

Lemma A.3. *If a link is a bottleneck then the sum of the sending rates into that link is equal to its capacity.*

Proof: A stream cannot be bottlenecked if the link has excess capacity. □

Lemma A.4. *If there are two or more streams bottlenecked at link j then they all must have equal fairness indices.*

Proof: This follows directly from our definition of relative rights in Section 4. Otherwise, the one with the lowest index could increase its sending rate by a small amount to increase its throughput.□

Define a bottleneck mapping by b where $b(i) = j$ if link j is the bottleneck link for stream i.

Theorem A.5. *Let $\vec{r}$ be a Nash equilibrium. Then there exists a bottleneck mapping b that satisfies the following:*
E1) For any bottleneck link j: $\sum_{i|j\in p_i} r_i = c_j$.
E2') For any i, i' with $b(i) = b(i') = j$: $r_i/w_i^j = r_{i'}/w_{i'}^j$.
I1) For all links j : $\sum_{i|j\in p_i} r_i \le c_j$.
I2) For any j and i, i' with $b(i) = j$ and $j \in p_{i'}$: $r_i/w_i^j \ge r_{i'}/w_{i'}^j$.

Proof: If E1, E2', or I2 are violated then the bottlenecked stream can increase its throughput by increasing its sending rate. I1 is simply the capacity constraint that applies for all nonwasteful allocations.□

Note that for E2' we have many redundant constraints, since if there are k streams bottlenecked at a specific link there are $k(k-1)/2$ constraints; we can express this with only $k-1$ constraints, by setting one specific bottlenecked stream's fairness index equal to each of the other bottlenecked streams' fairness indices. Define E2 to be this smaller set of equality constraints. These equality constraints can be used to develop an efficient integer program to find candidate Nash equilibria as discussed in Section A.8.

To simplify the discussion we define:

Definition A.6. Let BB be the set of $\vec{r}$'s which satisfy E1, E2, I1, and I2

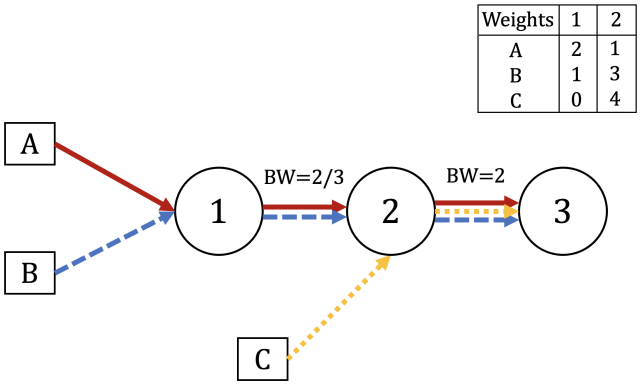

Figure 3: Example with no Nash equilibria but a Stackelberg equilibria.

However, even when BB is nonempty there need not be a Nash equilibria in the NHM, as is shown in the example in Figure 3. One can check all the bottleneck mappings to show that the only solution in BB (*i.e.,* that satisfies the necessary conditions for Nash equilibrium) is $(1/3, 1/3, 4/3)$, 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 is a Stackelberg equilibrium of the NHM 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 NHM scenario that has a nonempty BB and no Nash equilibrium, but has a Stackelberg equilibrium.

Thus we see that:

THEOREM A.7. *There exist $\vec{r} \in BB$ which are not Nash equilibrium in the NHM.*

A.3 The Modified Model (MM)

We now define the modified model (MM) where each stream has the same sending rate at each link (that is, we run HWFS on $\vec{r}$ at every switch, and then take the minimum output for each stream as the throughput $\vec{F}(\vec{r})$). As we show below, this model is much more amenable to analysis and conjecture that it bounds the results for the NHM.

THEOREM A.8.
1) Any $\vec{r} \in BB$ is a Nash equilibrium in the MM.
2) Any Nash equilibrium of the NHM is a Nash equilibrium of the MM.

Proof: The first result follows from the fact that at a stream's bottleneck, if it increases its rate then it cannot improve its throughput. This does not hold in the NHM model because increasing a stream's rate could change another stream's input into the bottleneck, but in the MM model each stream's input into each switch is the same. The second result follows from the fact that any Nash equilibrium of NHM is in BB (Theorem A.5) and the first result.□

A.4 Existence of Nash equilibria in the MM

First we notice that in the example in Figure 3 the optimal response for stream i to other r_j may be discontinuous and this underlies the lack of a Nash equilibrium for the NHM in that example. Using the notation where $(x, \vec{r}_{-i})$ denotes the vector with the i'th element defined by the first argument, and the other elements defined by the second argument (ignoring the i'th element of the second argument), define (for either NHM or MM, as specified) $BR(\vec{r})$ by

$$BR_i(\vec{r}) = \arg\max_x \vec{F}_i(x, \vec{r}_{-i}).$$

Now, in the example in Figure 3 let $Z = (1/3, 1/3, 4/3)$ be the unique solution in BB and note that this is a Nash equilibrium in the MM. A direct computation shows that in the NHM $BR_A(z_{-A}) = 1/2$, however, for small enough $\epsilon > 0$ define $Z' = (1/3, 1/3 - \epsilon, 4/3 + \epsilon)$ then in the NHM $BR_A(z_{-A}) = 1/3$ since in this case A will always suffer losses if it increases its rate unilaterally. Thus we see that the best reply function in NHM can be discontinuous. However, in the MM the best reply function is continuous which allows us to prove the existence of Nash equilibrium in MM.

THEOREM A.9. *The always exists at least one Nash equilibrium in the MM.*

Proof: We follow Nash's original proof on the existence of Nash equilibria. The best reply is clearly unique. To show that it is continuous, note that if $\vec{(r)}_{-i}$ changes by at most ϵ on each element then $BR_i(\vec{r}_{-i})$ changes by at most $|N|\epsilon$ (because, this is the most bandwidth it could gain, or the most it needs to reduce its sending rate to avoid losses). Let R be the set of $\vec{r}$ such that $\vec{r} = \vec{F}(\vec{r})$ and note that R is convex. A Nash equilibrium is a fixed point of $BR(\vec{r})$ and in the MM such a fixed point must exist from Brouwer's fixed point theorem since BR is a continuous function on a convex set.□

A.5 Non-degenerate Scenarios and the Robustness of Nash equilibria

We now consider the robustness of Nash equilibria in MM. In what follows, a perturbation of a scenario involves changing the link capacities, and not the connectivity or weights.

Definition A.10. An MM scenario is non-degenerate if the set of Nash equilibria is finite and robust. A scenario is degenerate if it is not non-degenerate.

Thus, under small perturbations of a non-degenerate scenario no Nash equilibria are created or destroyed and the locations of the existing Nash equilibria are continuous in the perturbations; *i.e.,* they only move a small amount for small enough perturbations.

The following applies to both the MM and the NHM.

Lemma A.11. *Given a bottleneck mapping $b(i)$, any $\epsilon > 0$, and a uniform random variable ξ on $(-\epsilon, \epsilon)^m$, where m is the number of bottleneck links, there will be a unique solution of E1,E2 for the perturbed capacities, $c + \xi$ with probability 1.*

Proof: Consider a Nash equilibrium $\vec{r}$ and a bottleneck mapping for it $b(i)$. For any link j with k streams bottlenecked at it there will be one equality constraint by E1 and $k-1$ by E2, so the number of equality constraints is equal to the number of bottleneck streams at this link. Thus the total number of equality constraints is equal to the total number of streams. (Note that the inequality constraints don't change this as they can only show that the solutions of these equality constraints, if they exist, are not feasible.) Thus we can write this as a block matrix equation $M\vec{r} = (\vec{c}, \vec{0})^t$, where the t represents the transpose, and $\vec{c}$ contains the capacities of all bottleneck links from E1 while the $\vec{0}$'s come from E2. If A is invertible there is a unique solution.

If A is not invertible, then there must be a linear relation between two or more rows of A and one of these rows must come from E1, since the rows of E2 can be seen to be linearly independent. Thus, for a solution to exist, this linear relation must be satisfied by $(c, \overline{0})^\dagger$. Now choose any c_j in that linear relation and note that if we change its value by any amount there will no longer be any solutions to the matrix equation. Thus, in the case that there could be multiple solutions, any perturbation of that specific c_j will lead to no solutions as it will break the linear equality among all the capacities. □

This lemma allows us to show that degenerate scenarios are rare.

Theorem A.12. *Given an MM scenario with arbitrary weights and paths, assume that the capacities are chosen from a continuous probability distribution on an open set of $\mathfrak{R}_+^m$ Then with probability 1, the scenario will be non-degenerate.*

A.6 Stackelberg Equilibria

Theorem A.13. *Any Stackelberg equilibrium in the MM is also a Nash equilibrium in the MM.*

Proof: Using the definitions introduced in the body of the paper, consider a Stackelberg equilibrium S_i which is not a Nash equilibrium. Recall that a Stackelberg equilibrium for i occurs at the largest value of r_i such that $A_i(r_i) = r_i$. Note that stream i can not be bottlenecked by any link otherwise it would be a Nash equilibrium. Then there must exist some ϵ such that increasing r_i by ϵ without changing any of the $\vec{r}_{-i}$ would be feasible. In this case, the Stackleberg leader i could also increase r_i by $\delta = \epsilon/n^2$. This would still satisfy $r_i = \vec{F}_i(\vec{r})$ since the best replies are continuous (as seen in the proof of Theorem A.9) and they would increase by at most $n\delta = \epsilon/n$ each so the total increase would be at most $n\epsilon/n = \epsilon$. Contradiction.

Thus, any Stackelberg equilibrium must satisfy the linear equalities $E1, E2$ and inequalities $I1, I2$ so will be a Nash equilibrium of the MM. □

After significant exploration of the NHM model both numerical and analytical, noting the the discontinuities in the best-reply can turn what are Nash equilibria in the MM into Stackelberg in the NHM, we conjecture the following:

Conjecture A.14.
1) If $\vec{r}$ is a Nash equilibrium in the MM then it is either a Stackelberg equilibrium or a Nash equilibrium in the NHM.
2) If $\vec{r}$ is a Stackelberg equilibrium in the NHM then it is a Nash equilibrium in the MM.

This forms the basis for Conjecture 5.2 in the body of the paper.

A.7 Extension to the more realistic model

The import of the calculations described in the body of the paper rely on the conjectured mapping between the NHM and the MM. Here, however, we want to observe that many of our results also apply to the real model (RM) and its modified counterpart HMM. As articulated in the body of the paper, we think the incentive properties of RCS depend mostly on the properties of NHM, but here we want to emphasize that the basic theoretical results on which our observations rely also apply to the more complicated models.

Recall that the realistic model includes the recursive hierarchical weighting schemes considered in the main text. It allows for arbitrary recursive grouping of streams at a link and arbitrary weights both within and between groups.

For example at some link j there might be three streams, where streams 1 and 2 are combined into a group stream and stream 3 is on its own. There is a weight for each group $(1,2), (3)$ which is used at the top level to determine the allocation between $r_1 + r_2$ and r_3. Then if the stream of $(1,2)$ is given less than requested, the allocation between 1 and 2 is decided recursively with relative weights between them.

Although the notation for this is quite cumbersome, the previous analysis holds with straightforward modifications to the proofs by redefining the MM to capture the recursive hierarchical structure. As before we do this by modifying the

RM to define the hierarchical modified model (HMM) where each stream has the same sending rate at each link.

Using the HMM, the proofs of Theorems A.8 and A.9 are essentially unchanged while the proofs of Theorems A.12 and A.13 follow from the observation that there are still n constraints in the hierarchical versions of equality constraints E1 and E2.

For example there will still be 3 equality constraints for the 3 streams if they are all bottlenecked at link j, the total capacity constraint on the link, a second constraint between the two group streams and a third constraint for the in group allocation of $(1, 2)$, so the previous analysis in Theorems A.12 and A.13 still holds.

Thus we have the following results:

- Theorem: Any Nash equilibrium of the RM is a Nash equilibrium of the HMM. (counterpart to Theorem A.8)
- Theorem: There always exists at least one Nash equilibrium in the HMM. (counterpart to Theorem A.9)
- Theorem: Any Stackelberg equilibrium in the HMM is also a Nash equilibrium in the HMM. (counterpart to Theorem A.13)
- Conjecture: If $\vec{r}$ is a Nash equilibrium in the HMM then it is either a Stackelberg equilibrium or a Nash equilibrium in the RM. (counterpart to Conjecture A.14)
- Conjecture: If $\vec{r}$ is a Stackelberg equilibrium in the RM then it is a Nash equilibrium in the HMM. (counterpart to Conjecture A.14)
- Theorem: Almost all networks in HMM are non-degenerate. If a network in HMM is non-degenerate then the set of equilibria is finite and robust. (counterpart to Theorem A.12)

A.8 Finding candidate Nash equilibria via mixed integer programming.

Pseudocode 1: Nash Equilibria Mixed Integer Program

```
MIP(flows, paths, nodes, node_to_weight, capacities):
    solutions = []
    for obj_fn in [max, min]:
        for flow_to_optimize in flows:
            prob = LinearProgram(obj_fn)
            allocs = LinearProgramDicts(flows)
            prob.obj_fn(allocs[flow_to_optimize])
            for flow in flows:
                prob.constraint(|b(flow)|==1)
                prob.constraint(E2)
                prob.constraint(I2)
            for node in nodes:
                prob.constraint(E1)
                prob.constraint(I1)
            prob.solve()
            solutions.append(allocs)
    return solutions
```

Candidate Nash equilibria can be found by solving a mixed integer program (MIP). This is done by encoding the bottleneck mapping $b(i)$ with binary variables and combining these with the linear equations E1, E2 and linear inequalities I1, I2. For moderately sized networks this MIP can be solved using standard off the shelf software.

A.9 Problematic Examples

Here we make more explicit the simplicity of WFQ networks and contrast it with the complexity of the NHM. A WFQ network is merely a special case of the NHM model where a stream i has the same weight w_i at every egress link. The well-known result (so we do not prove it here) on WFQ is that:

Theorem A.15. *For every network scenario, a WFQ network has a unique Nash equilibrium which is also the only Stackelberg equilibrium.*

This follows from identifying the link α and stream i that minimizes $\frac{C_\alpha w_i}{N_\alpha}$ where C_α is the link capacity and N_α is the number of flows on that link. We then know that $r_i = \frac{C_\alpha w_i}{N_\alpha}$, and can recurse the process. This is the only equilibrium in the network, and it has extremely strong incentive properties [26].

In contrast, in NHM, not every Nash is a Stackelberg, there are scenarios with no Nash equilibria, and there are cases with multiple Nash and even a continuum of Nash. In particular, we have already seen in Figure 3 that in NHM we can have a Stackelberg equilibrium that is not a Nash equilibrium; in fact, the scenario has no Nash equilibrium.

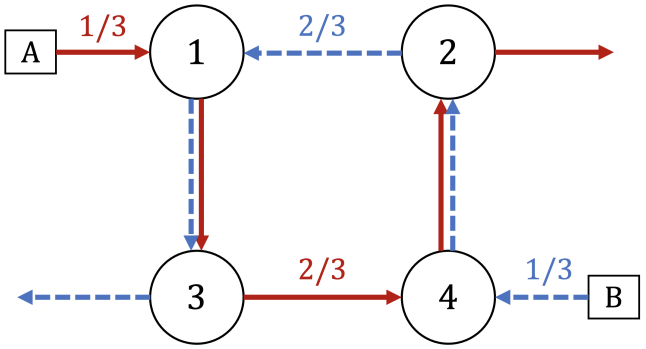

Figure 4: Example with a continuum of Nash equilibria $(\frac{1}{3}+x, \frac{2}{3}-x)$ where $x \in [0, \frac{1}{3}]$. All capacities are 1.

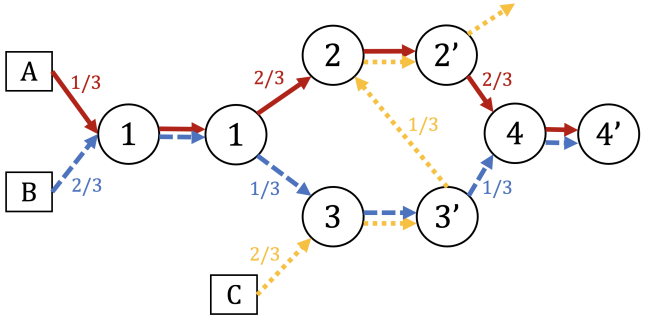

Figure 5: Example with 2 nash equilibria $(\frac{2}{3}, \frac{1}{3}, \frac{1}{3})$ and $(\frac{1}{2}, \frac{1}{2}, \frac{1}{2})$ with no intermediate equilibria. All capacities are 1.

Figure 4 depicts an example where NHM has a continuum of Nash. Here, all all allocations of the form $(\frac{1}{3}+x, \frac{2}{3}-x)$ with $x \in [0, \frac{1}{3}]$ are Nash. However, this is not a generic scenario, as perturbing the link capacities eliminates the continuum.

Finally, Figure 5 depicts an example where where NHM has two isolated 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. This is a robust scenario.

B. SCHEDULING ALGORITHM

Listing 2 shows the hierarchical weighted scheduling algorithm, which is adapted from Deficit Round Robin (DRR) [31]. Unlike DRR, which is constant-time, the algorithm's complexity scales with the depth of the 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 of its next child. In this case, our implementation will assign the child node any remaining deficit, but track the remaining unassigned scheduling quota for the next scheduling round.

The scheduling algorithm depends on the calculation of quanta. To calculate quanta amounts from weights, we first recursively calculate the minimum quantum required for the root node to divide one quantum among its children evenly. For leaf nodes, this is simply their weight. For non-leaf nodes, the minimum quantum is the sum of its child nodes' weights multiplied by the maximum of its child nodes' minimum quanta. This minimum quantum can then be divided at the root node (and recursively downward) according to the fraction of child nodes' weights. This formula ensures integer-valued quanta at every node. For large trees, the root node's quantum may need to be smaller than indicated by the above minimum quantum to avoid quanta that are too large and cause rate instability. In these cases, the effective weights will be approximate.

Pseudocode 2: Hierarchical deficit weighted round robin implementation.

```
Schedule(n) for root node:
loop:
  n.deficit += n.quantum
  for c in n.children:
    n.deficit -= c.quantum
    Schedule(n, c.quantum)

Schedule(n, credits) for non-leaf node:
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
      return
  if n.deficit == 0:
    c.leftover += c.quantum - q
    n.children.rotate() // move c to tail
    return

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(p)
  if n.queue is empty:
    n.deficit = 0 and set n as inactive
```