Beyond Jain's Fairness Index: Setting the Bar For The Deployment of Congestion Control Algorithms

Ranysha Ware
Carnegie Mellon University
rware@cs.cmu.edu

Matthew K. Mukerjee
Nefeli Networks
mukerjee@nefeli.io

Srinivasan Seshan
Carnegie Mellon University
srini@cs.cmu.edu

Justine Sherry
Carnegie Mellon University
sherry@cs.cmu.edu

ABSTRACT

The Internet community faces an explosion in new congestion control algorithms such as Copa, Sprout, PCC, and BBR. In this paper, we discuss considerations for deploying new algorithms on the Internet. While past efforts have focused on achieving 'fairness' or 'friendliness' between new algorithms and deployed algorithms, we instead advocate for an approach centered on quantifying and limiting harm caused by the new algorithm on the status quo. We argue that a harm-based approach is more practical, more future proof, and handles a wider range of quality metrics than traditional notions of fairness and friendliness.

ACM Reference Format:
Ranysha Ware, Matthew K. Mukerjee, Srinivasan Seshan, and Justine Sherry. 2019. Beyond Jain's Fairness Index: Setting the Bar For The Deployment of Congestion Control Algorithms. In *The 18th ACM Workshop on Hot Topics in Networks (HotNets '19), November 13–15, 2019, Princeton, NJ, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3365609.3365855

1 INTRODUCTION

In recent years, the networking research community has generated an explosion of new congestion control algorithms (CCAs) [1, 2, 5, 6, 25–27], many of which are being explored by Internet content providers [4, 19]. This state of affairs brings the community back to an age-old question: what criteria do we use to decide whether a new congestion control algorithm is acceptable to deploy on the Internet? Without a standard *deployment threshold*, we are left without foundation to argue whether a service provider's new algorithm is or is not overly-aggressive.

A deployment threshold concerns inter-CCA phenomena, not intra-CCA phenomena. Rather than analyzing the outcomes between a collection of flows, all using some CCA α, we need to analyze what happens when a new CCA α is deployed on a network with flows using some legacy CCA β. Is α's impact on the status quo is acceptable?

Our community has traditionally analyzed inter-CCA competition in two ways, which we refer to as 'fairness' and 'mimicry.' While both approaches are insightful, we argue that neither is a sound basis for a deployment threshold.

A throughput allocation is fair if it maximizes every users utility function given limited link capacity [21]. A end-host CCA, typically defines users as flows, aiming to maximize utility per-flow by ensuring that every flow sharing the same bottleneck link gets equal bandwidth. For example, CCA designers try to argue their CCA α is deployable if it is fair to Cubic (β), the default CCA in Linux [1, 3–6, 26]. However, a fairness-based deployment threshold suffers from three key issues:

(1) *Ideal-Driven Goalposting*: A fairness-based threshold asserts a new CCA α should equally share the bottelneck link with currently deployed CCA β. In practice, this goal is too idealistic to achieve in practice. The end result is that ideal-driven goalposts are simply ignored as impractically high requirements. For example, CCA designers have argued that it is acceptable to be unfair to Cubic because Cubic is not even fair to itself [1].

(2) *Throughput-Centricity*: A fairness-based threshold focuses on how a new CCA α impacts a competitor flow using CCA β by focusing on β's achieved throughput. However, this ignores other important figures of merit for good performance, such as latency, flow completion time, or loss rate.

(2) *Assumption of Balance*: Inter-CCA interactions often have some bias, but a fairness metric cannot tell whether the outcome is biased *for* or *against* the status quo. It makes a difference in terms a deployability whether a new CCA α *takes* a larger share of bandwidth than a legacy CCA β or *leaves* a larger share for β to consume: the former might elicit complaints from legacy users of β, where the latter would not. Jain's Fairness Index [12] assigns an equivalent score to both scenarios.

Mimicry is a separate approach where new algorithms replicate properties of TCP-Reno (*e.g.*, driving throughput as a function of the loss rate [18]) in order to be 'friendly'

ACM ISBN 978-1-4503-7020-2/19/1...$15.00
https://doi.org/10.1145/3365609.3365855

to legacy, Reno-derived TCPs. The issue with mimicry is that it binds new algorithms to repeating the often undesirable idiosyncrasies of Reno, stifling improvement and evolution [13]. We discuss the drawbacks of fairness and mimicry as the basis of a deployment threshold further in §2.

We advocate instead for a deployment threshold based on *harm*. Harm allows us to speak in quantifiable, measurable terms about the impact of deploying a new CCA to the Internet. One can use measurements or models to determine that, in the presence of a competing flow using CCA α, a flow using a CCA β suffers, *e.g.* a 50% reduction in throughput or a 10% increase in latency. We refer to this degradation as the harm.

Perhaps the most crucial aspect of harm is recognizing that we are not designing a clean-slate Internet. We believe that we need to shift our focus from if pairs of CCAs 'fairly' share and instead focus on how a new CCA impacts the *status quo* (whether or not the new algorithm damages the performance of existing traffic). We argue that our deployment threshold should be based on the amount of harm *already caused* by deployed algorithms.

If the amount of harm caused by flows using a new algorithm α on flows using an algorithm β is within a bound derived from how much harm β flows cause other β flows, we can consider α deployable alongside β.

Turning this insight into a concrete threshold is challenging; we present three approaches in §4. Nonetheless, we believe that a harm-based threshold is the right way forward. A harm-based threshold avoids ideal-driven goalposts like fairness by settling for outcomes that are unfair, but no worse than the status quo. A harm based-threshold does not suffer from the limits of throughput centricity. We can speak of throughput-harm as well as latency-harm, FCT-harm, or loss-harm. Lastly, a harm based-threshold prescribes no mechanism or behavior to replicate, which allows for a broader range of outcomes than mimicry.

In what follows, we discuss the limitations of fairness and mimicry in §2. We then introduce harm in §3, how to quantify it, and our intuition as to why a harm-based threshold is the right path forward for the Internet. In §4 we propose three possible harm-based thresholds. Finally, in §6, we leave open questions for the community and conclude.

2 FAIRNESS AND MIMICRY

Our goal in this paper is to identify a *deployment threshold*: a bound on the behavior of a new CCA when competing with legacy CCAs. If a new CCA meets the conditions of the threshold, we ought to consider it deployable. Furthermore, if a new CCA *does not* meet the conditions, it should be considered unacceptable for deployment. In this section, we discuss the limitations of prior approaches to evaluating new CCAs and their interactions with other algorithms. We argue that both fairness(§2.1) and mimicry based approaches (§2.2) are unsuitable for a deployment threshold. Through our discussion, we derive a set of desiderata for a deployment threshold, which we list in Table 1.

2.1 Limitations of Fairness

Fairness measures are the typical tool used for determining if a new CCA is deployable in the Internet [1, 6]. A fairness-based threshold, asserts if a CCA α is fair to a legacy CCA β, then the algorithm is deployable. Fairness is typically measured by looking at the throughput ratio between competing CCAs or by computing Jain's Fairness Index (JFI) [12], which returns a number between 1 (perfectly meeting the expected fair allocation) and 0 (the closer to 0, the more 'unfair').

Typically, fairness is measured assuming infinitely backlogged flows: each flow wants to use an equal fraction of the bottleneck link. In this case, we expect the throughput ratio and Jain's Fairness Index to be 1. In reality, not all flows are infinitely backlogged and not all flows can fully utilize their equal share. In that case, equal rate fairness has a well-known shortcoming: it does not account for the *demand* of each flow. Demand is the amount of resources a flow uses when operating in absence of contention. Consider a new CCA α competing against a TCP NewReno flow on a 10Gbps link. We know that the NewReno algorithm will fail to take advantage of the full link capacity due to its slow additive increase and aggressive reaction to loss [8].

Intuitively, it would seem that a new flow using α should be able to take advantage of the remainder of link capacity, but equal rate fairness disallows such an outcome. Including demand is important for a deployment threshold: a new CCA should not be penalized as 'unfair' to a legacy CCA when the legacy CCA, on its own, is incapable of claiming its equal share of the network. Hence, we list 'Demand-Aware' as our first item in Table 1. Unlike equal rate fairness, max-min fairness, allows for flows to increase its rate if it would not decrease the rate of any other flows [21]. Thus, when we refer to fairness throughout this paper, we refer to max-min fairness.

Although we argue against a fairness-based deployment threshold, fairness measures have many practical uses in the design of CCAs and scheduling systems. In this section, we do not attack fairness as a valued measure for systems in general. In particular, we believe throughput fairness is sometimes a desirable property, especially for intra-CCA interactions. Nonetheless, we object to using fairness measures as a threshold for determining CCA deployability for the reasons we discuss as follows.

2.1.1 Throughput-Centricity. Fairness strategies focus on sharing a resource like 'dividing a pie'. This is appropriate for performance metrics like throughput since there is a maximum link capacity which must be divided by all

Demand-Aware	Like max-min fairness, takes into account the fact that some flows have different demands than others.
Multi-metric	Addresses throughput, latency, flow completion time, or any other performance metric.
Practical	Practical, rather than ideal-driven (unlike fairness); it should be feasible for new CCAs to meet this threshold.
Status-Quo Biased	Does not suffer from the assumption of balance. Specifically, we should worry about the impact of a new CCA on the currently deployed CCAs, and should not focus on how deployed CCAs harm a new CCA.
Future-Proof	Useful on a future Internet where none of today's current CCAs are deployed; does not restrict the development of new CCAs based on the idiosyncrasies of currently deployed CCAs.

Table 1: Desiderata for a deployment threshold, derived from insights and shortcomings of fairness and mimicry.

competing flows. However, modern CCA designers consider many performance metrics which do not always easily map to a network resource which should be divided or shared, *e.g.*, latency, loss rate, flow completion time.

Consider a future Internet where the majority of Internet services use a TCP algorithm called TINYBUFFER which, like algorithms BBR and Vegas, has very little queue occupancy and therefore low latency and loss. Hence, TINYBUFFER provides very good user experience for video conferencing and voice calls. A new company wishes to introduce a new algorithm α, which is derived from Cubic and therefore fills buffers. Is this an acceptable deployment? Videochat users of TINYBUFFER would likely say no, since α flows competing with TINYBUFFER would increase latency and loss, harming their video calls.

Unfortunately, we cannot say that the behavior of α relative to TINYBUFFER is 'unfair' – buffer occupancy is not a resource we want to divide equally. Instead, it is a value we simply want minimized which is not captured by the concept of fairness. For this reason, we say that fairness is not 'multi-metric', our second requirement in Table 1.

2.1.2 Ideal-Driven Goal Posts. A second problem with fairness is that, even when we focus on throughput alone, it is simply very difficult to achieve. For example, Cubic and Reno, both known to have a 'short flow penalty' where short flows do not achieve their fair share of throughput before completing [14, 16]. BBR is also unfair to connections with shorter RTTs, allocating them lower throughput than competing connections with long RTTs [2].[1] If algorithm designers

[1]Reno's throughput allocation has the opposite bias.

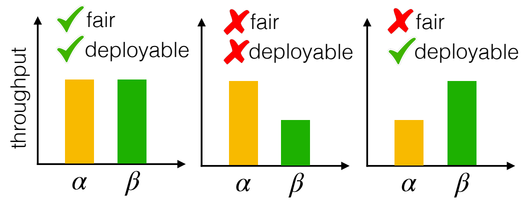


Figure 1: Fairness and the assumption of balance.

cannot achieve perfect fairness in the intra-CCA context, why would it make sense to expect algorithm designers to achieve perfect fairness in the more challenging inter-CCA context? We list our third requirement in Table 1 as being 'practical.'

Many readers at this point may find themselves thinking, 'Of course we don't expect new algorithms to be *perfectly* fair to existing ones!' But, even if we do not expect perfect fairness, the community still leaves algorithm designers with no real guideline for acceptability based on fairness. This often results in CCA designers making the argument that it acceptable for their algorithm to be somewhat unfair to legacy CCAs. [1, 5, 6]. Nonetheless, we lack a practical threshold and clear consensus on how far from perfectly fair sharing a new algorithm may be permitted to deviate.

2.1.3 The Assumption of Balance. We call our third and final objection to fairness the Assumption of Balance, meaning that it values the performance outcome of *both* the new CCA and the existing, deployed CCA.

To illustrate our objection, we look to Figure 1. We imagine a future Internet where most senders use some algorithm β; two senders are transmitting very large files over a 100Mbps link. Sender B is using β, but sender A is using some brand new algorithm α. Both senders' demands are 100Mbps – both desire to use as much capacity as they can. However, sender B achieves 90Mbps throughput and sender A only achieves 10Mbps. Is this fair? No. In both the above scenario and a scenario where the allocations are swapped (B receives 10Mbps, A receives 90Mbps) have the same JFI – 0.61. But, it should be perfectly acceptable to deploy α *if α is the one receiving unfair treatment* – no users other than user A, who chose to use α, would be impacted negatively.

It is highly unlikely that many new services will set out to deliberately deploy algorithms that penalize the performance of their own flows – but given the difficulty of achieving perfect fairness (§2.1.2) it is likely that this outcome may happen in some scenarios.[2] A very deployable, friendly algorithm would err on the side of harming their own connections where a more aggressive algorithm would err on the side of harming those

[2]Consider Google's BBR, which, when one BBR flow competes with one Reno flow will only consume 40% of link capacity – less than its fair share [2, 24]. Furthermore, 'background' CCA algorithms, such as TCP-Nice [23] and LEDBAT [20] deliberately only consume leftover bandwidth that is underutilized by other connections.

of others – and a good measure of deployability should be able to distinguish between the two. Thus, our fourth requirement in Table 1 is that the new threshold be 'status-quo biased.'

2.2 Limitations of Mimicry

A mimicry-based threshold asserts if a CCA α mimics the behavior of a legacy CCA β, then the algorithm is deployable. Two mimicry based approaches are:

TCP-Friendly Rate Control (TFRC) [18]: a CCA using TCP-Friendly Rate Control transmits at a rate $\leq \frac{MSS}{RTT*\sqrt{p}}$ for p the link loss rate; this formula describes TCP Reno's average sending rate over time [15, 17].

RTT-biased Allocation [7]: a CCA obeying RTT-biased Allocation grants more throughput to connections with low RTTs than to those with higher RTTs; this behavior is a property of TCP Reno.

Mimicry introduces an elegant solution to the challenge of ideal-driven goal posts: it should be acceptable to deploy a new CCA which introduces the same side-effects – fair or unfair – as the already deployed algorithm. A mimicry-based approach is always practical because the existence of the original, deployed algorithm demonstrates at least one way to achieve these performance outcomes (although as TFRC illustrates, a CCA may use a different algorithm than the original to achieve this outcome).

However, mimicry will not serve as a good threshold because it binds new algorithms to replicating the often undesirable idiosyncrasies of the deployed CCA, and hence stifles improvements and evolution. For example, TFRC limits new CCAs from achieving high throughput. Indeed, an animus for most of the new CCAs which are replacing Reno on today's Internet (*e.g.* Cubic [9], Compound [22]) was to supersede the $\leq \frac{MSS}{RTT*\sqrt{p}}$ rate, as this very limit is what prevents Reno from taking advantage of high-capacity links.

Similarly, RTT-biased Allocation is not an ideal outcome that was proposed from first principles: it is simply the throughput allocation that Reno achieves. Given this, perhaps it should be acceptable, on an Internet dominated by RTT-biased algorithms, to deploy yet another RTT-biased algorithm – but RTT-bias should not be enshrined as the *goal itself.*[3]

In a future Internet where no one any longer deploys Reno variants, a mimicry-based threshold would lack grounding; even worse, it could prevent us from reaching a future Internet with improved fairness, lower latency, *etc.* due to our replicating the inherent limitations of existing CCAs. Thus, our final requirement in Table 1 is that our threshold be 'future-proof.'

[3]Some will disagree with us, arguing that longer flows consume more network resources and therefore *should* receive lower throughput [7] – and call this 'RTT Fairness.' This could be a good reason to continue with RTT-biased allocation. Our point is that, if we really prefer Max-Min fairness, it would be bad to continue with RTT-biased allocation simply because we have required ourselves to mimic Reno's behavior.

3 HARM

We now present harm, and argue that a harm-based threshold would meet all of our desiderata. In the next section (§4), we present a few possible harm-based thresholds.

3.1 Calculating Harm

We imagine a TCP connection where Alice is video conferencing with her friend Bob. When running alone, the connection achieves 15Mbps of throughput, packets arrive with 40ms latency, and jitter is 10ms. When Alice's roommate Charlie starts a large file transfer, Alice's video conference connection drops to 10Mbps of throughput, latency increases to 50ms, and jitter increases to 15ms. Since all of these performance metrics became worse due to to Charlie's connection, we say that Charlie's connection caused *harm* to Alice's connection.

We can measure harm by placing it on a [0,1] scale (much like Jain's Fairness Index [12]) where 1 is maximally harmful, and 0 is harmless. Let x = demand (solo performance); let y = performance after introduction of a competitor connection. For metrics where 'more is better' (like throughput and QoE) harm is $\frac{x-y}{x}$. For metrics where 'less is better' (like latency or flow completion time) harm is $\frac{y-x}{y}$. On this scale, Charlie caused 0.33 throughput harm, 0.2 latency harm, and 0.33 loss harm.

The amount of harm done by one TCP connection to another depends not only on the algorithm(s) in use, but also the *network* and *workload.* For example, a connection with a very bursty traffic pattern will induce higher worst-case latency for competitor connections than a connection with very steady, paced traffic (a change in workload). Throughput harm between TCP Cubic and BBR varies depending on whether the network is shallow or deeply buffered [2] (a change in network). Note that we also include background traffic in our model of the 'network'.

Harm is multi-metric. It can be computed for latency, throughput, jitter, or any other quantifiable measure.

Harm is demand-aware. Harm is computed from a baseline of a TCP connection running on its own; new algorithms are not penalized when deployed algorithms perform poorly due to their own limitations. A flow which only ever consumes 10% of link capacity has no throughput harm done to it when another flow arrives and consumes the remaining 90%. A flow using a CCA which occupies all of the buffer space has no latency harm done to it when another buffer filling algorithm competes with it.

Harm is status-quo biased. Harm does not suffer from the assumption of balance because it does not involve the performance of connections using the new algorithm at all. Harm only measures the new connection's impact on existing flows.

In the next section (§3.2), we provide the intuition behind how to derive a threshold from harm, and why such a threshold would be practical and future-proof. Then (§4) we discuss several proposals for a threshold based on harm.

3.2 A Harm-Based Threshold

Simply measuring harm does not tell us whether or not the harm introduced by a new algorithm is acceptable. We take inspiration here from TCP mimicry (§2.2): the behaviors of deployed algorithms today should be our guidelines for what is acceptable. However, we need to relax mimicry to allow innovation and improvement. We suggest that, *if the harm done by a new CCA α to an widely-deployed CCA β is comparable or less than the harm done when β competes against β, we should consider it acceptable to deploy.* We say that the harm of the *alpha* upon *beta* is *bounded* by the harm already done by *alpha* to itself. We leave up to discussion how 'widely-deployed' a β must be in order to merit protection under our deployment threshold.

For example, a new algorithm α is developed for a future Internet where the predominant algorithm is called K-LATENCY. Every flow in K-LATENCY maintains a constant queue occupancy of exactly k packets – thus, latency increases linearly with the number of competing flows. A harm based threshold, grounded in how K-LATENCY connections interact in the intra-CCA scenario, would deem α acceptable with respect to latency so long as it never buffers more than k packets per flow.

The difference between bounded harm and mimicry can be subtle. In the above example, mimicry would demand that α always buffer exactly k packets per flow, just as K-LATENCY does (restricting improvement). On the other hand, bounded harm allows α to buffer *up to* k packets while competing with K-LATENCY (allowing for improvement, if feasible). Bounded harm allows for a broader range of outcomes than mimicry. Further the bound for α is undefined when not competing with K-LATENCY – α flows may behave differently when competing with other α flows or competing with some third CCA γ. For example, 'modal' algorithms like Copa [1], exhibit very diverse behaviors within one CCA, adjusting behavior online as different cross-traffic is detected. But, an algorithm need not be explicitly modal to have bounded-harm outcomes that are acceptable across multiple CCAs.

On an Internet with many competing algorithms β,γ,ϕ... (as today's Internet) one might ask why we are bounding harm from a new α to the harm that each algorithm does to itself. We suggest that α do harm to β flows that is bounded by the harm caused to β by other β flows, and that α do harm to γ flows that is bounded by the harm caused to γ by other γ flows. Why not bound α's behavior in the harm done by β to γ and vice versa?

One reason we reject this approach is that on today's Internet, there are many well-known pessimal cross-CCA outcomes (*e.g.*, BBR's starving Cubic on high capacity links [11]). A CCA designer for a new CCA α could use the existence of any single pessimal scenario to justify continuing that behavior with α. By bounding harm to a CCA by the harm *any* other CCA, we settle for the absolute lowest common denominator in performance outcomes.

Furthermore, bounding the inter-CCA harm (caused by a new α on an existing β) by the intra-CCA harm (caused by β to β) carries forward the design trade-offs made by CCA developers. CCA designers tune their algorithms for the outcomes they want under intra-CCA competition. The designers of Reno and Cubic allow the loss rate to increase with the number of competing Reno/Cubic flows [15]. The designers of BBR aim to keep buffers empty, but do allow the queue occupancy to increase when multiple BBR flow compete [10]. In this way, the designers implicitly encode their tolerance to performance degradation for each metric. As a consequence, bounding inter-CCA harm by the intra-CCA harm means that new CCAs will respect the implicitly expressed preferences of typical traffic.

In practice, we already see CCA designers try to make the argument that their algorithm is deployable because it is not any more aggressive towards the status-quo (Cubic) than it is to itself [1, 4–6]. Unforunately, they try to make this argument using fairness, which suffers from the limitations dicussed in §2.1. We believe explicitly measuring harm would give CCA designers a more clear language to argue for deployability. The networking community need only agree on concrete harm-based threshold to provide a guideline for how much harm is allowable.

Unlike fairness, a harm-based threshold is practical. The original algorithm is an existence proof that the demanded threshold is feasible and hence the goal posts *cannot* be set too high.

Unlike mimicry, a harm-based threshold is future-proof. A harm-based threshold does not require replicating the behavior of a deployed algorithm – only matching or improving upon its outcomes while competing with that algorithm. Furthermore, as CCAs die out in popularity or new ones arise, a harm-based threshold will shift to requiring similar harm to the new algorithms rather than the old ones.

The reader may wonder how an algorithm can be both status-quo biased and future proof. Consider the bi-modal Copa algorithm [1]. Unlike a mimcry approach, Copa only seeks to match the throughput of loss-based algorithms when it detects its competition is loss-based. When Copa is alone, it behaves like a delay-based algorithm, minimizing excess queueing. So, in a world, where legacy algorithms are

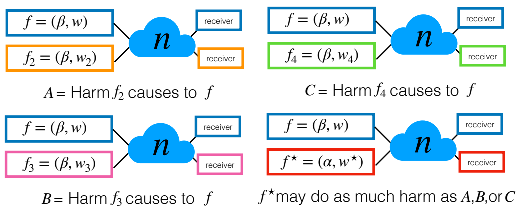

$f = (\beta, w)$
$f_2 = (\beta, w_2)$
n
receiver
receiver
A = Harm f_2 causes to f
$f = (\beta, w)$
$f_4 = (\beta, w_4)$
C = Harm f_4 causes to f
$f = (\beta, w)$
$f_3 = (\beta, w_3)$
B = Harm f_3 causes to f
$f = (\beta, w)$
$f^\star = (\alpha, w^\star)$
$f^\star$ may do as much harm as A, B, or C

Figure 2: Under Worst Case Bounded Harm, a new α may do as much harm to a β flow f as *any* other β flow in the same network.

nearly phased out, Copa will be able to behave differently than legacy loss-based algorithms. Further, if Copa were to become widely deployed, subsequent algorithms would then measure harm against Copa's delayed-based behavior.

Deriving a concise, usable harm-based threshold is challenging. So far, we have only described how to measure harm and why a harm-based threshold is overcomes the limitations of fairness and mimicry. However, we have not yet defined a concrete deployment threshold based on harm. We believe a concrete threshold should consider how harm plays out between existing algorithms. In the next section, we propose several harm-based thresholds – and invite the community to scrutinize and improve upon them.

4 CONCRETE THRESHOLDS

We now discuss several options for a concrete threshold, driven by our intuition from §3. We define the following variables and functions to help discuss each:

Let a TCP connection ('flow') $f = (a, w)$ for $a \in A$, the set of all congestion control algorithms, and $w \in W$, the set of all connection workloads (short flows, video streams, *etc.*).

Let M be the set of all performance metrics (throughput, latency, QoE, jitter, *etc.*).

Let N be the set of all network paths (with varying throughput, latency, loss rate, queue capacity, and background traffic).

Let $\textsc{harm}(f_i, f_j, n \in N, m \in M)$ be the harm done to f_i by f_j according to metric m in network n as defined in §3.

Note, we assume to compare the harm across workloads, you must compare them using harm functions with the same metric. Because of our status-quo bias, the harm function should be determined by the workload and CCA β. For example, if we assume β is adaptive bitrate video using BBR, the harm function might use some metric for quality of experience like rebuffering rate.

4.1 Worst Case Bounded Harm

We illustrate our first potential threshold in Figure 2. We imagine a network with numerous applications and servers all of which use a legacy CCA β. We want to deploy a new application with workload $w^\star$ using CCA α. Is this acceptable?

We can start by considering whether it is acceptable to deploy $(\alpha, w^\star)$ alongside a specific flow $f = (\beta, w)$. Worst case bounded harm looks to the *worst case harm* f might receive from *any* of the other services, with their workloads $w_1, w_2, \ldots$. If $(\alpha, w^\star)$ does not more harm than this worst case, we would consider it acceptable.

Definition: A TCP connection $f^\star = (\alpha, w^\star)$ for α a new algorithm and $w^\star$ a specific traffic workload, respects *worst case harm* with respect to metric m for an algorithm β iff

$$\forall f = (\beta, w \in W), \forall n \in N:$$
$$\textsc{harm}(f, f^\star, n, m) \leq \max_{w^\dagger \in W} (\textsc{harm}(f, (\beta, w^\dagger), n, m))$$

We similarly say that the algorithm α itself has *worst-case harm equivalence* with respect to m for β if all connections $f_n = (\alpha, w_n \in W)$ all respect worst-case harm in m.

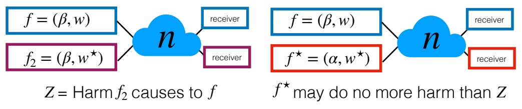
Suitability as a deployment threshold: Worst-case bounded harm, as a threshold, is too loose: it can allow the outcomes of pathological scenarios to become common through the deployment of a new CCA. Consider a CCA β which is widely deployed and has perfect performance under competition: an ideal fair-sharing allocation, no additional latency, jitter or loss due to new flows, *etc.*, with only one exception. A pessimal workload, $\hat{w}$ can cause any other flow to suffer starvation. However, $\hat{w}$ is extremely rare – so rare in practice that β generally works well. Nonetheless, a malicious protocol designer can take advantage of that – observing that there exists *any* workload that leads a flow f to starvation can justify that *all* of the flows using the new CCA cause starvation for f. This is the same logic – avoiding falling to the lowest common denominator – we used to argue against bounding harm across arbitrary CCA pairs (the max harm in β vs some γ, §3.2), and it is the same logic that leads us to reject bounding harm across different networks (the max harm across all $n \in N$).

4.2 Equivalent Bounded Harm

To overcome the least common denominator challenge from Worst-Case Bounded Harm, we consider an approach where we pin the workloads being compared. We illustrate the Equivalent Bounded Harm in Figure 3. We start by focusing on a pair of workloads w and $w^\star$ in a legacy network where all services use β. We want to switch the service with workload $w^\star$ to use α. With Equivalent Bounded Harm, α would be acceptable iff $(\alpha, w^\star)$ does no more harm to (β, w) than $(\beta, w^\star)$ would.

Definition: A TCP connection $f^\star = (\alpha, w^\star)$ for α a new algorithm and $w^\star$ an specific traffic workload, has *equivalent harm* with respect to metric m for an algorithm β iff

$$\forall f = (\beta, w \in W), \forall n \in N:$$
$$\textsc{harm}(f, f^\star, n, m) \leq \textsc{harm}(f, (\beta, w^\star), n, m)$$



$f = (\beta, w)$
$f_2 = (\beta, w^\star)$
n
receiver
receiver
Z = Harm f_2 causes to f
$f = (\beta, w)$
$f^\star = (\alpha, w^\star)$
n
receiver
receiver
$f^\star$ may do no more harm than Z

Figure 3: Under Equivalent Bounded Harm, a new α with workload $w^\star$ may do as much harm to a β flow as a β flow with workload $w^\star$ as well.

We similarly say that the algorithm α itself has *equivalent bounded harm* with respect to m for β if all connections $f_n = (\alpha, w_n \in W)$ are harmless in m.

Suitability as a deployment threshold: Equivalent bound-ed harm is too strict to serve as a threshold. Consider a CCA BIGFLOW where large flows competing with short flows lead to unfair outcomes. Large flows achieve 75% of available bandwidth capacity and short flows achieve only 25% of available bandwidth capacity. Requiring harm equivalence would entail that short flows using any new algorithm α would only ever be able to achieve up to 25% of the available link capacity when competing with BIGFLOW. Equivalent bounded harm hence falls too close to the trap of mimicry in constraining improvement.

4.3 Symmetric Bounded Harm

Our third proposal, shown in Figure 4, sits between the too-strict harm-equivalence and the too-permissive worst-case bounded harm. Symmetric Bounded Harm considers pairs of workloads $w, w^\star$ like harm-equivalence. For an existing CCA β with a flow f running workload w, the flow can receive as much harm from $(\alpha, w^\star)$ as *either* f would experience from $(\beta, w^\star)$ *or* as f would *inflict* on $(\beta, w^\star)$.

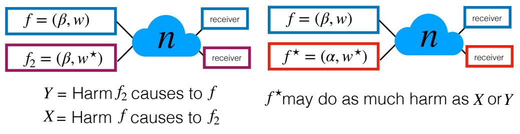
Returning to BIGFLOW, a large flow f_1 may only receive 25% throughput-harm from a small flow f_2, but since it also inflicts 75% throughput-harm on f_2, a small flow using a new CCA α can inflict up to 75% throughput-harm on f_1.

Definition: A TCP connection $f^\star = (\alpha, w^\star)$ for α a new algorithm and $w^\star$ an specific traffic workload, respects *symmetric-bounded harm* with respect to metric m for an algorithm β iff

$$\forall f = (\beta, w \in W), \forall n \in N:$$
$$\textsc{harm}(f, f^\star, n, m)$$
$$\leq \max(\textsc{harm}(f, (\beta, w^\star), n, m), \textsc{harm}((\beta, w^\star), f, n, m))$$

We similarly say that the algorithm α itself has *symmetric-bounded harm equivalence* with respect to m for β if all connections $f_n = (\alpha, w_n \in W)$ all respect symmetric-bounded harm in m.

Suitability as a deployment threshold: Symmetric bound-ed harm resonates with a sense of justice: 'do unto other flows as you would have other flows do to you.' It is not too restrictive, like harm equivalence, but it is not vulnerable to the expansion of harm as we saw in worst-case harm. For these reasons, we prefer symmetric-bounded harm as a potential threshold to the prior two harm-based threshold. Nonetheless, we believe that further work is needed to refine an ideal harm-based threshold.



$f = (\beta, w)$
$f_2 = (\beta, w^\star)$
n
receiver
receiver
Y = Harm f_2 causes to f
X = Harm f causes to f_2
$f = (\beta, w)$
$f^\star = (\alpha, w^\star)$
n
receiver
receiver
$f^\star$ may do as much harm as X or Y

Figure 4: Under Symmetric Bounded Harm, a new α with workload $w^\star$ may do as much harm to a β flow f as f does to $(\beta, w^\star)$ or as $(\beta, w^\star)$ does to f.

5 OPEN QUESTIONS

Defining a harm-based threshold is only a first step in setting the bar for deploying a new CCA in the Internet. Given a harm-based threshold, we can eventually develop a modern evaluation methodology for CCA deployability. This leaves many open questions and directions for future work:

- Is there an better threshold that improves on symmetric bounded harm?
- Given that Internet outcomes always have some distribution of results, is there 'leeway' in harm? Should we worry about average or worst-case results?
- How widely deployed must a legacy CCA be in order to merit protection by our threshold?
- What are the right workloads for deployability testing?
- If we have a threshold, should it be enforced? If so, how?

6 DISCUSSION & CONCLUSION

We have argued for the networking community to adopt a *deployment threshold* which provides a firm definition for when a new CCA is allowed to be deployed on the Internet, and when it is not. We believe that the right way forward is by analyzing harm, a way to quantifiably measure the outcomes of introducing a new CCA to the Internet.

We argue that the harm caused by a new CCA α should be bounded by the status quo: do no more harm to flows from a CCA β than β already inflicts upon itself. In this way, algorithms can improve upon outcomes under contention (do *no more harm* than) but are not required to meet overly idealistic goals. A challenge remains in turning this insight into a concise and practical formula for a threshold.

ACKNOWLEDGMENTS

We thank the anonymous referees, Neal Cardwell, and Keith Winstein for their valuable comments and helpful suggestions. We also thank Jana Iyengar for fruitful conversations about congestion control deployability. This research was funded by a Facebook Emerging Scholar Fellowship, NSF Grant # 1850384, and a Google Faculty Research Award

REFERENCES

[1] V. Arun and H. Balakrishnan. Copa: Practical Delay-Based Congestion Control for the Internet. In *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*, pages 329–342, Renton, WA, 2018. USENIX Association.

[2] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson. BBR Congestion Control. In *Presentation in ICCRG at IETF 97th meeting*, 2016.

[3] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson. BBR Congestion Control: An update. In *Presentation in ICCRG at 98th meeting*, 2017.

[4] N. Cardwell, Y. Cheng, S. Hassas Yeganeh, I. Swett, V. Vasiliev, P. Jha, Y. Seung, M. Mathis, and V. Jacobson. BBRv2: A Model-Based Congestion Control. In *Presentation in ICCRG at IETF 104th meeting*, 2019.

[5] M. Dong, Q. Li, D. Zarchy, P. B. Godfrey, and M. Schapira. PCC: Re-architecting Congestion Control for Consistent High Performance. In *Proceedings of the 12th USENIX Conference on Networked Systems Design and Implementation*, NSDI'15, pages 395–408, Berkeley, CA, USA, 2015. USENIX Association.

[6] M. Dong, T. Meng, D. Zarchy, E. Arslan, Y. Gilad, B. Godfrey, and M. Schapira. PCC Vivace: Online-Learning Congestion Control. In *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*, pages 343–356, Renton, WA, 2018. USENIX Association.

[7] S. Floyd. Connections with Multiple Congested Gateways in Packet-switched Networks Part 1: One-way Traffic. *SIGCOMM Comput. Commun. Rev.*, 21(5):30–47, Oct. 1991.

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

[9] S. Ha, I. Rhee, and L. Xu. CUBIC: A New TCP-friendly High-speed TCP Variant. *SIGOPS Oper. Syst. Rev.*, 42(5):64–74, July 2008.

[10] M. Hock, R. Bless, and M. Zitterbart. Experimental evaluation of BBR congestion control. In *2017 IEEE 25th International Conference on Network Protocols (ICNP)*, pages 1–10, Oct 2017.

[11] G. Huston. BBR TCP. http://www.potaroo.net/ispcol/2017-05/bbr.html, May 2017.

[12] R. Jain, D.-M. Chiu, and W. R. Hawe. A Quantitative Measure of Fairness and Discrimination for Resource Allocation in Shared Computer Systems. DEC Research Report TR-301, 1984.

[13] A. Legout and E. Biersack. Beyond TCP-Friendliness: A New Paradigm for End-to-End Congestion Control. Technical report, 1999.

[14] Q. Li, M. Dong, and P. B. Godfrey. Halfback: Running Short Flows Quickly and Safely. In *Proceedings of the 11th ACM Conference on Emerging Networking Experiments and Technologies*, CoNEXT '15, pages 22:1–22:13, New York, NY, USA, 2015. ACM.

[15] M. Mathis, J. Semke, J. Mahdavi, and T. Ott. The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm. *SIGCOMM Comput. Commun. Rev.*, 27(3):67–82, July 1997.

[16] R. Mittal, J. Sherry, S. Ratnasamy, and S. Shenker. Recursively Cautious Congestion Control. In *Proceedings of the 11th USENIX Conference on Networked Systems Design and Implementation*, NSDI'14, pages 373–385, Berkeley, CA, USA, 2014. USENIX Association.

[17] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP Throughput: A Simple Model and Its Empirical Validation. In *Proceedings of the ACM SIGCOMM '98 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, SIGCOMM '98, pages 303–314, New York, NY, USA, 1998. ACM.

[18] J. Padhye, J. Kurose, D. Towsley, and R. Koodli. A Model Based TCP-friendly Rate Control Protocol. In *Proceedings of NOSSDAV '99*. Citeseer, 1999.

[19] A. Shah. BBR Evaluation at a Large CDN, Sep 2019.

[20] S. Shalunov, G. Hazel, J. Iyengar, and M. Kuehlewind. Low Extra Delay Background Transport (LEDBAT). RFC 6817, Dec. 2012.

[21] R. Srikant. *The mathematics of Internet congestion control*. Springer Science & Business Media, 2012.

[22] K. Tan, J. Song, Q. Zhang, and M. Sridharan. A Compound TCP Approach for High-Speed and Long Distance Networks. In *Proceedings-IEEE INFOCOM*, 2006.

[23] A. Venkataramani, R. Kokku, and M. Dahlin. TCP Nice: A mechanism for background transfers. *ACM SIGOPS Operating Systems Review*, 36(SI):329–343, 2002.

[24] R. Ware, M. K. Mukerjee, S. Seshan, and J. Sherry. Modeling BBR's Interactions with Loss-Based Congestion Control. In *Proceedings of the Internet Measurement Conference*, IMC '19, pages 137–143, New York, NY, USA, 2019. ACM.

[25] K. Winstein and H. Balakrishnan. TCP Ex Machina: Computer-generated Congestion Control. In *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, SIGCOMM '13, pages 123–134, New York, NY, USA, 2013. ACM.

[26] K. Winstein, A. Sivaraman, and H. Balakrishnan. Stochastic Forecasts Achieve High Throughput and Low Delay over Cellular Networks. In *Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation*, NSDI'13, pages 459–472, Berkeley, CA, USA, 2013. USENIX Association.

[27] D. Wischik, C. Raiciu, A. Greenhalgh, and M. Handley. Design, Implementation and Evaluation of Congestion Control for Multipath TCP. In *Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation*, NSDI'11, pages 99–112, Berkeley, CA, USA, 2011. USENIX Association.