互联网拥塞管理原则

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

[1] 加州大学伯克利分校，[2] 苏黎世联邦理工学院，[3] BreezeML，[4] 布朗大学，[5] 麻省理工学院，[6] 哥伦比亚大学
[7] 华盛顿大学，[8] 耶路撒冷希伯来大学，[9] 国际计算机科学研究所

摘要

鉴于 TCP 友好性范式存在技术缺陷且其遵守程度日益降低，我们必须重新思考互联网应如何管理带宽分配。我们从基本原理出发探讨这一问题，但仍在互联网当前架构和商业安排的限制范围内。我们提出了一种新的框架——递归拥塞份额（RCS），它能独立于流所使用的拥塞控制算法来分配带宽，同时又与互联网的经济状况相一致。我们通过博弈论计算、模拟以及网络仿真表明，RCS 能够实现这一目标。

CCS 概念

· 网络 → 网络设计原则。

关键词

网络架构

ACM 参考格式：

Lloyd Brown、Albert Gran Alcoz、Frank Cangialosi、Akshay Narayan、Mohammad Alizadeh、Hari Balakrishnan、Eric Friedman、Ethan Katz-Bassett、Arvind Krishnamurthy、Michael Schapira、Scott Shenker。2024 年。《互联网拥塞管理原则》。载于 2024 年 ACM SIGCOMM 会议（ACM SIGCOMM '24），2024 年 8 月 4 日至 8 日，澳大利亚新南威尔士州悉尼。ACM 出版，纽约，美国，15 页。https://doi.org/10.1145/3651890.3672247

1 简介

除了是一项技术奇迹，其架构能够适应规模、速度、技术和用途方面令人难以置信的变化之外，互联网还是一项大规模的去中心化资源共享实验。由于计算机通信具有突发性，互联网依靠分组级的统计复用来实现合理的效率。为了应对不可避免的过载，互联网依靠基于主机的拥塞控制算法（CCAs）。

采用这种方法，一个流所获得的带宽在很大程度上取决于其拥塞控制算法（CCA）的激进程度。互联网社区很快意识到，用户会有动力部署越来越激进的拥塞控制算法，从而导致网络过载。为防止这种情况发生，互联网社区非正式地要求所有拥塞控制算法都必须是 TCP 友好的（以下简称 TCPF），其定义见 [24]：“如果一个流的到达率不超过 TCP 流的到达率，则该流即为 TCP 友好的。”[1] TCPF 主要适用于公共互联网上的广域网流量，本文主要关注这种情况。在数据中心、企业以及私有广域网（WAN）等私有部署中，存在单一的管理机构，因此有专门的带宽分配解决方案。

TCPF 存在诸多实际和技术问题。先前的研究表明，TCPF 很难强制执行[46]，而且我们对竞争性拥塞控制算法（CCA）动态的理解在大规模情况下会失效[38]。此外，TCPF 限制了 CCA 快速提升的能力，使其无法实现完全效率[49]，并且阻碍了新的对延迟敏感的 CCA 的出现（如 Copa [5] 和 Nimbus [28] 所示）。显然，TCPF 已不再是部署新 CCA 的严格要求，实际上，非 TCPF 的 CCA 将会得到广泛部署。例如，TCP 不友好的 CCA BBR [17, 18, 47] 已被谷歌、亚马逊、Akamai、Dropbox 和 Spotify 广泛用于其大量流量。

鉴于 TCPF 存在严重缺陷且已不再被主要互联网参与者所遵循，我们应当思考是否存在能够替代 TCPF 模式的合适方案。这一简单却核心的问题正是本文的重点所在。为此，我们从基本原理出发，探讨何种新的概念框架能够取代 TCPF。然而，尽管我们从基本原理进行推理，但并非从零开始。我们假定，在我们的设计/部署时间范围内，互联网架构（例如 IP、BGP 以及尽力而为的服务模式）及其商业安排（例如，ISP 如何收费、相互对等以及广泛遵循无谷路由[26]）不会发生根本性变化。因此，我们寻求一种概念基础，以指导互联网如何分配带宽，该基础应满足以下两点：（一）能够在现有架构内实现（尽管需要额外的协议和机制）；（二）所提供的带宽分配应与当前各方之间的商业安排相一致。

本文做出了以下贡献：

· 我们将 CCA 独立性（CCAI）（第 2 节）的目标明确为共享带宽的基本目标。

与先前文献[12]中的具体主张以及文献中的一般性假设（如[20]）形成对比的是

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

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

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

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

（参考了许多相关论文）我们表明，若天真地（即，权重设置与拓扑结构无关）应用加权公平队列或其分层变体，则无法实现我们的目标（第 2 节和第 4.4 节）。

· 我们推导出三项原则，以实现通用可控人工智能，同时与互联网经济保持一致（第 3 节）。

基于这些原则，我们提出了一种方案，该方案在每个节点的分层加权公平共享算法（第 4 节）中设置权重，既依据用户访问协议（这是设置权重的传统方法），又依据数据包在网络中所经路径（这并非传统方法）。这种依赖于网络拓扑结构的权重设置方式新颖且对于实现 CCAI 至关重要。

我们通过模拟和仿真表明，在绝大多数情况下，RCS 能够实现 CCAI（第 5 节、第 6 节）。

2 替换 TCPF

在探讨如何取代 TCPF 之前，有人可能会问，为何我们需要任何一种指导互联网如何共享带宽的框架。原因很简单：如果没有一个连贯的资源共享框架，随着时间的推移，可能会部署越来越多激进的拥塞控制算法，由此导致的整体拥塞加剧会对互联网造成损害。

TCPF 的关键问题以及缺乏框架的问题在于网络处于被动地位，因此在拥塞链路上，激进的拥塞控制算法（CCA）会获得更多的带宽。我们提议走向另一个极端，要求网络主动确保所有合理的 CCA 在相同的静态条件下都能获得相同的带宽。我们将此称为 CCA 独立性（CCAI）。CCAI 消除了对单一标准 CCA 的需求，促进了 CCA 的广泛多样性和创新。例如，提供 CCAI 将使部署最小化延迟的 CCA 成为可能，这是长期以来的一个挑战。由于我们假定短期内互联网的架构或经济模式不会发生根本变化，因此本文探讨了在与当前互联网经济模式一致的框架内实现 CCAI 的挑战。

尽管有关网络辅助拥塞控制的文献浩如烟海，但目前还没有这样的框架被提出。例如，作为 TCPF 替代方案的两个主要竞争者——每流公平性（即通过公平排队实现[20, 36]）和网络效用最大化（即受凯利工作的启发[32, 33]）——都不符合当前互联网的商业现实。这是因为这两种方法都专注于单个“流”（即力求在流之间实现公平性或最大化流效用之和），但流在互联网的商业协议中没有任何作用（见[10]）；流没有带宽或效用的“权利”，也不是用户收费的单位。

最近，一份正在制定中的 IETF 草案主张通过一种与当前尽力而为架构共存的新互联网架构来提供低延迟、低丢包服务（L4S）[11]。L4S 提议使用传统的公平队列方法在网络中将对延迟敏感的流量与传统的填充缓冲区流量隔离开来。然而，L4S 并未提供完全的拥塞控制算法独立性（CCAI）；相反，它在两类流量之间建立隔离，并为其中一类流量规定了拥塞控制算法（受 DCTCP [2] 的启发）。而通过 RCS，我们更进一步：在与 L4S 相当的部署工作量下，我们提供了完整的 CCAI，从而能够实现最小化延迟的拥塞控制。因此，借助 RCS，不仅能够实现 L4S 的目标，还能提供有原则的带宽分配。

因此，为了应对上述挑战，我们需要一种新的方法，而非新的分组调度机制。也就是说，尽管分层公平队列是正确的隔离机制，但要实现 CCAI，需要根据拓扑结构仔细配置队列权重和分配。我们的主要贡献在于推导出一套原则，以构建与互联网经济模型相一致的权重和队列分配框架；我们将此称为“递归拥塞份额”（RCS）。RCS 受到 [12] 中介绍的方法的启发；遗憾的是，正如第 6.2 节所示，该先前方法无法实现 CCAI。

然而，在探讨这些原则之前，我们先来回答三个关键问题。

什么样的拥塞控制算法（CCA）是“合理的”？ 我们的 CCAI 目标要求，只要拥塞控制算法是合理的，那么一个流的带宽就不应受其自身或其他流所选择的 CCA 的影响。这就需要对“合理”给出定义：我们说一个 CCA 是合理的，如果它在静态环境中能有效利用可用带宽，同时避免持续且严重的丢包。我们提出这一条件是因为它能防止流无谓地损害其他流并导致拥塞崩溃，就像在文献 [40] 中讨论的死包现象那样。避免持续且严重的丢包与我们所知的所有已提出的 CCA 都是兼容的（与 TCPF 不同，TCPF 的限制要严格得多）。基于丢包的 CCA 会持续产生少量丢包（但不是严重丢包），而 BBR 可能偶尔出现严重但不持续的丢包（例如，如果流遇到带宽的快速下降）。实际上，唯一可能出现持续且严重丢包的现实场景是分布式拒绝服务（DDoS）攻击，而运营商已经投入大量资源来预防和缓解此类流量。

在 CCAI 之后，CCA 的作用是什么？CCAI 的一个关键特性在于，由于 CCA 的带宽权利得到了保障，因此 CCA 可以自由地在诸如丢包率、延迟和抖动等其他指标上进行优化。尽管目前设计一种最小化延迟的 CCA 被认为是不切实际的，因为当与不仅包括填充缓冲区的流量，还包括其他最小化延迟的流竞争时，会存在公平性问题[4]，但采用 RCS 将能够采用这些及其他优化算法。此外，虽然 RCS 保障了带宽权利，但它并未规定带宽分配；这使得 CCA 设计者能够解决传统问题，即在寻找可用带宽的同时尽量减少自身造成的延迟，同时消除了公平性方面的顾虑。

RCS 能否部署？虽然我们开发 RCS 的目标是为互联网带宽分配提供一个有原则的概念框架，而非一个可立即部署的机制，但在第 7.3 节中我们论证了 RCS 的原型设计与大规模部署并不冲突。更重要的是，网络运营商及其客户都有采用 RCS 框架的动力；而如今，运营商无法向支付更高费用以获取更高级别服务的客户保证端到端的带宽权利，但借助 RCS，这种保证变得可行。同样，如今想要获得更好的端到端性能的客户可能需要部署具有预留容量的专用广域网，但借助 RCS，他们能够以更低的成本购买其应用所需的带宽权利。我们在第 7.2 节中进一步讨论了采用 RCS 的这些经济激励因素。

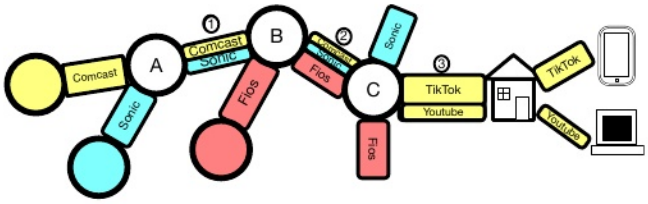


图 1：RCS 原理示意图：（1）相对权利。（2）这些权利应递归应用：队列首先考虑 Fios（红色）和由 Sonic（蓝色）与 Comcast（黄色）组成的聚合体。如果必须从聚合体中丢弃一个数据包，则在上游域 A 考虑 Sonic 和 Comcast 的相对权利。（3）端点在更细的粒度上管理流量控制。在 Comcast 聚合体中，端点优先处理 TikTok 流量而非 Youtube 流量。

3 “一致”的通用人工智能原则

我们的目标是以符合互联网商业安排的方式实现 CCAI，但“符合互联网商业安排”意味着什么却远非显而易见。在此，我们陈述了三条原则，用以描述其内涵。我们在第 4 节中讨论如何实现这些原则。由于这些原则应当指导当前和未来的拥塞管理，所以我们既不将其与当今的流量或网络技术特征相联系，也不对哪些应用占据主导地位做出假设。我们在图 1 中对这些原则进行了说明。

原则 #1：带宽分配应在网络拥塞时执行，并且应以相对权利的形式来描述。这意味着，当网络拥塞时，它会根据数据包的相对权利（下文将解释）来决定丢弃哪些数据包。相比之下，将带宽权利表述为保证速率（即用户可预期的端到端带宽的绝对水平）会极大地降低统计复用，因此是不切实际的。这一条件并不排除 RCP [22] 或 XCP [31]，但排除了 IntServ [39]；前两者仅提供临时估计（基于某种相对权利的概念），而 IntServ 则提供持久保证（并非基于相对权利）。

原则 #2：这些相对权利应与当前的商业安排相挂钩，尊重其粒度、递归性质以及资金流动情况。用户为边缘接入付费，因此现行的商业安排是以这些接入协议的粒度为依据，而非以单一流量为单位。此外，这些接入协议具有递归性（即数据包从一端传送到另一端，是因为发送方的运营商向下一跳运营商付费，下一跳运营商再向后续跳运营商付费，以此类推）。资金同样从接收方的域流向该域的提供商，依此类推。这些跨域安排对于网络如何承载流量至关重要，因此我们认为其递归性质必须在网络管理拥塞方面发挥作用。

原则 #3：虽然网络决定两个端点之间的带宽分配，但端点应决定在它们之间流动的流量的构成。

这一基本要求是明确的，但问题在于应由哪一端来掌控。原则 #2 的指导方针应决定是发送方还是接收方负责控制此流量。决策应遵循资金流向，发送方决定何种流量进入网络（可采用诸如 Bundler [16] 这样的机制），接收方决定何种流量离开网络（可采用诸如 Crab [45] 这样的机制）。在中间情况中，应根据出现拥塞时的资金流向来确定。尤其鉴于“零费率”存在的风险 5 [6, 7, 9]，由接收方而非发送方决定接收何种流量这一点至关重要。

4 从原则到实践

接下来，我们将这些原则转化为一种计算带宽分配的算法。为了便于阐述，我们首先考虑静态环境下的分配情况，在这种环境中，用户以固定速率通过具有固定容量的链路发送数据。我们为 RCS 提出的机制能够处理现实中的动态环境，但很难以一种有条理的方式对这种环境进行推理。在开发出这种带宽分配算法之后，我们接着在第 5 节中探讨 RCS 分配是否实现了 CCAI。肯定地回答这个问题是本文的核心贡献。我们进一步评估了我们的方法与实际 CCA 的交互情况（第 6 节）。然后，我们讨论了（i）一些其他实际问题（第 7 节），（ii）RCS 与相关工作的比较（第 8 节），以及（iii）以一些总结性评论结束（第 9 节）。

4.1 原则 #1：相对权利

我们从三个范围探讨相对权利的影响：链路（指具有多个入口和一个出口的技术）、交换机（具有多个入口和多个出口）以及域（交换机网络）。我们将这三个易于处理的案例作为“构建模块”，以更普遍地思考互联网中的拥塞管理问题。在前两种情况下，拥塞点位于出口处，因为（i）我们假设一个交换机的出口是另一个交换机的入口，并且成对的出口和入口具有相同的容量（因此任何拥塞都会由前一个出口处理，而非入口），以及（ii）我们假设交换机具有全内部带宽。在第三种情况，即域中，拥塞的位置将取决于具体场景。在本节中，我们将假设相对权利源自发送方的接入协议（而非接收方），但在 4.2 节中我们将对此进行一般化处理。

我们使用“用户”这一术语来指代在网络边缘签订接入协议（即提供其一定水平互联网服务的协议）的实体，而“流”则指代从同一入口点进入网络并从同一出口点离开网络的流量集合。因此，一个流就是由一个用户发送并被另一个用户接收的流量。术语“CCA”指的是流对拥塞的响应方式；实际上，这种响应是由几种不同的基于流的拥塞控制算法和应用行为（例如打开更多连接或限制流媒体数据）组成的，但是

3 Ours are not the only possible principles; we offer them as a starting point. A different set of principles might result in both different bandwidth allocations and different competition dynamics, and we leave the analysis of such frameworks to future work.

4 Edge ISPs typically throttle a user's bandwidth on their access line to their contractual rate, regardless of congestion. Our focus here is on congestion internal to the Internet.

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

为方便起见，我们将其建模为单个 CCA。6 之后，对于存在多个入口和出口的情况，例如交换机和域，我们使用术语 AGGREGATE $_{(}$I, α$_{)}$ 来表示从入口 $_{I}$and 进入并在出口 α 离开的流量。

单链路的相对权利。考虑一条链路，有若干流以速率 RI 发送数据。我们用 AI 表示各流离开链路的带宽（即各流的出链速率），其中 $_{RI}$≥$_{AI}$: 严格不等式表示网络丢弃了流 I 的数据包，我们称这样的流为“受约束的”。链路是工作保守型的，因此 $\acute{I}_{IAI}$= $_{MIN}$[$\acute{I}_{IRI,C}$]，其中 $_{C}$is 表示链路的带宽。在此背景下，我们根据权重 WI 定义相对权利，并按这些权重的比例为受约束的流分配带宽。具体而言，对于任意两个流 I, $_{J}$with AI$_{<}$ $_{RI}$and $_{A}$J$_{<}$ R_{J}, 以下关系成立：WAI A = WJ??≥ I

对于所有其他流 K，有 AK WK。等式 $_{I}$and $_{J}$requires 表示两个受约束的流所获得的带宽与其权重成比例。不等式 $_{J}$and $_{K}$requires 表示任何不受约束的流所获得的带宽不会超过其受约束时的带宽。此定义意味着 $_{AI}$= $_{MIN}$[R I, W Iλ]，其中 $_{λ}$≥0 是允许 $\acute{I}_{IAI}$= $_{MIN}$[$\acute{I}$ IRI, C] 的最小值。

这导致了我们所说的“管道式”行为。当一个流增加其带宽需求 RI 时，起初其需求会完全得到满足，然后会被限制在某个最大带宽。这种曲线上的急剧“拐点”使得拥塞控制算法（CCA）很容易找到允许的最大带宽，并且不会奖励那些造成持续丢包的流（即，它们在发送速率超过拐点后不会获得额外的带宽）。

作为对比，如果一个链路不主动管理拥塞，而只是采用先进先出的分组调度方式，那么带宽分配则由以下公式给出： $_{AI}$= min [R I, R $_{If}^{C}$ $_{R}$]，并且平均分组

$$\frac{j}{j}$$

延迟（对所有流来说都相同）是 I JRJ 的某个函数。

当流量达到链路容量时，这一数值会急剧上升。如果我们将所有其他 RJ$_{>0}$ 固定，即使 $_{RI}$and 流量 I 的数据包数量 $_{RI}$is 非常小（例如，如果剩余负载 $\acute{I}_{J\neq IRJ}$is 远远大于链路容量），其也会遭受严重的丢失和延迟。

单个交换机上的相对权值。单链路方法的最小泛化是让每个出口在所有出口 α 上应用单链路定义，权重 $_{WI}$for 每个入口 $_{I}$applied 。这是我们在 RCS 中采用的方法。我们也可以考虑权重是静态的但依赖于每个出口的情况：即从入口 $_{I}$to 到出口 $_{α}$has 的 AGGREGATE $_{(}$I, α$_{)}$ 具有权重 WIα。为简洁起见，我们在此不做这种泛化的处理，但我们的结果同样适用于这种情况。

有人可能会认为，权重应取决于当前的流量矩阵，入口处分配的总权重应在出口处按当前流量分配比例进行分配。例如，假设所有入口和出口的容量均为 $_{C}$= 1，I αWI 对于所有 I，α $_{=}$ 1，权重 $_{Wα}$for 已给定

$$\frac{}{i}$$

$_{I}$are 与相对流量成比例（即，如果一个入口的三分之二流量流向一个出口，则该聚合流量就获得）

考虑这样一个情况：两个入口 I 和 $_{J}$send 的流量流向两个出口 α 和 β。回想一下，合理的CCA会在不出现严重且持续的丢包的情况下最大化带宽；在这个模型的背景下，这意味着它们会选择最大的 $_{RI}$such 值，使得 $_{RI}$= AI。假设入口 $_{I}$sends 将其全部流量（速率为 RI）按比例 $_{X}$ and （1 - X） 分配到出口 $_{α}$while 和 $_{J}$splits，其总速率为 RJ。那么，聚合流量 AGGREGATE（I, α） 的权重为 1，聚合流量 AGGREGATE（J, α） 的权重为 X，聚合流量 AGGREGATE $_{(}$J, β$_{)}$ 的权重为 1 −$_{X}$。入口 $_{J}$does 在出口 $_{α}$are 不会遭受持续丢包的唯一分配选择是：(i) $_{X}$= 1 和 $_{RJ}$= 12（其中 $_{RI}$= 12）以及 （ii） $_{X}$= 0 和 $_{RJ}$= 1（其中 $_{RI}$= 1）。这是因为，一旦入口 $_{J}$dilutes 将其流量分配到两个出口，就会有部分数据包被丢弃。因此，按流量比例分配权重可能会导致病态分配，所以我们排除这种设置权重的方式。

在域内的相对权利。我们不认为拥塞仅发生在域的边缘，但我们确实认为相对权利由用户与其域之间以及域与域之间的接入协议来确定。如果域内没有内部拥塞，那么该域的行为类似于交换机。然而，如果域内确实存在拥塞，那么它应当在所有发生拥塞的内部链路或交换机上执行相对权利（根据出站时的权重 $_{WI}$assigned 来确定）。我们认为大多数域目前以及将来都会进行管理以避免内部拥塞，除了某些特定的热点区域（例如电缆调制解调器终端系统（CMTS）和跨洋链路），因此这种执行措施无需在域内广泛部署。为方便起见，在接下来的内容中，我们假设域内不存在内部拥塞。

依照 [12] 中的定义，我们将这些权重称为 $_{WI}$congestion 分享量，它们是用户与其所属域达成协议的一部分。 $_{WI}$is 的值并非直接与访问带宽相关，但可以推测，访问速度更高的协议通常会有更高的拥塞分享量。

4.2 原则 #2：递归与追踪资金流向

我们将上文所述的在单个域中分配带宽的方法作为基本构建模块，并讨论如何利用第二个原则将其扩展到多个域。这种跨域扩展以及我们证明其在绝大多数情况下足以实现 CCAI 是我们工作的主要贡献。

“递归”是什么意思？仍以用户的数据包进入其入口网络（称其为域 A）时在入口处分配权重的情况为例，当这些数据包离开 A 域（通过 A 域的某个出口链路）时，其相对权利应由用户与 A 域之间的访问协议来决定。当这些数据包通过链路 L 从域 A 传输到域 B 时，7 为了一阶近似，这些数据包离开 B 域时的相对权利应由 A 域和 B 域在链路 L 上的访问协议来决定。虽然 B 域决定丢弃 A 域多少数据包是由 A 域和 B 域之间的访问协议驱动的，但当 B 域决定丢弃 A 域哪些数据包时，该决定应由 A 域与该流量相关联的用户之间的各种访问协议所推导出的相对权利来驱动（如图 1 所示）。这就是

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

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

我们所说的访问协议的递归性是指：（i）数据包在网络中传输时，我们根据其当前所在域的协议递归分配相对权利（由于本示例中假设不存在内部拥塞，这些权利仅在域出口处相关），（ii）我们以分层方式应用这些权利，首先应用其当前权利来决定总带宽，然后再考虑其先前的权利。因此，RCS 为每个流关联一个访问权利的层级结构。我们通过调度数据包来强制执行相对权利（见第 6 节），所以如果一个流超出其带宽分配，其队列最终会溢出，其数据包将被丢弃。

“追踪资金流向”意味着什么？在互联网中，资金通常从终端用户流向域名，再流向提供更广泛路由覆盖的其他域名。资金的这种流动最终会到达相互免费对等互联的一级网络提供商。典型的路由路径是沿着这个域名层级结构向上（从客户到提供商）然后向下（从提供商到客户）进行。因此，对于两个终端用户之间的流量，会存在一个点，在此点上数据包从顺着资金流向（向上层级）转变为逆着资金流向（向下层级）。我们利用这一观察结果来定义两条规则，以控制相对权重如何引导带宽共享。我们首先考虑一个域名与终端用户之间的关系，然后解释这两条规则如何同样适用于域名之间的流量。

第一条规则是，与原则 #3 保持一致，在从域到其用户（即聚合的目的地）的出口处，域应根据与该用户的商业协议推导出用户的相对权利。例如，考虑来自两个不同内容提供商的两个媒体流，其总带宽大于域到用户的出口带宽；用户应有权分配权重，以表达这些流的相对权利。

类似地，第二条规则是，当终端用户的流量进入一个域（即聚合体的源）时，其与该域的商业协议应决定其在该域出口处的相对权利。也就是说，当两个用户的流量超过该域出口链路的容量时，相对权重由发送用户的权重决定。需要注意的是，在聚合体既从终端用户进入一个域又从该域出口到另一个终端用户的情况下，此规则与第一条规则相冲突。在这种情况下，适用第一条规则；即接收方决定聚合体的相对权利。这种接收方优先的裁决方式很重要，因为它可以防止零费率，即一家公司向提供商域付费，以将其流量优先传递给用户，而排除其他流量。

当递归应用时会产生怎样的分配结果？鉴于进入和离开互联网的数据包的这两个特定规则遵循了资金流动规律，我们现在寻求递归应用这种方法。尽管当今的域间协议比高-雷克斯福德模型[26]要复杂得多，但我们认为以下两个陈述在绝大多数情况下都是成立的：（i）对于域 A 和 B 之间的特定逻辑链路，要么 A 向 B 支付费用，要么 B 向 A 支付费用，要么双方都不支付；（ii）互联网路径上的支付结构在高和雷克斯福德所描述的意义上是“无谷”的。因此，使用“客户/对等方/提供商”这一术语来指代给定链路上的资金流动，我们可知，在无谷路径的情况下，有以下两个事实成立。首先，如果出站 $\alpha$is 至客户，那么所有后续的跳转都是至客户（并且它们决定了在该出站处分配给聚合体的所有层级权重）。其次，如果出站 $\alpha$is 至对等方或提供商，那么所有之前的跳转都是来自客户（并且它们决定了在该出站处分配给聚合体的所有层级权重）。因此，所有在出口处的聚合体 αhave 都以其在相同方向（要么是前向跳数，要么是后向跳数）确定的权重进行计算。由此产生的带宽分配是相对权重递归应用的结果。

定义 WI，α 为分配给 AGGREGATE (I, α) 的权重。以权重由前序入口递归分配到出口 αwere 的情况为例，计算过程如下：首先计算所有入口 I 分配的权重下每个 AGGREGATE (I, α) 的分配量，从而得到一组分配量 AI。然后，对于每个 AGGREGATE (I, α)，计算所有通过入口 I(with（即其前序入口分配的权重）进入的聚合体的分配量，将总带宽视为 AI。此过程沿着域的层次结构一直递归到最终用户。

我们将这种分配方式称为分层加权公平共享（HWFS），它与文献中各种分层加权公平队列算法[8, 44]在给定相同权重集时所实现的静态分配完全相同。本工作的创新之处在于权重的配置；我们建议根据其余的拓扑结构和访问协议来计算这些权重。

4.3 原则 #3：端点控制

该原则指出，虽然网络决定分配给每个流的带宽，但端点（包括入口端和出口端）应决定该流的构成。例如，尽管根据大学网络之间的接入协议，相对权利应决定两个大学网络之间的总带宽分配，但这些网络可能希望优先处理研究数据的批量传输，而非视频流传输。RCS 对此优先级排序未作任何说明，只是指出端点域应对其进行控制。当然，端点也可以决定采用默认的先进先出策略，在这种情况下，端点流中最激进的拥塞控制算法会占用更多的带宽分配。不过，RCS 会阻止这些拥塞控制算法影响其他聚合体可用的带宽。

要理解如何实现这一点，区分流在遇到拥塞时可能出现的三种情况是有用的。第一种情况是在用户进入网络时，用户可以使用其自身的内部机制来控制流的构成。第二种情况是在网络内部的某个位置，例如两个域之间的出站链路上。关于 Bundler [16] 和 Crab [45] 的最新研究提供了一种机制，使用户能够远程控制流的内部构成（我们在附录 C 中提供了更多关于 Bundler 和 Crab 如何实现这一点的细节）。第三种情况是在端点域向目标用户出站时；这是前一种情况的特殊情况，可以应用相同的机制。请注意，与前面的原则一致，如果在发送方权重适用时发生拥塞，则由发送方用户确定构成，反之亦然。

4.4 我们的工作与[12]有何关联?

本文的灵感来源于文献[12]，该文献提出了递归拥塞份额的基本思想。然而，文献[12]中的详细方法存在两个主要局限性。首先，文献[12]仅应用了一层权重：即由流量最近经过的入口分配的权重。对于直接从一个域流向另一个域的用户流量（确切地说，流量从用户X流向域A再流向域B的情况），这已经足够。然而，这并不能涵盖流量从用户X和Y流向域A，再流向域B，最后流向用户Z的情况。如果拥塞发生在域B的出口处，那么中转提供商就无法区分属于用户X和Y的流，从而导致竞争更激烈的CCA用户获得更多的带宽。因此，文献[12]中的版本甚至对于一些相对较短的路径也无法实现CCA公平性。

第二个局限性在于，[12] 中的设计忽略了接收方的作用，仅考虑了入口点分配的权重。这引发了公平性问题（即，内容提供商是否应该决定我网络接入线路的优先级？），并且还意味着分配方案在路径的下行部分并不遵循资金流动。这两个缺陷意味着需要提出这里更为复杂但功能更强大的方案。

5 RCS 达到 CCAI 标准了吗?

上述原则确保了 RCS 与互联网的经济模式相一致。接下来，我们探讨 RCS 是否实现了我们的 CCAI 目标。在本节中，我们将从博弈论的角度来回答这个问题，其中用户试图在不造成持续损失这一合理约束条件下，各自优化其吞吐量。我们通过两种方式来分析此博弈：（1）采用混合整数线性规划（MILP）模型来确定是否存在多个博弈论均衡；（2）通过模拟博弈论动态来确定合理的 CCA 是否会收敛到这些均衡。

5.1 恰到好处的游戏理论

在我们的模型中，控制流的各个 CCA（拥塞控制算法）是游戏中的“玩家”，我们假设它们的行为是理性的（即，追求效用最大化）。游戏的定义由网络拓扑结构和带宽分配规则（即 RCS）决定，这些规则在给定一组输入速率 RI 的情况下，确定了最终的吞吐量速率（如前所述的符号表示）。玩家的策略是其发送速率，其收益要么是其发送速率（如果其输出速率与发送速率相同），要么是 -∞（如果其输出速率小于发送速率）：这一条件仅仅是为了确保不存在导致持续丢包的均衡状态，并非意在反映现实中的收益情况。因此，我们提出问题：这些游戏会收敛到何种均衡状态？为回答这个问题，我们运用了博弈论中的两个概念。

第一个概念是大家熟悉的纳什均衡[25]。当所有参与者（即 CCA）都在采用其最优策略（即发送速率），且假定其他所有参与者已选定其策略并保持不变时，就达到了纳什均衡。那也就是说，对于每个 I，$_{RI}$is 为最大值，使得 $_{AI}$= RI，假设除 $_{I}$keep 之外所有玩家的发送速率固定。这是一个均衡状态，在此状态下，任何玩家单方面偏离均衡都无法获益。

第二个概念是不太为人熟知的斯塔克尔伯格均衡[25]。在这个模型中，“领导者”（即某个 CCA）首先承诺采取某种策略（即发送速率）。游戏中的其他“跟随者”CCA 观察到这一领导者的行动并做出反应，在由领导者策略固定下来所形成的游戏子集中达到纳什均衡。当领导者 $_{I}$has 选择了其最优策略（即选择了使 $_{RI}$such 取得最大值的策略，假设在每种情况下，其他参与者的集合都会在子游戏中针对领导者的策略达到纳什均衡）时，就会出现斯塔克尔伯格均衡（假定只有一个领导者 I）。通俗地说，我们可将领导者视为测试其每种策略，观察其他参与者会收敛到何处，并选择能带来最佳结果的策略。

对于给定的一个博弈，如果只存在一个纳什均衡且该均衡也是唯一的斯塔克尔伯格均衡，那么无论每个参与者有多“激进”，该博弈都恰好存在一个稳定的均衡。然而，如果存在与纳什均衡不同的斯塔克尔伯格均衡，或者存在多个纳什均衡（这意味着必然存在多个斯塔克尔伯格均衡），那么激进的参与者（即斯塔克尔伯格领导者）可以试图操纵博弈以实现使其吞吐量最大化的结果（我们在附录 B 中展示了这种情况成立的示例拓扑结构）。我们在第 5.3 节中表明，这种操纵很少能为领导者带来更好的均衡结果。此外，这种操纵在实际中也难以实现，因为斯塔克尔伯格领导者要么必须无所不知（以便能够计算出其最优策略），要么必须在很长的时间段内对其行为的反应进行采样（以便其他参与者能够收敛到一个均衡），然后才能寻找其最优策略。

5.2 贪婪有回报吗?

为了弄清楚多重纳什均衡或非纳什斯塔克尔伯格均衡是否常见，我们考虑了两个模型。第一个模型基于随机拓扑结构，我们设计该模型旨在增加出现病态情况的可能性：每条流都随机选择路径，并在路径上的每个入口处使用随机权重。在这个模型中没有聚合（每个路由器对每条流单独处理），因此每个路由器仅对其所处理的每条流应用自身的权重。由于流从不聚合，所以无需参考之前的跳数。我们从一个完全连接的 10 节点拓扑结构开始，然后通过随机选择节点序列生成 40 条 4 跳流。

第二种模型使用从 CAIDA AS 关系数据集 [15] 中子采样的拓扑结构。我们在数据集中随机选取一个自治系统（AS）作为初始流的起点。然后，以 0.7 的概率将此流的下一跳（或前一跳——我们在两个方向上扩展流）设置为该 AS 的一个邻居，同时保持符合 Gao-Rexford 的路径，否则在邻居处终止该流。当我们将一个新的 AS 添加到流的路径中时，以 0.5 的概率在该 AS 处生成一个新的流。我们使用相同的规则扩展此流。我们停止扩展拓扑结构

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

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

一旦我们生成了 40 条流。由于 CAIDA AS 关系数据集不包含容量或权重信息，我们为每条链路随机生成这些信息。

我们使用一款商业混合整数线性规划求解器 [37] 来计算两种模型下的分配情况。我们不允许流出现持续亏损的情况。

随机模型。混合整数线性规划（MILP）首先定义分配变量，其中 $_{AS}$denotes 分配使流 ⇔ 达到均衡。从给定的场景出发，我们定义一组链路 $_{L}$with 容量 CL、$_{S}$as 所有流的完整集合、$_{SL}$as 每条链路 L 上经过的流的集合，以及一组权重，使得 WL，S 表示流 $_{S}$at 链路 L 的权重。然后，我们创建瓶颈指示变量 BL，S，使得 BL，S == 1 当且仅当流 $_{S}$is 在链路 L 处出现瓶颈。最后，我们为链路创建一个指示变量，使得 $_{CL}$== 1 表示 $_{L}$is 达到容量。然后我们定义以下约束条件：

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

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

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

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

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

约束条件（1）确保任何流都不会在其路径之外的链路上出现瓶颈。约束条件（2）确保每个流在每个拥塞链路上都能获得其加权公平份额（M 是某个较大的常数）。约束条件（3）确保每个流在恰好一个链路上出现瓶颈。最后，约束条件（4）确保没有链路出现超额预订。需要指出的是，我们使用此混合整数线性规划（MILP）公式来模拟 RCS 下的带宽分配情况，而非对现状进行建模。具体而言，由于 RCS 的“管道式”行为规范（第 4.1 节），瞬态交互不会决定 RCS 中的带宽分配；因此，在此分析中我们未对其进行建模。在我们的仿真实验中评估部分部署场景时（第 6.2 节），我们会考虑瞬态交互。

为了确保我们能全面描述给定流的所有纳什均衡，我们轮流对每个流进行优化 $_{S}$to 确定在这些约束条件下该流的最大和最小分配量。因此，对于给定的拓扑结构，纳什解为每个流提供了两个均衡（分别对应最大值和最小值），不过在大多数情况下，这两个值是相同的。要生成以流 $_{S}$as 为领导者的斯塔克尔伯格解，我们进行优化 $_{AS}$and 允许 $_{S}$to 在没有链路或单个链路发生瓶颈的情况下，将约束条件（3）替换为约束条件（5）。

我们在表 1 的第一行展示了结果，其中很少有拓扑结构存在有问题的均衡。当单独考察各个流时，结果甚至更有力。在所有场景中总共 748,920 个流中，只有 68 个（0.009%）流从激进的斯塔克尔伯格领导者策略中获益。任何流试图通过采用斯塔克尔伯格策略来改善自身所获得的平均百分比收益仅为 0.011%。因此，在这个模型中，贪婪并不能带来（太多）回报。

拓扑学	总计	多种；多个；多重；斯塔克尔伯格	非纳什均衡
随机的	一万八千七百三十八 (1.81%)	338 (1.81%)	598 (3.19%)
CAIDA 抽样	两千八百九十七	0	0

表 1：不同拓扑结构下的均衡类型。我们从已发表的有关自治系统（AS）关系的数据[15]中推导出 CAIDA 抽样拓扑结构，并根据生成 AS 图的启发式方法（§5.2）推导出随机拓扑结构。结果表明，在采用 RCS 的情况下，对于随机拓扑结构，多重纳什均衡和非纳什斯塔克尔伯格均衡都很少见，而对于 CAIDA 抽样拓扑结构，这两种均衡出现的频率则更低。

CAIDA 抽样模型。对于 CAIDA 抽样模型，在我们的混合整数线性规划（MILP）计算中，域在入口处为聚合体分配权重。两个域之间的出口链路反映这些协议。因此，在此 MILP 计算中，我们仅假设所有流量都遵循资金流动来建模权重（但在第 5.3 节和第 6.2 节中考虑更一般的情况）。为了实现 CAIDA 模型，我们在上述描述的基础上添加了以下约束条件：

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

此附加约束强制执行递归加权分配。在层次结构的每个深度 $_{D}$of（其中 $_{D}$= 0 为根节点）对于来自扁平模型的所有流 $_{S}$ in $_{SL}$(borrowed，我们定义 AGGL，S 为包含流 $_{S}$at 链接 L 的聚合，以及 WAGG，L 为分配给聚合 AGG 的权重。最后，我们定义 $_{DL}$to 为链接 L 的层次结构的最大深度，AGGSL，D 为链接 L 在层次结构深度 $_{D}$of 的聚合集，以及 $_{UI}$to 为聚合 I 包含的流集。

我们在表 1 的第二行展示了结果。我们发现，在 CAIDA 抽样拓扑中不存在多个纳什均衡或非纳什斯塔克尔伯格均衡，这意味着采用斯塔克尔伯格策略对任何发送方都没有益处。

5.3 合理流量会受益吗?

混合整数线性规划（MILP）模型表明，在绝大多数情况下，只存在一个纳什均衡。现在的问题是，那些目光短浅、只根据当前情况调整自身行为的碳捕获与封存（CCA）机构能否达到这一唯一的纳什均衡?

为了模拟短视博弈论动态，我们实现了一个“最佳回应”代理，它在最小和最大发送速率之间迭代多个发送速率，并计算每个速率下的效用。在达到的最佳发送速率周围的两个发送速率被用作下一次迭代的最小和最大值：请注意，最佳发送速率是在不出现丢包的情况下能达到的最高速率。当最大速率和最小速率之间的差值小于一个小的 时，发送方终止操作。

我们使用了上述的 CAIDA 拓扑生成器，并从 1030 个 8 流拓扑中进行抽样，其中考虑了仅发送方权重（606 个）和发送方与接收方权重（424 个）。我们为拓扑中的主机分配了 CCA（使得每种 CCA 至少出现一次），并进行了多次迭代，轮换 CCA 分配。在所有这些拓扑中，代理都收敛到了一个单一的均衡状态。

我们还研究了在更贴近实际的动态情况下这一结论是否成立。我们实现了几种传统拥塞控制算法（CBR、AIMD、BBR、拥塞控制、延迟敏感型）的简化版本，并规定如果一个流的发送速率进入一个重复循环（该循环因拥塞控制算法的不同而有所差异），则认为该流处于平衡状态。同样，在 1052 种拓扑结构中，智能体都收敛到了一个单一的平衡状态。

5.4 总结

这些结果表明，在大多数情况下，RCS 确实足以实现 CCAI，而且试图利用 RCS 的收益甚微（即，平均收益极低，即便 CCA 是全知的）。是什么导致了这种程度的有效性？在单个节点实现 CCAI，使用分层加权公平队列（无论是应用于单个流还是聚合流）是必要的，但显然这还不够，正如我们在第 4.4 节的简单示例中所展示的那样。关键在于给定出口处分层树中的权重是基于该出口上游或下游的对等协议集。

6 实施与评估

接下来，我们通过数据包仿真来确定 RCS 的带宽分配实现是否为实际的 CCA 实现提供了 CCAI。首先，我们描述了一种可以实现 HWFS 的调度算法的实现，即分层差额加权轮询（HDWRR）。然后，我们展示了使用 Mininet [29] 进行的数据包仿真结果，其中使用了三种具有不同激进程度的 CCA：Reno、Cubic 和 BBR。我们比较了以下几种情况的结果：（i）上文第 5.3 节中描述的理想化最佳响应分配，（ii）HDWRR，（iii）使用差额轮询（DRR）算法的每流公平性，以及（iv）先进先出（FIFO）调度。

6.1 分层赤字加权轮询

为了评估 RCS 在实际 CCA 实现中的分配情况，我们首先必须实现一种能够实现这些分配的机制。为此，我们使用了一种基于公平队列调度器的缺陷轮询（DRR）[41] 实现的调度算法 HDWRR。我们在附录 A 的清单 1 中展示了 HDWRR 的伪代码实现。与 DRR 不同，DRR 是常量时间复杂度，而 HDWRR 的复杂度会随着权重树的深度而增加。此外，虽然 DRR 总能为给定队列分配其全部配额，但 HDWRR 必须跟踪树中非叶节点剩余缺陷不足以容纳其下一个子节点的全部配额的情况。在这种情况下，我们的实现会将剩余缺陷分配给子节点，但会跟踪下一轮的剩余调度缺陷。

6.2 Mininet 模拟

为了评估我们的 HDWRR 实现及其与真实 CCA 的交互情况，我们用约 150 行 Rust 代码在用户空间中实现了一个原型 HDWRR，作为 TUN/TAP 设备。完整的实现，包括可选的 DRR 和 FIFO 调度器，约有 1500 行 Rust 代码。

我们考虑了四种实验配置：（1）FIFO，代表当前状态；（2）DRR，按流而非按聚合分配，类似于公平队列；（3）单跳，使用 HDWRR 但将权重树限制为深度 1（如 [12] 所述）；（4）HDWRR，即上文所述的我们的实现。

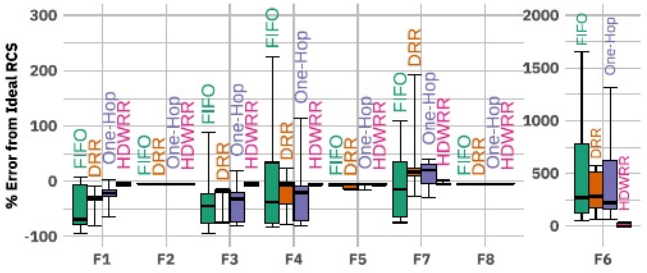


图 2：在具有真实通信信道适配器（CCA）的仿真拓扑结构中实现的分配情况。完美方案对于所有流的误差均为 0%。本文中的所有箱线图均显示百分位数（p5、p25、p50、p75、p95）。请注意 F6 的 y 轴不同。

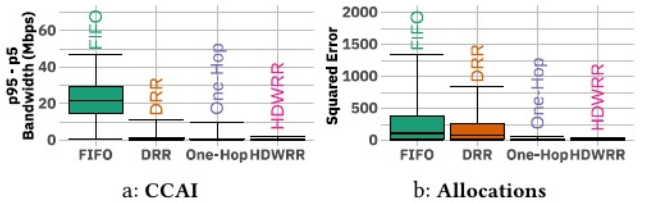


图 3：（a）展示了每种方案下第 5 百分位数和第 95 百分位数所接收带宽的差值。差值越小，表示 CCA 选择对所接收带宽的影响越小。（b）展示了在该方案下各流所接收的分配量与 RCS 下预期分配量之间的平方误差。

我们从两个方面对这四种配置进行评估。首先，它们是否提供 CCAI，即所有流是否都能获得一致的带宽，而不论其（或其他流）使用何种 CCA？其次，带宽分配是否与经济关系相符？我们通过将各流实际获得的带宽归一化到我们的模拟器（§5.3）计算出的份额来表示分配是否与经济关系一致。为评估这些问题，我们使用了在 §5.2 中描述的相同的 CAIDA 抽样拓扑生成器。我们生成了 8 流和 15 流的拓扑结构，并且对于每个拓扑结构，我们进一步生成 10 个随机的 CCA 分配，其中为每个流随机分配 Reno、Cubic 或 BBR 作为其 CCA。我们展示了单个拓扑结构中所有流的结果（图 2），以及在我们生成的 10 个 CAIDA 抽样拓扑结构中的分布情况。虽然我们总结了各拓扑结构的统计数据，但我们注意到 CCAI 和带宽分配不均在最坏情况下影响最大，因为如果拓扑结构的任何部分不支持 CCAI 或经济上一致的带宽分配，那么 CCA 就必须针对这种最坏情况来悲观地设计。

先进先出（FIFO）。不出所料，采用先进先出调度既不能提供符合 CCAI 的带宽分配，也不能提供符合 RCS 的带宽分配。考虑图 2 中的流 F1、F3、F4、F7 和 F6：这些流获得的带宽最多比 RCS 分配低 100%（即饥饿状态），而 F6 获得的带宽最多比 RCS 分配高 15 倍。同样，在各种拓扑结构（图 3）中，先进先出调度既不能提供一致的带宽分配（p5 和 p95 带宽之间相差 28 Mbps，左图），也不能提供经济上一致的分配，相对于 RCS 分配的平方误差较大。

DRR 和单层分层。传统的公平队列以及单层分层（仅增加一层分层）能够提供 CCAI。

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

图 4：在 RCS 中，突发型聚合体与非突发型聚合体获得的带宽分配量相似。

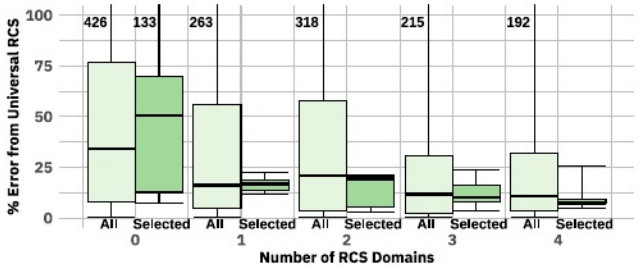


图 5：x 轴表示在给定流路径上运行 RCS 的出口数量。我们展示了与全网部署 RCS 的流量分配相比所达到的流量分配的错误率。顶部的数字为异常值 p95 错误。

但两者都会导致不符合 RCS 规范的带宽分配。实际上，图 3 显示，DRR 和 One-Hop 的带宽分配差异远小于 FIFO；它们的带宽分配在不同拥塞控制算法的选择下保持一致。不幸的是，DRR 和 One-Hop 的分配结果可能与 RCS 相去甚远；在图 2 中，流 F1、F3、F4、F6 和 F7 的带宽分配严重不足或过度分配。

HDWRR 是唯一一种经评估的调度算法，能够同时实现 CCAI 和与 RCS 相一致的分配。从图 2 中我们可以看出，所有流的分配都几乎与理想的 RCS 分配相同，而且在不同的拓扑结构中，带宽分布和平方误差都很低。

6.3 突发流量

到目前为止，我们一直专注于使用传统的 CCA 实现来评估各种场景。在本小节中，我们考虑聚合体是否可以通过改变其流量的突发性来影响所获得的分配。我们同时考虑“过度”和“不足”两种突发性形式：“过度”突发的聚合体在短时间内发送的流量超过预期的 RCS 分配，“不足”突发的聚合体发送的流量低于预期分配。我们进行了一项实验，在该实验中，“过度”突发和“不足”突发的聚合体与使用 TCP 立方体的控制聚合体进行竞争。

我们在一个简单的拓扑结构上运行了 RCS，其中两个聚合体（另一个聚合体的权重为 TCP 聚合体的一半）汇入一个共享的瓶颈。我们将一个聚合体固定为使用 Cubic 作为其 CCA，并考虑另一个聚合体的三种配置：“常规”，即另一个聚合体也使用 Cubic，以及上述的突发超量和突发减量。我们将固定聚合体的权重设定为突发聚合体的两倍。我们通过分析来自互联网核心路由器的包跟踪数据生成突发流量，生成的短流大小以达到目标平均发送速率。对于突发超量，我们将此流量生成器配置为平均发送速率比理想 RCS 分配高出 50%，对于突发减量，则配置为比理想 RCS 分配低 50%。在这两种情况下，都存在网络瞬时负载高于和低于理想 RCS 分配的时间段。对于每种突发性配置，我们通过 20 次实验迭代来测量每个聚合体分配与理想 RCS 分配的百分比偏差。

当然，聚合流量可以在任何时候通过减少发送的数据量来获得少于其理想 RCS 分配的带宽，而我们的工作保护调度实现会将此带宽分配给其他流量。相反，在图 4 中，我们评估了两个因素：首先，突发流量的聚合是否可以通过突发流量获得比其 RCS 分配更多的带宽；其次，突发流量的聚合是否能够通过突发流量迫使其他聚合体的资源使用量低于其 RCS 分配值。我们发现这两个问题的答案都是否定的。在“突发超额”情况下，与理想 RCS 的偏差百分比仍低于 2%，这也低于无突发性情况下的 p95 偏差。同样，突发性对其他聚合体带宽分配的影响也不大；在“突发不足”情况下，我们观察到的非突发聚合体的 p5 偏差比理想 RCS 分配低 16%，这与无突发性情况下的 p5 偏差（比理想 RCS 分配低 12%）相当。

6.4 渐进式采用

RCS 能否在无需全面普及的情况下发挥其优势？在最普遍的情况下，答案是否定的：网络中任意相互竞争的流量都可能在其他地方压制单个域的 RCS 流量。不过，我们发现了一种常见的情形，在这种情形下逐步采用 RCS 既有益又有效：沿单一路径达成双边商业协议。例如，如果某个域的提供商存在上游拥塞，那么这两个域可以相互使用 RCS 来告知对方各自流量的相对优先级。在这种情况下，尽管这两个域的流量仍可能受到其他未参与域流量的干扰，但它们会尊重彼此流量的相对优先级。

在图 5 中，我们展示了对这种增量部署类型的评估。我们生成了一个 12 流拓扑结构，并首先在没有参与的 RCS 域的情况下评估流的分配（最左边的一组箱线图）。然后，我们随机选择一条跨域路径长度至少为 4 的流；请注意，对路径长度设置下限代表了 RCS 增量部署前景的最坏情况，因为路径越长，出现竞争的可能性就越大。沿着所选流的路径，我们逐步将该路径上每个域的出站流量切换到 RCS，以观察 RCS 的带宽分配规则，并测量每次切换后的分配情况。在每种情况下，我们都会在 5 个 30 秒的迭代中测量测试流以及拓扑结构中所有流的分配偏差，偏差越小表明 RCS 对增量部署场景具有更强的适应性。我们定义并报告给定流和拓扑结构（由深度参数化）的分配百分比误差，即该流在给定拓扑结构中所获分配与在所有出口均运行 RCS 的拓扑结构中所获分配之间的绝对百分比差异。在我们为此实验所采样的拓扑结构中，该流的路径长度为 4，但我们注意到使用 4 个 RCS

这并非是全面部署：还有 17 个非 RCS 域的流量可能会干扰正在测试的流。

我们发现，即使仅在一个域强制执行 RCS，被测试流的分配偏差也会从 51% 的中位数大幅降低至 17%。当所有四个域都强制执行 RCS 时，偏差进一步降至 7%，但其他流也从中受益，其分配偏差的中位数从原来的 34% 降至 11%。然而，当我们考虑所有流，而不仅仅是被测试流时，从全局 RCS 部署来看，分配偏差要高得多——即使四个域都强制执行 RCS 分配，5 次迭代中的 p95 偏差仍高达 192%。当然，许多导致偏差的流在其路径中并未遇到强制执行 RCS 的域，因此我们不期望这些流获得符合 RCS 的分配。不过，对于被测试流而言，我们的实验表明，逐步采用 RCS 确实会收敛到 RCS 的分配结果。

7 实际问题

7.1 附加机制

RCS 信令。要实现 HWFS，每个出口都需要了解数据包所属的聚合层级以及与这些聚合层级相关的权重。提供这种状态需要（i）在同一个域内的入口和出口之间传输信息的协议，以及（ii）在域的出口与相连域的入口之间传递信息。对于第一个任务，有许多可能的实现方式，但或许最直接的方法是在域内的每个入口和每个出口之间建立持续的信令连接，并且它们定期（双向）交换有关前缀及其相关权重层级的信息。对于第二个任务，只需要将从域内信令接收到的信息摘要转发给相连的域即可。

RCS 端点控制。回想 RCS 的第三条原则：虽然网络应确定流的总带宽分配，但该流的端点应确定哪些单独的流使用该带宽。支持更复杂的流分配策略存在实现上的挑战：流的瓶颈（即出现拥塞的出站链路）很可能不在其控制范围内，而这种控制必须在该范围内实施。

为应对这一挑战，我们采用了 Bundler [16] 和 Crab [45] 中所述的方法，将拥塞（以及因此产生的数据包队列积压）从网络中的出口转移到了适当端点处的链路上（详见附录 C）。

复杂情况。如前所述，对于域内的拥塞，拥塞链路可以采用某种形式的硬件流量控制，使用上述提议的信令机制。我们认为，域将继续被管理，以使内部拥塞情况很少出现，除非是特定的汇聚点，而这些汇聚点可以由域安排纳入其内部信令中。

到目前为止，我们的描述都假定是通过专用双边链路建立直接的对等关系。许多域通过互联网交换点（IXP）进行对等，这些交换点共用一个基础架构，即一个域的接入线路在多个对等关系中共享。然而，这种通过 IXP 建立的对等安排几乎总是免费的，这意味着权重由接收域设定，并且 IXP 可以使用上述相同的信令信息在通往接收域的出口处强制执行这些权重。

7.2 政策与激励措施

虽然这项工作没有引发任何伦理问题，但它确实引发了有关政策和激励措施的疑问。

激励措施。RCS 的设计旨在让带宽分配与资金投入挂钩。这正是促使互联网服务提供商部署 RCS 的关键所在。从纯粹的本地层面来看，一个域可以从内部开始实施 RCS，根据其入口点的协议来处理出口点的拥塞情况；当出现拥塞时，拥塞份额较大的一方将获得更多的带宽。这直接为这些接入协议增添了额外价值，并可能带来更高的收入和相对于其他提供商的竞争优势。然后，在双边基础上，两个相连的域可以达成协议，相互尊重对方的拥塞份额；这同样为彼此提供了增值服务。拥有此类安排的域可以宣称，购买更高的拥塞份额不仅在其域内有益，而且在下游的下一个域中也能获益。以此类推，这种逻辑反复应用，就会产生显著的激励措施，从而推动 RCS 的部署。这与当今的互联网形成了鲜明对比，在当今的互联网中，域名仅能控制其客户的本地流量，因此客户常常被迫以极高的成本建立完全私有的广域网（或者，如果无法做到这一点，就只能忍受网络拥堵）。

网络中立性。关于网络中立性的定义有很多，区分这些定义很重要。以下是网络中立性三种不同的表述方式，我们自创了一些术语来标注各种观点。当然，我们的定义过于简化，但它们抓住了彼此之间的核心差异。我们按照严格程度从高到低依次列出：

- 机制中立性：这一定义侧重于数据包处理机制，排除了这些机制中各种形式的歧视。
- 支付中立性：这一定义同样侧重于限制数据包处理机制中的歧视行为，但明确允许基于客户为服务支付的费用进行区别对待。这其中包括付费获取更高优先级的优先级方案。
- 竞争中立性：米斯拉对网络中立互联网的构想：“一个平台，在这个平台上，互联网服务提供商（ISP）既不通过定价也不通过服务质量（QoS）为特定的应用程序/服务提供竞争优势”[7, 34]。这里关注的重点不是机制，而是歧视对生态系统的影响。如第 3 节所讨论的，零费率就是这种（以及前两种）网络中立性形式的违反，对竞争对手流量的封锁也是如此。

RCS 显然符合网络中立性的第二和第三种定义，但不符合第一种定义，因为虽然它允许支付水平决定数据包的处理方式，但没有其他因素影响数据包的处理（因此，特定的应用程序/服务不存在竞争优势）。我们认为这是一个合理的折衷方案，鉴于网络接入带宽已经取决于支付水平，我们只是允许利用这一信息。

为说明网络内部如何处理拥塞情况。我们承认，如果网络按照所付费用的比例分配拥塞份额，将会存在严重的公平性问题。然而，RCS 机制并不需要如此严格的按比例分配政策。相反，RCS 机制赋予运营商灵活性，以限制拥塞份额的最小值和最大值，从而确保基本的接入水平，并对具有更强商业实力的客户分配资源设置上限。我们注意到[1]将网络中立性的概念进行了拓展，以涵盖各种形式的公平性，我们认为这是一个有趣的研究课题，但超出了本文的研究范围。

7.3 部署与可扩展性

我们原型 HDWRR 实现（第 6.2 节）是软件形式的，当然这不适合互联网规模的部署。部署 HWFS 有两个相关的技术考量：每个数据包所需的运算次数以及所需的路由器队列数量。实现 HWFS 需要（如清单 1 所示）每个数据包进行 O（D）次运算，其中 $_D$is 是权重树的深度。先前的工作 [3] 表明 $_D$is 通常 ⩽3。因此，所需的计算复杂度较低，与现代路由器硬件兼容。

就队列而言，从最纯粹的形式来看，RCS 规定权重树应包含与互联网存在商业关系的每个上游（或下游，对于由接收方指定的权重而言）实体的权重。当然，如今这样的实体已经多达数百万个。

这在当前硬件条件下还无法实现，但在不久的将来或许能够做到。一家路由器供应商私下告诉我们，他们很快就会推出支持近 50 万个队列的产品，并且具有大约 7 到 8 层的层级结构。鉴于目前估计任意两个随机目的地之间的平均自治系统路径长度为 4 跳[14]，以及互联网扁平化的趋势[30]，我们认为 7 到 8 层的层级结构已经绰绰有余。

不过，如果我们能够降低 RCS 的状态要求，使其能够在当前硬件上部署，那就更好了。为此，我们提出了两种可能的近似方法，以减少任何单个路由器需要维护的状态量。我们注意到，我们尚未在互联网规模上对这些近似方法进行评估（而且我们认为，在没有当前流量模式数据的情况下，尝试这样做也不切实际）；因此，我们将互联网规模的 RCS 实现设计留待未来的工作。

将调度移至状态层面。我们的原型在每个拥塞链路上强制执行带宽分配，并在整个拥塞份额树上执行。另一种方法是将某些聚合子树的调度工作卸载到上游/下游路由器（取决于资金流向）。当然，这些上游/下游路由器并非这些聚合的自然瓶颈，因此在能够对其子树执行带宽分配之前，必须先通知其聚合带宽分配情况。我们注意到，我们用于端点控制的系统（即 Bundler [16] 和 Crab [45]）正是执行此功能，通过在聚合上使用 CCA 来隐式（或显式 [22, 31]）地传达此速率。由于带宽分配随时间变化，而聚合上使用的 CCA 只能发现带宽的延迟

近似值，因此带宽分配只能近似理想 RCS 计算。

动态状态分配。由于只有受约束的聚合体在超出其限制时才需要丢弃数据包（不受约束的聚合体在瓶颈处才会丢包），因此可以使用与受约束聚合体数量成比例的状态量来实现近似于 HWFS 的功能，而不是使用与总聚合体数量成比例的状态量。这样的算法需要根据聚合体速率的变化动态调整所保存的状态，因此在调整状态以反映当前使用情况时，偶尔无法完全精确地实现所定义的 HWFS。这种做法能带来多少节省呢？我们自己在尝试测量互联网拥塞方面的经验表明，拥塞现象相对罕见，而 Dhamdhere 等人的先前研究也证实了这一点：“我们没有发现美国接入互联网服务提供商与直接连接的传输和内容提供商之间的域间链路存在普遍的长期拥塞现象”[21]。由于我们预计受约束的聚合体数量相对于总聚合体数量而言较少，因此我们推测这种近似方法将是实现 RCS 的有效方式。

8 相关工作

我们首先讨论两个有望取代 TCPF 的主要竞争者，最后简要列出其他相关工作。

那么，另外两种主要的替代方案又如何呢？如前所述，文献中提出了两种主要的 TCPF 替代方案。第一种方案最初由 [36] 提出，并被许多人进一步探讨，即每流公平性。然而，此类方法的真正益处并非在于它们提供了道德上更优越的资源分配；相反，其真正的好处在于这些方法实现了流之间的隔离，从而达到了 CCAI。Briscoe 在 [10] 中“拆解”了这一方案，他指出由此产生的分配在经济上毫无意义。无论流的定义如何（例如，按源、按目的、按源目的对、按五元组），都与当前互联网的商业安排毫无关系。当然，正如 Briscoe 所指出的，TCPF 在经济上也没有意义；因此，实现了 CCAI 但经济上不合理，每流公平性严格来说只是比 TCPF 稍有改进。相比之下，通过 RCS 我们力求实现 CCAI 和经济合理性，而每流公平性显然无法做到这一点。

TCPF 的另一种替代方案是网络效用最大化（NUM），其中每个流都有一个效用函数，目标是最大化该效用。这一方法由凯利在[32, 33]中提出，并引发了大量相关文献。NUM 的基本思想是，拥塞信号可以作为影子价格，衡量特定流对其他流造成的拥塞程度。如果各个拥塞控制算法（CCA）优化自身效用减去该影子价格，那么系统在均衡状态下将实现效用总和的最大化，这便是社会最优的结果。

这一核心理念可以有两种运用方式。最直接的方式是实际收取影子价格（即用户必须支付其产生的所有影子价格费用），然后让用户据此自私地进行优化。为此，用户需要在 CCA 中定义其（通常是未知的）效用函数。然而，就我们的目的而言，最相关的反对意见在于，这

这种办法需要对用户使用互联网的收费方式做出重大改变，这显然超出了我们的研究范围。

人们也可以将拥塞信号作为提示，让拥塞控制算法（CCA）对其作出反应，就好像它们是自我优化的一样。这种方法实质上要求在路由器上采用通用的拥塞控制算法和通用的拥塞信号机制。因此，这将用自愿的网络利用度量（NUM）范式取代自愿的 TCP 友好范式。但这并不能解决激励问题，因为那些忽略这些拥塞信号的拥塞控制算法会获得更好的服务。在更有限的部署环境中，比如数据中心，这种方法更可行，因为网络服务于的是运营商的需求，而非个体用户。有关此类示例，请参阅[35]。

在我们的环境中，NUM 存在的一个更为深刻的难题在于，它与互联网当前商业模式的粒度不一致。在当前的商业模式中，各实体（可能是家庭用户、企业或其他提供商）以相对稳定的价格从服务提供商处购买服务。这些实体向其提供商支付费用以获得发送和接收数据包的能力。在这个过程中存在一定程度的效用最大化，但这是在购买接入服务的实体层面，而非单个网络流的层面。RCS 通过确保其流量在网络中传输时所受到的待遇在一定程度上反映其购买的接入级别（以拥塞份额、其提供商从其对等方获得的拥塞份额等来衡量）来解决这一层面的效用问题。

那么其他相关工作呢？关于拥塞控制的文献浩如烟海，为本研究提供了背景，我们无法一一提及。不过，最近有三篇立场论文对 TCP 友好性（TCPF）的问题提出了新的见解：[46] 对 TCP 友好性及其存在的问题进行了精彩的讨论，[42] 从新颖的角度探讨了拥塞控制算法（CCA）的多样性，[13] 则对 TCPF 在决定带宽竞争中的作用提出了质疑。实际上，即便 TCPF 当前并未决定带宽分配，RCS 也提供了一个有原则的框架，而非随意或晦涩难懂的框架。

与此同时，有两项工作提出了我们第 6 节所述 HDWRR 机制的替代方案。HCSFQ [48] 扩展了经典的 CSFQ [43] 以近似实现 HWFQ。我们之所以没有采用这种机制，是因为它要求核心交换机估计层次结构中所有聚合体的到达率。Gearbox [27] 则遵循一系列近似 WFQ 的工作思路，通过少量的物理 FIFO 队列实现了层次化日历队列。将 Gearbox 扩展到近似 HWFQ 会加剧日历范围溢出的问题，这也是我们没有探索这一方向的原因。

9 结论

有人可能会认为这篇论文毫无必要（如今的互联网运行状况还算不错）、未经检验（其提出的机制尚未在大规模环境中得到验证）、不切实际（其机制在短期内无法部署）以及反应过度（将 BBR 的采用视为一场灾难，而非在长期遵循 TCP 友好性原则的历史中的一次小小波动）。我们对上述所有指责都甘愿认罪，但认为这些反对意见大多偏离了我们的核心观点。

本文绝非声称能以经过充分测试且易于部署的方案来解决一个紧迫的实际问题。相反，我们的目标是解决自纳格尔 1985 年的论文[36]以来一直悬而未决的一个基本概念性问题，即：如何协调 CCA 独立性的目标与互联网的商业现实？这是一个重要的问题，因为前者（CCA 独立性）无疑是有益的，它将极大地促进 CCA 的创新，而后者（商业现实）在可预见的未来不太可能发生根本性的改变。

我们认为，即便没有迫在眉睫的危机，这种困境也值得我们从智力层面予以关注。我们的方法似乎解决了这一概念上的困境。这一解决方案无论在概念上还是实践中都不简单，要从理论上更好地理解这一方法，并使其更具实际可操作性，还有很多工作要做。尽管如此，我们认为这篇论文仍是一个有益的开端。

致谢

我们感谢 Aurojit Panda、Tejas Narechania、我们的指导人 Sachin Katti 以及匿名审稿人的宝贵意见。本研究得到了美国国家科学基金会（NSF）的资助，项目编号分别为 2201489、2212102、2213387 和 2242502，同时也得到了 IBM、英特尔和博通公司的资助。

参考文献

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

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

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

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

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

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

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

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

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

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

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

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

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

[14] T. Böttger, G. Antichi, E.L. Fernandes, R. Lallo, M. Bruyere, S. Uhlig, and I. Castro. The Elusive Internet Flattening: 10 Years of IXP Growth. In RIPE 78, 2018. Cited on page 11.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[46] Ranysha Ware, Matthew K. Mukerjee, Srinivasan Seshan, and Justine Sherry. Beyond Jain's Fairness Index: Setting the Bar For The Deployment of Congestion Control Algorithms. In HotNets, 2019. Cited on pages 1 and 12.

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

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

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

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

附录

附录是未经同行评审的辅助材料。

附录 A HDWRR 实现

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

清单 1：HDWRR 的实现。

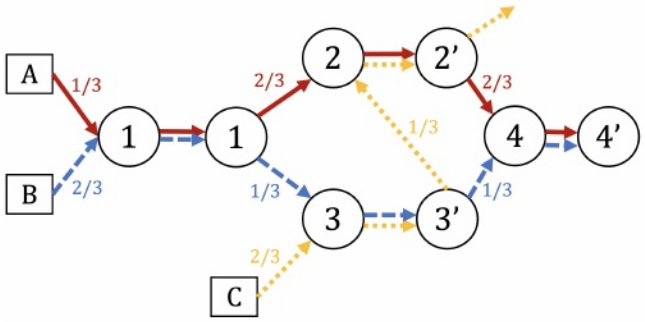
附录 B 博弈论示例

B.1 多个纳什均衡

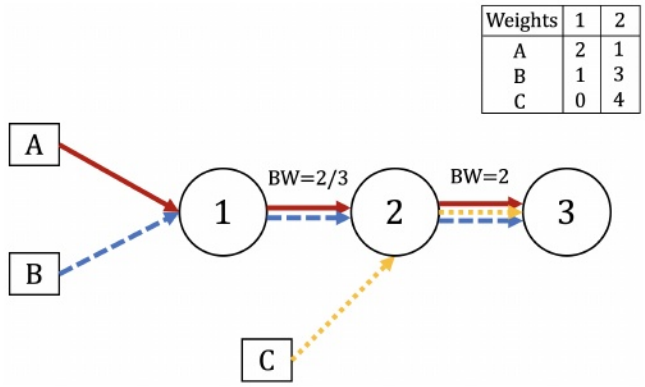
现在我们来说明，即使存在 RCS，拓扑结构也可能存在多个均衡状态。图 6a 展示了一个具有两个纳什均衡的例子，即 （ $_{3}$2 , 13 , 13 ） 和 （ 12 , 12 , 12 ） 均为纳什均衡。如果任何流量试图增加，它们会立即在其中一个出口点遭受持续的损失。回想一下，当我们根据 CAIDA 数据对现实拓扑结构进行建模时（§5.2），我们没有发现 RCS 存在这种行为的任何实例。

B.2 非纳什斯塔克尔伯格

图 6b 中的示例描绘了一个不存在纳什均衡但存在唯一斯塔克尔伯格均衡的场景。可以验证，唯一满足第 5.1 节中所述纳什均衡必要条件的解是 （13， $_{3}$1， 43），但这并非纳什均衡。这是因为 A 可以将其速率提高一个很小的 ，这会降低 B 的吞吐量，而 C 的吞吐量保持不变，从而使其自身的总吞吐量增加 。尽管这不是纳什均衡，但这个解



示例中存在两个纳什均衡（23，13，13）和（12，2，1，12），且不存在中间均衡。所有容量均为 1。



b：不存在纳什均衡但存在斯塔克尔伯格均衡的示例。

图 6：具有不良收敛特性的示例拓扑结构。

这是一个斯塔克尔伯格均衡，因为如果 A 提高其速率，那么（由于斯塔克尔伯格模型考虑的是从领导者的变化中各流如何做出反应）除了 B 会降低其速率之外，C 也会略微提高其速率，从而导致 A 遭受损失；这使得 A 的吞吐量低于其发送速率，从而使其效用完全丧失。因此，从斯塔克尔伯格均衡的角度来看，A 在当前速率下状况最佳。因此，这是一个没有纳什均衡但存在斯塔克尔伯格均衡的场景示例。

附录 C 端点控制

在 4.3 节中，我们介绍了端点控制原则，该原则规定端点应负责确定其流的组成，而非网络。然而，要实施这种控制，需要在瓶颈链路（即队列积压的链路）上制定处理数据包的策略，而这些瓶颈链路往往不在端点的控制范围内。在此，我们转向两项近期的研究成果，即 Bundler [16] 和 CRAB [45]，边缘域可以与 RCS 结合使用这两项技术来实施其调度策略。

Bundler [16] 的核心思想是将流队列的积压从瓶颈点（端点域之外）转移到发送方，以使发送方能够执行其策略。迁移队列需要动态且准确地估计速率。

该流所传送的内容以及在发送方对流量进行速率限制。这两项任务通过在发送方和接收方之间插入发送箱和接收箱来完成。发送箱和接收箱测量拥塞信号，并在发送箱中利用基于延迟的拥塞控制算法来确定流的发送速率。然后，发送箱根据所确定的速率对出站流量进行速率限制，从而在本地建立一个常驻队列。选择基于延迟的拥塞控制来进行速率估算，这不仅对机制至关重要（基于丢包的控制环路会使瓶颈链路的队列充满），而且还能使对延迟敏感的组成流实现低延迟。

CRAB [45] 则侧重于让用户能够对其接入链路瓶颈处的入站流量进行加权，并获得加权最大最小公平份额速率。计算这些速率需要权重、瓶颈容量和需求。由于不清楚流量是受自身需求限制还是在瓶颈处受竞争限制，因此很难直接从瓶颈链路之后感知到的速率来确定流量的需求。为解决这一问题，CRAB 在接收端对流量进行轻微限速，使其略低于感知速率，并观察当提供更多的带宽时流量的行为，以判断流量的需求是否得到满足，从而确保流量永远不会获得超出其在加权最大最小公平性下应得的份额。

更具体地说，CRAB 会以轮次为单位估计接收方接入链路的总容量以及每个流的感知速率。在一轮速率估计之后，会对流进行分析，以确定它们是否属于以下一种或多种情况：增长型（正在使用分配给它的额外带宽）、饱和型（正在增长或已达到分配的带宽）、非饱和型（未达到分配的带宽）。如果存在至少一个非饱和流和一个饱和流，那么 CRAB 就会启动带宽重新分配过程。在重新分配期间，CRAB 首先确定每个流可以出借的总带宽量。然后，它运行一种水填充算法来确定每个流的新速率，从而将此额外容量分配给每个饱和流。如果出借带宽的流后来被归类为增长型，那么 CRAB 会归还所有出借的带宽。最后，CRAB 必须持续了解瓶颈链路的当前容量，以便准确确定速率。理想情况下，CRAB 只在可用容量实际发生变化时才降低观测到的容量，而不是在流需求突然下降时。因此，当观测到的总容量下降时，CRAB 首先尝试重新分配带宽，前提是存在任何饱和流，否则就降低观测到的容量。CRAB 还通过在测量期间提高速率来探测更高的总吞吐量。观测到的容量的这些变化会导致对每个流的加权最大最小公平速率进行进一步的重新计算。