FlowTele: Remotely Shaping Traffic on Internet-Scale Networks

Bo-Rong Chen[1,†] Zhuotao Liu[2,†] Jinhui Song[1] Fanhui Zeng[1] Zhoushi Zhu[1]
Siva Phani Keshav Bachu[1] Yih-Chun Hu[1]
[1] University of Illinois at Urbana-Champaign [2] Tsinghua University

ABSTRACT

Internet content providers often deliver their content through bandwidth bottlenecks that are out of their control. Thus, despite often having massively over-provisioned upstream servers, the content providers still cannot control the end-to-end user experience. In this paper, we explore the possibility of *remote traffic shaping*, allowing the content provider to allocate its share of a remote bottleneck link across its users using a metric other than TCP fairness, while remaining TCP-friendly to cross traffic on the bottleneck link. To evaluate this approach, we designed FlowTele, the first system that shapes outbound traffic on an Internet-scale network to optimize provider-selected metrics, using source control with neither in-network support nor special client support. Our extensive evaluations over the Internet show that by strategically reallocating bandwidth among provider-owned co-bottlenecked flows, FlowTele improves the provider's total revenue by roughly 20%∼30% in various network settings, compared with both (i) status quo TCP fair-share and (ii) recent practice by content providers that proactively throttles video quality during the COVID-19 pandemic, while being TCP-friendly to cross-traffic. Besides revenue, we also study other metrics, such as QoE fairness, that a content provider may like to optimize using FlowTele.

CCS CONCEPTS

• **Networks** → **Network management**;

ACM Reference Format:
Bo-Rong Chen[1,†] Zhuotao Liu[2,†] Jinhui Song[1] Fanhui Zeng[1] Zhoushi Zhu[1] Siva Phani Keshav Bachu[1] Yih-Chun Hu[1] . 2022. FlowTele: Remotely Shaping Traffic on Internet-Scale Networks. In *The 18th International Conference on emerging Networking EXperiments and Technologies (CoNEXT '22), December 6–9, 2022, Roma, Italy.* ACM, New York, NY, USA, 20 pages. https://doi.org/10.1145/3555050.3569139

1 INTRODUCTION

Internet traffic increasingly originates from a small number of large content providers. For instance, six major content providers headquartered in the USA send about 43% of global Internet traffic [13]. Although these content providers have massively provisioned their own network infrastructure through datacenters in different locations [54], private backbone WANs [30] and Internet peers [53, 64], their traffic may still experience bottlenecks on the downstream links beyond their control before reaching their customers. Google has consistently published "Video Quality Reports" [21] to raise consumer awareness that some ISPs do not provide Google with sufficient bandwidth to serve HD video to their mutual customers. Such bandwidth limitations sometimes are introduced by traffic engineering where ISPs can limit traffic they deem undesirable; for example, Verizon demanded interconnection fees from Level3, and when Level3 refused to pay, Verizon limited bandwidth at Level3's peering points [58]. The FCC also reports that some ISPs lack sufficient bandwidth to handle peak bandwidth demand [10]. More recently, some content providers (such as YouTube and Netflix) proactively throttled video quality to reduce bandwidth pressure on downstream links during the COVID-19 pandemic [20, 60].

The key motivational insight for this work is that despite the massively over-provisioned upstream network infrastructure and widely-deployed CDNs, content providers are still unable to simultaneously provide all customers with the highest-quality services because their flows often experience bottlenecks on downstream links beyond the control of the content provider. Therefore, if a content provider can divide its share of the downstream bottleneck bandwidth in a manner other than TCP-fair (while remaining TCP friendly to external cross traffic), they can shape the link bandwidth usage towards maximizing metrics of their choice. For example, some content providers may aim to improve social welfare, moderate political extremism, improve diversity, or improve access for disadvantaged users; YouTube [31], Facebook [19], and Twitter [46] already disfavor certain content. Some content providers may adapt the bandwidths they provide based on the requirements of each style of video, increasing bandwidth for high-motion video such as sports (as in [45]); others may improve revenue during peak hours by favoring paid subscriptions or advertising traffic. To date, no system exists that allows content providers to dynamically allocate bandwidth for social and economic benefits despite limited downstream network capacity.

We recognize multiple challenges to achieving such *cross-flow bandwidth reallocation on remote links* in open and decentralized networks like the Internet. First, designing allocation strategies is non-trivial. For instance, maximizing revenue subject to bandwidth limitations does not mean statically throttling "low-valued" users. Rather, such optimization requires delicate cross-flow bandwidth reallocations backed by user value and behavior modeling to optimize aggregate user value while minimizing impact on user retention.

Second, given a desired bandwidth allocation, content providers cannot easily impose optimized allocations on the bottleneck link. (i) The content providers do not own the entire path, so they cannot deploy in-network queuing or throttling mechanisms, as providers could in private WAN [30, 33] and datacenters [23]; (ii) the Autonomous System (AS) owning the bottleneck (or slowest) link on

†: co-primary authors with equal contribution.

CoNEXT '22, December 6–9, 2022, Roma, Italy

ACM ISBN 978-1-4503-9508-3/22/12.
https://doi.org/10.1145/3555050.3569139

the path may refuse to cooperate; and (iii) the content provider likely does not want to disclose its desired allocation policies to these remote ASes (*e.g.,* due to privacy concerns). Thus, the second challenge is how to *remotely* enforce provider-desired allocation policies on bottleneck links with neither in-network support at remote ASes nor recipient networking stack upgrades.

To meet these challenges, we propose FlowTele (*tele* inspired by *telekinesis*), the first system architecture on which content providers can build to remotely shape its network flows based on a combination of self-selected factors, while remaining TCP-friendly to cross traffic sharing the same downstream bottleneck path. FlowTele is powered by a stack of innovative designs. First, we design fAllocator, which computes the per-flow bandwidth allocations to optimize content-provider selected metrics. To demonstrate the capability of fAllocator (and the generality of FlowTele as a whole), we instantiate fAllocator with two concrete designs for optimizing economic and social benefits, respectively, in the context of video streaming applications, the most bandwidth hungry applications that dominate the Internet traffic [9, 11]. Specifically, we design vAlloc to optimize aggregate advertising revenue, and qAlloc to optimize quality of experience (QoE) fairness among co-bottlenecked flows.

Second, we design fShaper, a source-control mechanism to remotely enforce accurate weighted fair share across a set of flows. An abstraction of fShaper is that of a cross-flow congestion window reallocation algorithm that redistributes congestion windows among some co-bottlenecked flows based on fAllocator's output. Through extensive Internet-scale evaluations, we demonstrate that fShaper achieves both *internal weighted fairness*, *i.e.,* the outbound rates of controlled flows converge to desired weighted fair shares, and *external TCP friendliness*, *i.e.,* the controlled flows in aggregate exhibit fairness against TCP cross traffic. Finally, based on insights of prior work (*e.g.,* [28]), we evaluate FlowTele based on one method for flow co-bottleneck detection.

Contributions. To the best of our knowledge, FlowTele is the first system architecture on which content providers can build to reclaim complete control over their flows, even if they traverse uncontrolled bottleneck links owned by downstream ISPs. Notably, FlowTele simultaneously achieves *source-driven* and *in-network* traffic management, a benefit that has to date been unachieved in globally distributed and heterogeneous networks, such as the Internet, where flow sources have incomplete control over in-network routing and protocol deployment. FlowTele provides the high-speed adaptation needed to handle peak loads to fill buffers (such as client video buffers) of new flows while remaining TCP-friendly and managing the buffers of old flows. We fully implement FlowTele in about 9900 lines of code and evaluate it through large-scale Internet experiments and analytical simulations. The evaluation results show that FlowTele can achieve a 20%∼30% improvement when optimizing advertising revenue in various network settings, compared with TCP fairshare and a recent quality-throttling practice applied by some content providers during the COVID-19 pandemic. Further, *a single instance of FlowTele* can efficiently perform bottleneck allocations for hundreds of flows on a single CPU thread.

Limitations. Due to the vast range of possible metrics that a content provider may wish to optimize, this paper does not attempt to exhaustively enumerate or evaluate all such metrics. Instead, this paper aims at demonstrating the feasibility and performance of a system across two metrics under a range of parameterizations of user value and user behavior. As a network management technology, FlowTele has both ethical and unethical uses. In this paper, we focus on the technologies at the foundation of such systems, and leave the public policy implications to future work.

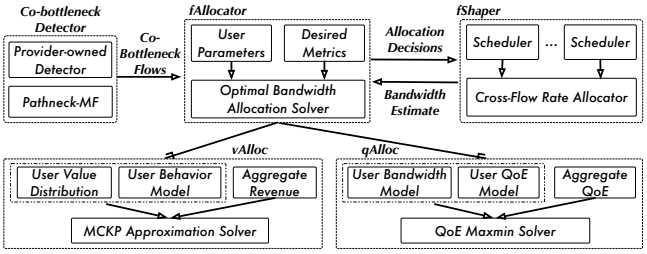


Figure 1: The architecture of FlowTele. We show two instances of fAllocator: vAlloc to optimize aggregate user value and qAlloc to fairly distribute quality of experience.

2 ARCHITECTURE

We illustrate the architecture of FlowTele in Figure 1.

FlowTele Applicability. FlowTele is designed for *bandwidth-constrained* applications where available bandwidth dictates user behavior or experience, and where a variety of bandwidths all create viable (though potentially different) experiences. Applications such as VR gaming, videoconferencing, video livestreaming, and federated computing could all benefit from the FlowTele architecture. When flows are largely application-limited, reallocation of bandwidth between flows brings little benefit, and the application of FlowTele is limited. When different content is sent for different experiences, FlowTele must control the content sent to provide a specific experience. In video delivery, traditional client-driven Adaptive BitRate (ABR) streaming is incompatible with FlowTele's need to deliver fAllocator-selected content, and some application redesign (*e.g.,* accommodating server-selected experiences) is needed. In this paper, we design and evaluate FlowTele for flows emanating from a single datacenter, and its use with distributed CDNs is left as future work.

fAllocator Overview. The key piece that allows FlowTele to optimize metrics selected by the content-provider-selected across multiple co-bottlenecked flows is fAllocator, the brain that decides the desired bandwidth allocation among co-bottlenecked flows in order to achieve certain metrics. Since video streaming applications are bandwidth hungry and dominate Internet traffic [11], we describe fAllocator for such applications; however, the methodology of fAllocator can be applied to new allocator designs for the applications described above.

fAllocator's goal is to find the (bitrate, client buffer) pair for each flow that maximizes the content-provider-selected metrics subject to some aggregate limit on required bandwidth. fAllocator is architecturally agnostic to the provider's choice of metric, and even how that metric is computed. For example, a content provider whose primary revenue source is advertising can aim to maximize revenue, particularly during times of low bottleneck bandwidth. In this case, the metric is aggregate user value that the provider can extract over the bottleneck; we present vAlloc in § 3 to take this

approach. Another example is a content provider that chooses to provide equitable QoE, even across disparate network connections (and consequently disparate buffer requirements). In this case, the metric is QoE and the aggregation could be the minimum across all metrics, *i.e.,* aiming to maximize the experience of the lowest-experience user. We develop qAlloc in § 4 to take this approach.

fShaper Overview. The design of FlowTele requires the content provider to shape outbound traffic to achieve the per-flow bandwidth allocations specified by fAllocator while simultaneously remaining friendly to non-participating traffic sharing the same bottleneck links. We design fShaper, a source-control based mechanism to redistribute the provider's total TCP fairshare over a bottleneck link among the set of provider-owned flows in accordance with fAllocator's outputs. We implement fShaper on TCP and demonstrate its effectiveness via substantial Internet-scale experiments. As an entirely sender-based mechanism, fShaper relies on neither in-network support nor special receiver implementations.

Co-Bottleneck Detection. Though not the focus of this paper, FlowTele requires some form of co-bottleneck detection, since strategic bandwidth reallocation can only be performed between co-bottlenecked flows. FlowTele could apply any existing co-bottleneck detection approach, such as [1, 17, 24, 25, 32, 34, 49, 51, 52, 63]. We demonstrate a design based on a modified version of Pathneck [28] in our experiments (§ 6.3). These experiments set a lower-bound for co-bottleneck detection performance, since content providers can further infer co-bottleneck links by monitoring their servers deployed within the ISPs [22]. Video Quality Reports [21] also reveal that Google has some information about ISP-induced congestion.

Application Integration. Each FlowTele application needs a fAllocator, which generates allocation decisions on a per-flow basis. The application then must send content consistent with the allocation decision, and use fShaper as a transport to deliver that content. For example, in our video streaming examples, the client connects to a server-side application that takes fAllocator decisions, directs each client to download specific chunks, and to arrange the buffer in a specific manner. The fShaper and co-bottleneck detection mechanisms are generic, applicable across all applications.

3 VALLOC DESIGN

In this section, we present vAlloc, an instance of fAllocator that aims to maximize the aggregate user value under a downstream bandwidth constraint.

3.1 vAlloc Ad Revenue Exploration

A key observation of vAlloc is that a user's economic value to a content provider may not be uniform, and may even follow a heavy-tailed distribution. Thus, a small shift in the behavior of high-value users can provide disproportionate returns to a content provider that faces downstream congestion. This non-uniform distribution of user values arises from several causes: (i) many content providers rely on advertisements, the values of which are often heavy tailed; (ii) users of social media provide additional value to content providers by providing content that can attract more viewers; such social influence is often heavy-tailed (*e.g.,* as measured by Twitter and Instagram followers or YouTube subscribers); (iii) even for content providers that charge a flat monthly rate, the unequal

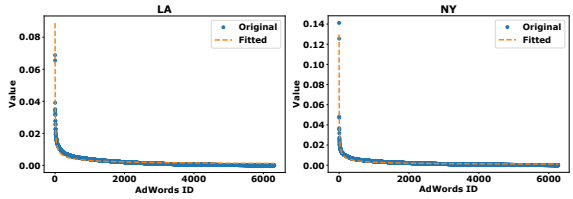


Figure 2: The advertisement values of our collected Google Adwords. We fit the raw data using Weibull distributions.

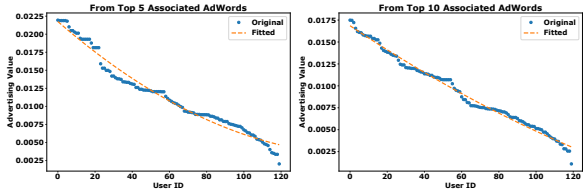


Figure 3: The value distributions of user profiles synthesized based on the Facebook and Google datasets and quadratic fit.

distribution of viewing frequency means that different users have different value in terms of revenue per hour.

We collected and analyzed two real-world datasets to quantitatively validate the non-uniform distribution of user values. First, we gathered a collection of Google AdWords by providing Google's AdWord suggestion mechanism with all two-letter prefixes (aa, ab, . . . , zz) and compiling all suggested words, yielding over 6000 AdWords. Next, we chose a set of 14 cities across the United States by sampling the 2010 US Census list of Metropolitan Statistical Areas and determined the per-impression cost of each selected AdWord in each selected city. For each city, we sort the ad terms from highest value to lowest value and plot the ad term number on the x-axis against its value per impression on the y-axis. Two representative results are shown in Figure 2; we present more results in § F. We fit each city's data to a Weibull distribution. These figures show that the value distribution of our collected AdWords is heavy-tailed.

In the 120 Facebook profiles from the ADS dataset [50], each profile has a list of likes and dislikes, which we manually mapped into the list of our harvested Google AdWords. For each profile, we select the top 100 most valuable AdWords to reflect each profile's potential interest in associated products. We considered four different advertising schemes: one where each advertisement is randomly drawn from the top 5, 10, and 20 highest-value AdWord categories of interest to the profile, and all such categories. For each advertising scheme, each user has a specific value, which we fit using three distributions: quadratic, exponential, and Weibull. Figure 3 shows the top-5 and top-10 strategies with quadratic fit; other results are in § F.

Our goal in this exploration is not to find a single user value distribution valid in all applications because user values vary across providers, applications, economies, and seasons. Rather, we put together a collection of distributions that represent user value distributions characterizing our diverse datasets. Thus, even without knowing the true user value distribution in any particular application, our analysis strongly suggests that user values are often nonuniform. Furthermore, the distributions we choose affect only our quantitative evaluation results; any service provider implementing FlowTele would have run-time information on user value and would not need to rely on a third-party user value distribution.

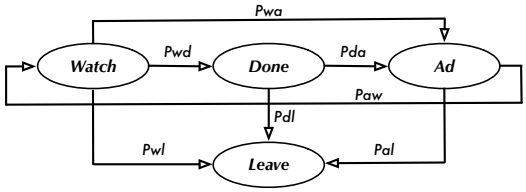


Figure 4: The finite state machine for modeling user behavior in video streaming applications.

Though the datasets we analyzed did not include video streaming ad values as they are inaccessible to us, limited public disclosures by content creators (*e.g.,* [61]) demonstrate that ad values are far from uniformly distributed.

3.2 User Behavior Modeling

Modeling Overview. In order to quantify the impact of bandwidth reallocation on provider revenue, we design a user behavior model for video streaming. FlowTele does not rely on a specific user behavior model; rather, it can adapt to the pattern of behavior collected by each content provider. We developed our user behavior model mainly to evaluate vAlloc, based on previous work [15] that shows that buffering time is the single most significant factor in user retention. While that work showed that resolution was relatively unimportant, recent data suggests otherwise; for example, the YouTube video quality report [21] distinguishes between ISPs that allow HD-quality video and sub-HD-quality video. As a result, we design a model that includes both client buffer time and video quality; this model is based on the Finite State Machine illustrated in Figure 4. In our model, a user progresses through various states of the streaming process, labeled Watch (when the user watches content) and Ad (when the user watches an advertisement). We also have two transient states: Done when the current video has finished playing and Leave when the user leaves the current session.

Since traffic management can only impact the *streaming quality*, in this model, different bandwidth allocations can only affect a user's willingness to continue to watch the video and advertisement (*i.e.,* the transition probabilities p_{wl} and p_{al} in the state machine). Other parameters are either system-related or user-specific, and cannot be impacted by FlowTele. For instance, at $p_{\text{dl}} = 0$, the user's watching time is entirely driven by video quality metrics, while at $p_{\text{dl}} = 1$, video quality may increase a user's watch time but will not help serve another ad in this session. p_{da} is the probability that an advertisement is presented by the provider after a video is finished.

Computing p_{wl} and p_{al}. Prior studies [15, 45] examine how buffer time and playback bitrate (video resolution) affect user engagement. We use data from these studies to model p_{wl} and p_{al} using the methodology below. In a real deployment, content providers could build their customized model using the method described in § E.

We model user reaction to buffer times using available-bandwidth measurements of several residential networks. (1) We took measurements and built a Markov model of the available network bandwidth to capture realistic bandwidth fluctuations for a certain ISP-advertised bandwidth (denoted as $\text{bw}_{\text{advertise}}$). (2) For a set of typical bitrates {BR} [67] and buffer times {BUF} [15], we run a Markov simulation to determine the buffering experiences during a video streaming session, as quantified by buffering ratio (fraction of time spent buffering) and buffering events (number of distinct playback stalls). (3) Given the buffering experience, we derive the watching time t_{watch} based on [15]. (4) Given a video streaming session with length $\text{video}_{\text{length}}$, we formulate the p_{wl} as the departure probability of a Poisson process with parameter $\lambda = t_{\text{watch}}$.

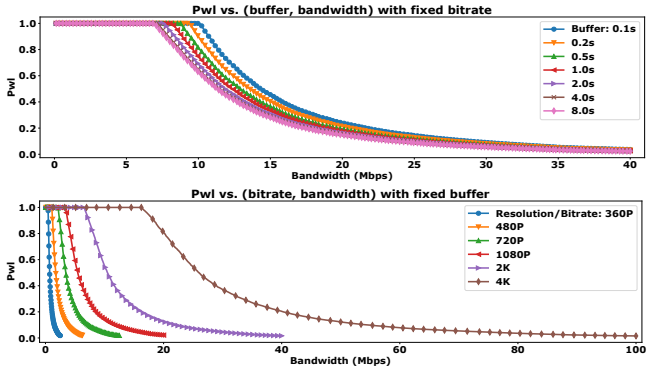


Figure 5: The p_{wl} during a 260-second video watching session, with varying (br, buf, $\text{bw}_{\text{advertise}}$).

We model p_{al} in the same way as p_{wl} and summarize the above procedure in Algorithm 1. Figure 5 shows the computed p_{wl} for a 260-second video streaming session (the average video length for top YouTube videos [44]). For clearer presentation, we break the three dimensions (br, buf, $\text{bw}_{\text{advertise}}$) into two pairs.

3.3 Bandwidth Allocation Algorithm

vAlloc chooses the optimal bandwidth allocation in each control interval as follows. Let the system state be $\text{st}^t = \cup_{i \in \mathcal{S}}(\text{br}_i^t, \text{buf}_i^t)$, where br_i^t and buf_i^t are the playback bitrate and buffer time for user u_i. vAlloc aims to choose the next system state st_{t+1} to maximize the expected value across all users, subject to available bandwidth.

Expected User Value. To compute the expected value of a user in state $(\text{br}_i^t, \text{buf}_i^t)$, we model the streaming system as an infinite stream of alternating videos and advertisements, and compute the asymptotic user value based on the transition probabilities (*e.g.,* p_{wl} and p_{al}) determined by the state. Specifically, we set p_1 to the probability that the user will continue to the *next revenue point*, and p_2 to the probability that the user will continue from one revenue point to the next one. For simplicity of notation, we describe the asymptotic user value when videos and advertisements have fixed length of $\mathcal{L}_v$ and $\mathcal{L}_a$, respectively, and the value for all advertisements is $\tilde{v}$ (a value decided by the user value); however, our model easily extends to heterogeneous video length and advertisement values. The asymptotic user value over the infinite stream is computed as

$$\begin{aligned} \mathcal{V}_{u_i} &= \sum_{k=0}^{\infty} \tilde{v} \cdot p_1 \cdot p_2^k = \frac{\tilde{v} \cdot p_1}{1 - p_2}, \text{where} \\ p_1(t) &= \frac{\mathcal{L}_v}{\mathcal{L}_v + \mathcal{L}_a} \cdot (1 - p_{\text{wl}})^{\frac{\mathcal{L}_v - t}{t}} \cdot (1 - p_{\text{dl}}) \cdot (1 - p_{\text{al}})^{\frac{\mathcal{L}_a}{t}} \\ &\quad + \frac{\mathcal{L}_a}{\mathcal{L}_v + \mathcal{L}_a} \cdot (1 - p_{\text{al}})^{\frac{\mathcal{L}_a - t}{t}}; \text{and} \\ p_2 &= (1 - p_{\text{wl}})^{\frac{\mathcal{L}_v}{t}} \cdot (1 - p_{\text{dl}}) \cdot (1 - p_{\text{al}})^{\frac{\mathcal{L}_a}{t}}. \end{aligned} \tag{1}$$

$p_1(t)$ depends on the current state of the user (whether in watching a video or advertisement) and the time until the next revenue point; p_2 is constant.

Optimal State Transition. Given this user and revenue model, vAlloc aims to choose an optimal next-system-state st^{t+1} from st^t. For each user u_i, the cost of moving from the current state $(\text{br}_i^t, \text{buf}_i^t)$ to the next state $(\text{br}_i^{t+1}, \text{buf}_i^{t+1})$ depends on the relative bitrate br_i^t and br_i^{t+1}. Specifically, if $\text{br}_i^{t+1} \leq \text{br}_i^t$, the current buffered content is still usable, so the bandwidth cost is $\text{br}_i^{t+1} \cdot \max\{0, \tilde{t} + \text{buf}_i^{t+1} - \text{buf}_i^t\}$, where $\tilde{t}$ is the length of the control interval. Otherwise, we retain a limited buffer of $\text{buf}_{\text{retain}}$ at the current bitrate to avoid a rapid reduction in buffer length that could lead to buffering delay, at a bandwidth cost of $\text{br}_i^{t+1} \cdot \max\{0, \tilde{t} + \text{buf}_i^{t+1} - \min\{\text{buf}_i^t, \text{buf}_{\text{retain}}\}\}$,

Because we choose our bitrate and buffer time in discrete steps, vAlloc's problem is an instance of the Multiple-Choice Knapsack Problem (MCKP) [55], where for each item (user) the choices are the possible (br, buf) pairs, the value of each choice is the expected user value of that pair, the cost is the bandwidth cost computed as described above, and the capacity of the knapsack is the estimate available bandwidth given by fShaper (see details in § 5). Because MCKP is NP-hard, we do not always use an exact solution; rather, we use the FPTAS fully-polynomial approximation to find a solution that has value at least 99% of the optimal solution. We summarize the vAlloc allocation in Algorithm 1.

Improving Allocations. The MCKP makes an optimal decision only for the next interval. Considering future intervals is an instance of the Multi-Dimensional Multiple-Choice Knapsack Problem (MMKP), since each interval is one dimension in this problem. Because MMKPs do not have fully-polynomial approximations unless P=NP [41], we use single-dimensional MCKP.

3.4 Standalone vAlloc Evaluations

In this section, we present the evaluation of standalone vAlloc.

Comparison with Other Allocation Schemes. We compare vAlloc with two classes of bandwidth allocation schemes: the *status quo* where users' bandwidth is decided by the transport protocol and a strawman design where content providers ignore the lowest-valued users and serve the rest equally. We considered both user-value distributions derived from real data in § 3.1.

Figure 6 shows the aggregate user values achieved by vAlloc and other bandwidth management mechanisms. We consider a user population where new users come over time independent of the number of existing users. vAlloc achieves over 95% of the total user value when available bandwidth drops to roughly 50% of maximum bandwidth, as labeled in Figure 6. The advantage of vAlloc over other schemes peaks when the available bandwidth is about 50% of the maximum bandwidth, and vAlloc is even better as the distribution becomes more heavy-tailed. Finally, our results show that simply removing low-valued users is not optimal.

Allocation Sensitivity. We evaluate vAlloc using different parameters for the user behavior model and value distribution. Since actual parameters vary from provider to provider, we aim to show that vAlloc's allocation provides strong improvements across a variety of different parameters. Since bandwidth reallocation impacts only the three leaving probabilities (*i.e.,* p_{wl}, p_{al} and p_{dl}) in the state machine, we evaluate vAlloc with different p_{wl} and p_{dl} (we choose $p_{\text{al}} = p_{\text{wl}}$). We set p_{wl} by linearly scaling the probability distributions shown in Figure 5. For p_{dl}, we evaluate several discrete values. We also considered different parameters for both our user value distributions (quadratic and Weibull). Figure 7 shows our results. In general, vAlloc performs well across a variety of user models, providing least benefit when bandwidth allocation cannot affect user value (*i.e.,* $p_{\text{dl}} = 1$ or flat user value distributions), and more benefit when bandwidth allocation has greater impact (*i.e.,* $p_{\text{dl}} = 0$ or more heavy-tailed user value distributions).

Social Fairness of vAlloc. Finally, we show that vAlloc does not significantly negate user experience despite bandwidth reallocation. In this segment, vAlloc is configured to split the total available bandwidth into two buckets: one bucket for optimizing total user

Inputs:
Current user set $\mathcal{S}$ with value distributions in § 3.1
The state of user i in control interval t: $\text{st}_i = (\text{br}_i^t, \text{buf}_i^t)$
{BR, BUF}: all possible choices of bitrates/buffer times
Outputs:
The system state for the next control interval st^{t+1}

Func: vAlloc **Allocation:**
// An instance of Multiple-Choice Knapsack Problem (MCKP)
maximize $\left\{\sum_i \textbf{ExpectedValue}(\text{br}_i^{t+1}, \text{buf}_i^{t+1}, \textbf{UserModel})\right\}$
subject to $\forall i\ (\text{br}_i^{t+1}, \text{buf}_i^{t+1}) \in \{\text{BR}, \text{BUF}\}$
$\sum_i \textbf{BandwidthCost}(\text{br}_i^t, \text{buf}_i^t, \text{br}_i^{t+1}, \text{buf}_i^{t+1}) \leq \mathcal{B}$
$\mathcal{B}$: the bandwidth estimate given by fShaper
return $\text{st}^{t+1} \leftarrow \cup_{i \in \mathcal{S}} (\text{br}_i^{t+1}, \text{buf}_i^{t+1})$

Procedure: UserModel
$\mathcal{M}_{\text{user}}$: $\{p_{\text{wl}} \mid (\text{br}, \text{buf}, \text{bw}, \text{Video}_{\text{length}})\}$
// Build a Markov Model of available bandwidth via measurements
$\mathcal{M}_{\text{bw}} \leftarrow \text{MarkovModel}(\text{bw}_{\text{advertise}}, \text{Measurements})$
for $(br, buf) \in \{BR, BUF\}$ **do**
// bw is the advertised value in the meansurements
// The actual bw values in the simulation are drawn from $\mathcal{M}_{bw}$
$\text{BufferingExperience} \leftarrow \text{MarkovSim}(\text{br}, \text{buf}, \text{bw} \in \mathcal{M}_{\text{bw}})$
// Derive the watch time using the model in [15]
$t_{\text{watch}} \leftarrow \text{ExperienceModel}(\text{BufferingExperience})$
// Quantify p_{wl} *as the departure probaility of a Poisson Process*
$p_{\text{wl}} \leftarrow \text{Poisson}(t_{\text{watch}} = \lambda) \mid \text{Video}_{\text{length}}$
return $\mathcal{M}_{\text{user}}$

Func: ExpectedValue$(\text{br}_i^{t+1}, \text{buf}_i^{t+1}, \mathcal{M}_{\text{user}})$
return user value computed by Equation (1)

Func: BandwidthCost$(\text{br}_i^t, \text{buf}_i^t, \text{br}_i^{t+1}, \text{buf}_i^{t+1})$
if $br_i^{t+1} \leq br_i^t$ **then** *// The current bufferred content is still usable*
return $\text{br}_i^{t+1} \cdot \max\{0, \tilde{t} + \text{buf}_i^{t+1} - \text{buf}_i^t\}$
else *//* buf_{retain}*: the amount of buffer retained for bitrate* br_i^t
return $\text{br}_i^{t+1} \cdot \max\{0, \tilde{t} + \text{buf}_i^{t+1} - \min\{\text{buf}_i^t, \text{buf}_{\text{retain}}\}\}$

Algorithm 1: The vAlloc Allocation Algorithm

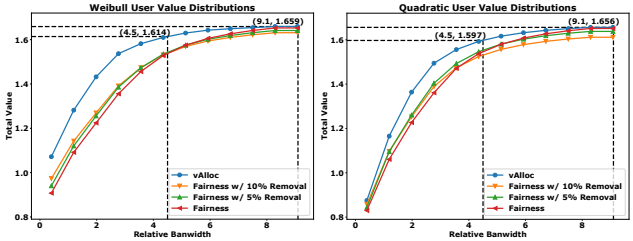
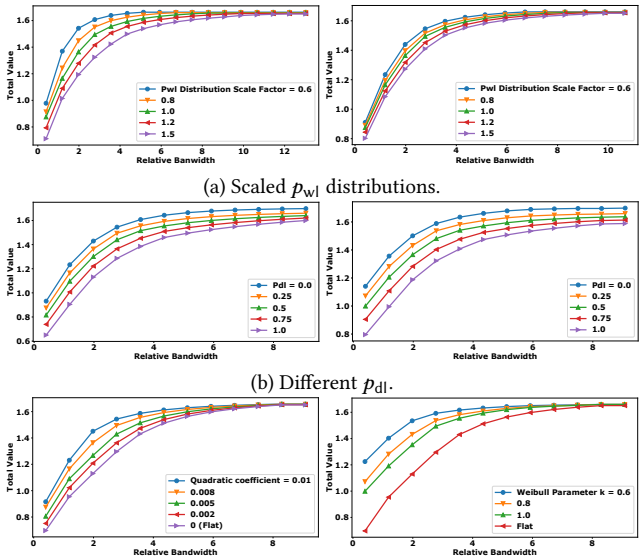


Figure 6: The aggregate user value achieved by vAlloc and three other bandwidth allocation mechanisms.



(a) Scaled p_{wl} distributions.

(b) Different p_{dl}.

(c) Different user value distribution parameters.

Figure 7: vAlloc's performance under different user behavior model and value distribution parameters. The left-side graphs use quadratic user value distributions, and the right-side graphs use Weibull user value distributions.

values and one bucket for optimizing social fairness, as measured by watching times of different users. Since we cannot directly impact watching times, we use streaming Quality of Experience (QoE, as measured by VMAF scores [36, 45]) in our problem formulation, assuming that better QoEs yield longer watching times. We evaluate two user repopulation schemes: constant inbound rate (denoted as F#U), as described earlier, and constant number of users (denoted as FIR), where every departing user is immediately replaced with a new user with value randomly chosen from the user value distribution. Both user value distributions obtained in § 3.1 are evaluated.

Our results are shown in Figure 8. Even when vAlloc is fully configured to optimize aggregate user value (the 100-0 Split), the bottom-quartile users' watching times drops only slightly (~10%) relative to the average watching time. Fairness improves when vAlloc allocates some bandwidth for QoE fairness. When fully configured to optimize fairness, vAlloc provides watching time fairness to all users, regardless of their values, as expected.

4 QALLOC DESIGN

Content providers may prefer to optimize metrics other than the aggregate user value. In this section, we present qAlloc, which allows content providers to optimize *user experiences* by optimizing the fairness of Quality of Experience (QoE) among all users. *qAlloc and vAlloc are not comparable; therefore it is invalid to say "qAlloc is better than vAlloc" or vice versa. They are two instances of fAllocator that show that FlowTele can be used to optimize very different metrics.*

Video playback QoE is driven by both video quality and rebuffering events. Because users have varying Internet connection quality in both ISP-advertised bandwidth (*i.e.,* the best-case bandwidth) and bandwidth variance, some users need larger client video buffers to have the same QoE as other users. Furthermore, different videos have different bandwidth requirements for the same quality level.

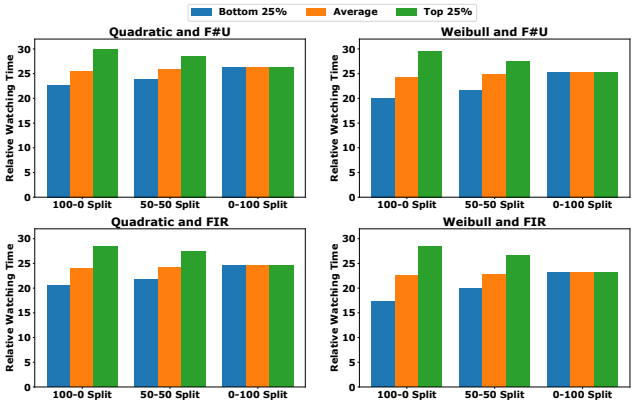


Figure 8: vAlloc imposes small impact on the watching times of low-valued users despite bandwidth reallocation.

We developed qAlloc to enable FlowTele to *evenly* distribute the QoE among users in case of downstream bandwidth constraint.

4.1 Video Streaming and QoE Model

Researchers have proposed various QoE models for streaming services [6]; our instantiation of qAlloc focuses on video bitrate, video type and rebuffering events. Generally, higher bitrate and less rebuffering events result in a higher quality, and the video type makes a difference as videos involving more motions, more sudden motions, and more scene changes need a higher bitrate to achieve a similar user experience [45]. Our allocator aims to provide fairness across QoE using a model that averages QoE for a single flow over next K chunks based on the Equation (2), which in turn is based on Model Predictive Control (MPC) [65], and reasonably characterizes the impact of bitrate, bandwidth and video type on QoE:

$$QoE^K_{\text{avg}} = \frac{1}{K}\left(\sum_{k=1}^{K} q_c(R_k) - \lambda \sum_{k=1}^{K-1} |q_c(R_{k+1}) - q_c(R_k))| - \mu T_r\right), \quad (2)$$

where R_k is the bitrate of chunk k, $q_c(R)$ is the per chunk video quality given bitrate R, T_r is the total rebuffering time over the next K chunks, and λ, μ are the penalty factors related to quality switching and rebuffering respectively.

Video Quality Metric. To obtain the per-chunk video quality function, we use the Mean Opinion Score (MOS) [59] as $q_c()$ in Equation (2) and choose AVT-VQDB-UHD-1 as our dataset. This dataset contains 16 Ultra-High-Definition (UHD) videos encoded with H.264, HEVC and VP9 and varying frame rates based on several short movies and provides a corresponding MOS for each segment [48]. We plot the relationship of MOS with video bitrate for all the videos of the dataset in Figure 9.

The MOS has two critical parameters: λ, the quality switch penalty, and μ, the rebuffering penalty. The choices of MPC [65], Minerva [45], and qAlloc are listed in Table 1. qAlloc is consistent with Minerva except that μ is scaled by a factor of $\frac{1}{25}$ to account for the range of MOS values as compared to VMAF values.

4.2 QoE Optimization Problem

The goal of qAlloc is to provide fair QoE by maximizing the lowest performing user's average QoE subject to the total bandwidth constraint through bitrate adaption. qAlloc operates with three time intervals: the look-ahead period (over which the QoE is evaluated), the chunk duration (the boundaries of which allow us to change

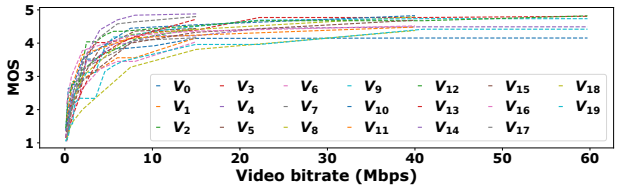


Figure 9: MOS versus video bitrate for the videos in the dataset [48].

System	λ	μ	$q_c()$ & range
MPC [65], Balanced	1	3000	rate ∈ [350, 3000]
MPC, Avoid Instability	3	3000	
MPC, Avoid Rebuffering	1	6000	
Minerva [45]	2.5	25	VMAF ∈ [0, 100]
qAlloc	2.5	1.0	MOS ∈ [1, 5]

Table 1: Comparison of λ, μ in the QoE modeling in Equation (2).

resolution), and the control interval (frequency at which control choices are calculated). Because we evaluate the QoE over the entire look-ahead period, when the look-ahead period is less than the remaining video buffer time, no stalling events occur and the QoE is entirely dependent on bitrate. We choose a long look-ahead period so qAlloc will value long buffer times when needed to offset extended periods of low bandwidth. The chunk duration is chosen to be a moderate value, on the order of seconds, since chunk downloads incur some minor overhead, but smaller chunks allow more rapid fine-tuning of bandwidth. The control interval is chosen to be very short (several times per second), allowing qAlloc to rapidly adjust to changes in the available bandwidth.

Let T be the look-ahead period, T_c be the chunk duration, and t_0 be the current time. Define K_T as the number of chunks within $[t_0, t_0+T)$, where the chunk at t_0 is counted while t_0+T is not. The buffer length of flow i when downloading chunk k is $\ell_{k,i}$, and the bitrate of the current playing chunk is $R_{0,i}$. Let $R_{k,i}$ be the bitrate of video i at chunk k, $\mathcal{B}_{k,i}$ be the allocated bandwidth of video i at chunk k. The goal is to optimize the minimum QoE among all the flows by controlling $(R_{k,i}, \mathcal{B}_{k,i})$.

However, compared to the typical QoE optimization problem with video bitrate adaption, our problem is more complicated for two reasons. First, qAlloc allows choices of not only the video bitrate but also the *bandwidth allocation* (impacting the remaining buffer at the end of the time period), significantly increaseing the number of choices for each stream. Second, to improve QoE fairness, qAlloc allocates across multiple flows jointly. A full search of $(R_{k,i}, \mathcal{B}_{k,i})$ would represent an exponential search space. Instead, qAlloc uses an online algorithm that reevaluates the QoE at each control interval. Although qAlloc makes allocation decisions to optimize QoE for the entire (long-term) look-ahead period, those decisions are reevaluated at each subsequent (short-term) control interval. Hence, some prior long-term allocations may be overridden by new short-term decisions. When computing QoE for $[t_0, t_0+T)$ using Equation (2), we simply assume the target bitrate and bandwidth allocation are constant across the entire look-ahead period.

The average QoE for flow i in the next K chunks is $QoE^K_{i,\text{avg}}$, *i.e.,* Equation (2) with R_k, T_r replaced by $R_{k,i}, T_{r,i}$. The total bandwidth of the co-bottleneck link is $\mathcal{B}_{\text{total}}$, and the advertised bandwidth by user/flow i's ISP is $\mathcal{B}_{\text{adv},k,i}$. Then our optimization problem can be formulated as:

$$
\begin{aligned}
\text{maximize} \quad & \left\{ \min_i \ \text{QoE}^{K_T}_{i,\text{avg}} \right\} \\
\text{subject to} \quad & \textstyle\sum_i \mathcal{B}_i \le \mathcal{B}_{\text{total}}; \\
& R_{1,i} = R_{2,i} = \cdots = R_{K_T,i}; \\
& T_{r,i} = \textstyle\sum_{k=1}^{K_T} \left(\frac{R_{k,i} T_c}{\min(\mathcal{B}_{\text{adv},k,i}, \mathcal{B}_i)} - \ell_{k,i} \right)_+ (t_k - t_{k-1}),
\end{aligned}
$$

where $(x)_+ := \max(x, 0)$, and $\ell_{k,i}$ follows the chunk-based buffer dynamics, *i.e.,* decreases as playing, and gets increased by T_c only when a new chunk is fully downloaded.

§ A presents our solver design.

5 FSHAPER DESIGN

The design of FlowTele requires the content provider to shape outbound traffic to achieve the per-flow bandwidth allocations specified by fAllocator while simultaneously remaining friendly to non-participating cross traffic sharing the same bottleneck links. Though a source can readily control its outbound bandwidth through UDP flows, using UDP would require client-side deployment, and maintaining external TCP-friendliness would require the use of a mechanism like Equation-Based Rate Control [18]. A source can likewise limit its TCP outbound bandwidth by controlling its sending rate, but the bandwidth it gives up does not get automatically reallocated to co-bottlenecked flows sent from the same source, but rather shared among remaining co-bottlenecked flows. As a result, the source also needs to estimate its aggregate *TCP fairshare* of the bottleneck-bandwidth so that fAllocator can honor this limit during bandwidth allocation.

We designed fShaper to meet these requirements and challenges, and to provide: (i) *internal weighted fairness*: the component flows of a fShaper source should have throughputs that converge to the target weights given by fAllocator, while having (ii) *external friendliness*: the component flows *in aggregate* neither undershoot nor overshoot the throughput of uncontrolled flows when competing with other cross traffic on a bottleneck link; *i.e.,* the flows controlled by fShaper in aggregate are friendly to external TCP cross traffic.

5.1 Design Detail

At its core, fShaper first estimates the source's *total* TCP fairshare of the bottleneck-bandwidth, and then allocates the total fairshare among the component flows (based on fAllocator's allocation decisions) by directly overwriting each flow's congestion window and leaving it fixed for each allocation interval. To this end, a fShaper source divides its component flows into three categories: recipient, donor and calibrator. Recipient flows have target weights (given by fAllocator) greater than their TCP fairshare and donor flows have target weights less than their TCP fairshare. Calibrator flows are passive observers for which fShaper does not update the congestion windows, *i.e.,* they follow a TCP congestion control algorithm (CCA). Calibrator flows statistically represent the source's per-flow TCP-friendly rate. Calibrator flows must be network-limited, and in a network-limited application, such flows should exist. By measuring the sending rates of calibrator flows, fShaper obtains an accurate measurement of the source's fairshare of the bottleneck.

Fairshare Estimation. We estimate the source's fairshare as $\mathcal{B}_{\text{fairshare}} = \frac{\mathcal{N}_{\text{total}}}{\mathcal{N}_{\text{calibrator}}} \cdot \mathcal{B}_{\text{calibrator}}$, where $\mathcal{B}_{(\cdot)}$ and $\mathcal{N}_{(\cdot)}$ are the total throughput and the number of flows, respectively. We estimate calibrator flow throughput as $\mathcal{T}_i = \frac{\mathcal{F} \cdot \text{MSS}}{\text{RTT}}$, where $\mathcal{F}$ is the number of

Content Provider Location	User Location	Cross-Traffic Sender Location	Provider-User / CrossTraffic-User RTT (ms)	Bottleneck Capacity (Gbps)
GCP Tokyo	Taiwan	GCP Taiwan	~40 / ~7	up to 1
GCP Los Angeles	Illinois	GCP Montréal	~55 / ~31	up to 1

Table 2: The settings of standalone fShaper evaluations.

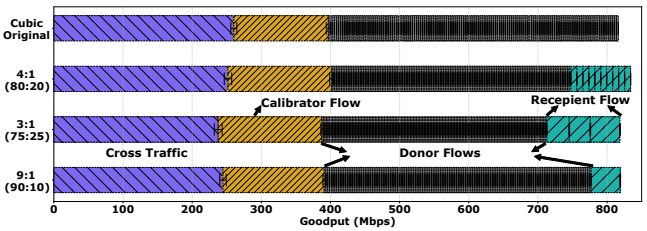
packets in flight, MSS is the segment size, and RTT is the round-trip time. Though flow throughput can be estimated from loss rate [43], we use values readily available in the Linux TCP implementation.

Congestion Window Updates. The bandwidth available for controlled flows is $\mathcal{B}_{\text{available}} = \mathcal{B}_{\text{fairshare}} - \mathcal{B}_{\text{calibrator}} = \left(\frac{\mathcal{N}_{\text{total}}}{\mathcal{N}_{\text{calibrator}}} - 1\right) \cdot \mathcal{B}_{\text{calibrator}}$. We distribute $\mathcal{B}_{\text{available}}$ among recipient and donor flows according to $\mathcal{T}_j = \frac{\text{weight}_j}{\sum_i \text{weight}_i} \cdot \mathcal{B}_{\text{available}}$, where weight_j is the j-th flow share computed by fAllocator. To achieve the target throughput $\mathcal{T}_j$, fShaper directly updates cwnd for flow j, setting $\text{cwnd}_j = \frac{\mathcal{T}_j \cdot \text{RTT}_j}{\text{MSS}}$.

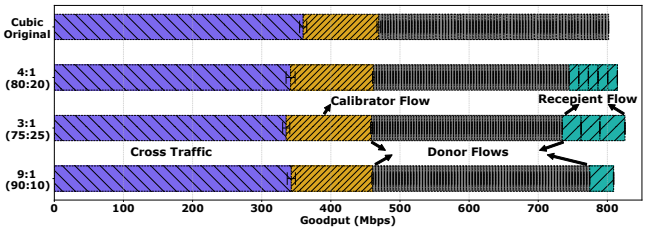
The cwnd of controlled (recipient and donor) flows and $\mathcal{B}_{\text{available}}$ are updated every T seconds, and we average the values of packets-in-flight and RTT, sampling every t seconds to smooth the effects of time synchronization across different TCP flows. Based on extensive experiments, for our evaluation environment, we set T to 0.1 second and t to 10 milliseconds. We supplement additional details in § B.1.

Controlled Flow Behavior. Because controlled flows' cwnd is directly set by fShaper, they do not react to congestion signals such as loss and latency. Instead, cwnd is set by dividing a target bandwidth by that flow's RTT, thus limiting the impact of RTT differences between calibrator and controlled flows. Controlled flows are selected only after they complete slow start; thereafter, neither slow start nor congestion avoidance operate on controlled flows. On timeout, the leading packet is retransmitted. On single packet loss, fast retransmit retransmits the lost packet, but cwnd is unaffected. Finally, changes to cwnd are accompanied by changes to pacing rate, so the network load created by the controlled flows in aggregate are relatively stable over time. Though controlled flows do not respond to normal CCA signals, they use congestion signals measured by calibrator flows to control their usage of the bottleneck link, which show slightly *less-than-fair-share* usage in our evaluations (§ 5.2 and § 6.1). fShaper is designed for environments where losses and other CCA signals are almost entirely caused by a single co-bottleneck segment, in which case these signals should be substantially similar for all flows.

Why Use Calibrator Flows? (i) It provides a simple way to make flows friendly to any given TCP variant. For instance, BBRv2 [8] uses a rich model which makes estimating the expected throughput complicated; using calibrator flows can avoid replicating such complexity while still providing external friendliness, as shown by our evaluation results in § C.2. (ii) It allows controlled flows to reach their target weights instantly. Other methods, such as MulTCP [12] or a variant used in Minerva [45] (denoted MulCubic), adjusts cwnd by changing the multiplicative-decreasing factor in TCP, need several RTTs to converge to the target weights. By contrast, fShaper directly adjusts cwnd in a centralized manner and relies on calibrator flows to achieve external friendliness. We prefer fast convergence since fAllocator outputs new target weights frequently, as we demonstrated by evaluation in § 6.1. In this paper, we select calibrator flows at random.



(a) Controlling majority of bottleneck bandwidth.



(b) Controlling minority of bottleneck bandwidth.

Figure 10: fShaper results using the US network setting of Table 2. The y-axis plots target allocations and achieved allocations (in parentheses) for the aggregate recipient flow and aggregate donor flow.

5.2 Standalone fShaper Evaluations

We evaluate the standalone fShaper using the network settings shown in Table 2. All flows are co-bottlenecked at the last link. We evaluate fShaper's ability to achieve three different ratios of control. For each set of ratios, we used 200 and 100 FlowTele flows for two scenarios, of which 25% are randomly selected as calibrator flows. We chose 25% for balancing the throughput variance introducing by the scaling and the proportion of controlling flows that can benefit from fAllocator. We configure recipient and donor flows in each case as: (i) *the 9:1 case*: a single recipient flow with a target share 10%, (ii) *the 3:1 case*: three recipient flows with target shares [40%, 30%, 30%], and (iii) *the 4:1 case*: 5% recipient flows with total target share 20% equally distributed among them. All donor flows are equal in each case. Cross traffic consists of multiple Cubic flows.

For each set of ratios, we considered two cases: one in which fShaper flows comprise over half of the bottleneck capacity, referred to as *majority scenario*, and one *minority scenario*. We plot the experimental results in Figure 10.

The results demonstrate that fShaper achieves accurate internal weighted fairness among FlowTele flows in all controlled cases, while maintaining the same level of external friendliness as TCP Cubic flows. Specifically, the total goodputs of FlowTele flows in all cases are very close to the Control Group where all flows are running TCP Cubic, indicating FlowTele flows in aggregate are TCP-friendly to external cross-traffic. Within the FlowTele flows, the allocation for recipient flows (in aggregate) and donor flows (in aggregate) are close to the target allocation in all cases. The donor flows in all cases achieve almost equal goodputs (with very small standard deviation among them). The recipient flows in all cases also achieve the desired weighted fair shares.

We further evaluated standalone fShaper in other network settings. All our experiments (presented in § C.2) show that fShaper can achieve desirable weighted fair shares.

No.	Content Location	User Location	RTT (ms)	Bottleneck Capacity (Gbps)	Bottleneck Type	Cross Traffic
1	Lab	Lab	~10–40	emulated by `tc`	Known	N/A
2	Chicago InstaGENI	Illinois	~12	up to 1	Dynamic	Real Users
3	Vermont InstaGENI	Texas InstaGENI	~40	up to 0.5	Dynamic	Self Generated
4	GCP Ohio	Illinois	~15	up to 1 (ISP advertise)	Known	Self Generated
5	Digital Ocean New York	Toronto	~13	emulated by `tc`	Known	FCC Traces
6	GCP Ohio	Illinois	~20	up to 1	Known	Self Generated

Table 3: The network settings used in integration evaluation.

6 EVALUATION

We implement a prototype[1] of FlowTele in roughly 9900 lines of code (mostly in Python and C++). This section evaluates FlowTele, as an integrated system, over the current regime of TCP fairshare for optimizing content provider defined metrics. We also study a co-bottleneck detection mechanism that can be used in FlowTele, as well as the system overhead.

6.1 Integration Evaluation

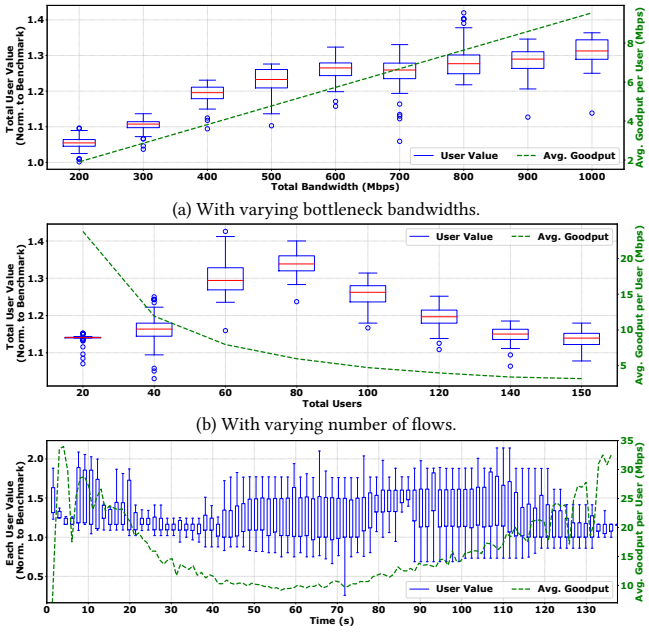
To thoroughly evaluate FlowTele, we consider a wide range of settings with different locations of content providers and users, different RTTs, different bottleneck capacities, whether the bottleneck is dynamic or known *a priori*, and different cross traffic. We summarize the settings in Table 3.

For the user-model state machine (Figure 4) parameters that are system-related and not impacted by FlowTele, we set $p_{\text{wd}} = 1$ only when a video is finished and 0 elsewhere, $p_{\text{wa}} = 1 - p_{\text{wl}} - p_{\text{wd}}$, $p_{\text{da}} = 1$ (*i.e.,* a new ad is added whenever the previous video finishes), and $p_{\text{aw}} = 1 - p_{\text{al}}$; p_{dl} is user-specific and we set it as 0.25 by default. We set $p_{\text{wl}} = p_{\text{al}}$ and compute them dynamically using our model described in § 3.2. We further study different $p_{\text{dl}}, p_{\text{wl}}, p_{\text{al}}$ in § 3.4. The control interval of vAlloc is 1 second, and fShaper updates cwnd every 0.1 second. We randomly selected 25% of FlowTele flows as calibrator flows (§ C.2 describes the choice of calibrator flow ratio). We assume videos are encoded at Constant Bit Rate using bitrates recommended by YouTube [66]. We send synthetic video chunks; a practical implementation would send chunks selected by vAlloc.

Controlled Environments. We start the evaluation in our lab where we control the entire network path from the content provider to the users (Table 3 setting 1). We impose a first-hop link capacity of 200Mbps–1Gbps using droptail buffers using the `tc` command. We ran 100 FlowTele flows. Throughout this section, we sample user/flow values from a Weibull distribution modeled after the top-10 AdWords value as described in § 3.1. We started all flows together with the same set of user values. The results are shown in Figure 11(a). As the bottleneck capacity increases, the aggregate value achieved by FlowTele also increases, reaching a median of 31% gain given a 1Gbps bottleneck. This is because FlowTele has larger optimization space with higher capacity.

In Figure 11(b), we fix the link capacity (500Mbps) and vary the number of flows. FlowTele achieves the highest gain (over 35%) when the number of flows is neither too small (uncongested, little need for optimization) nor too large (over-congested, little space for optimization).

[1]Source code available at: https://github.com/flowtele/FlowTele

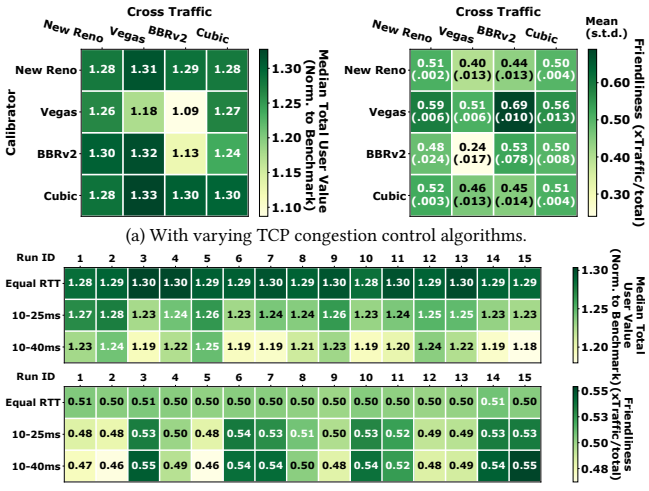


(a) With varying bottleneck bandwidths.

(b) With varying number of flows.

(c) With flows joining and leaving dynamically.

Figure 11: Aggregate user value gain achieved by FlowTele in the controlled environments.



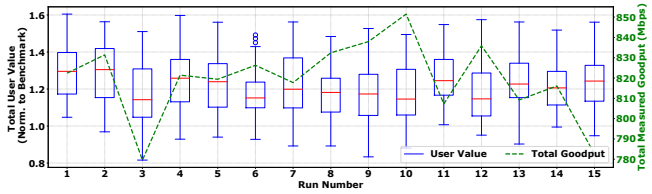
(a) With varying TCP congestion control algorithms.

(b) With varying RTTs.

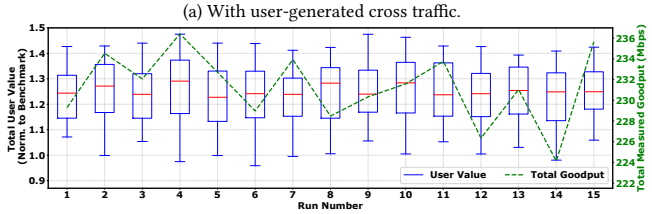
Figure 12: Performance gain and friendliness of FlowTele when interoperating with different CCAs and RTTs. "Friendliness" is the fraction of bandwidth used by cross-traffic.

In Figure 11(c), we consider users that join and leave dynamically. In particular, 5 new flows join every 5 seconds until all 100 FlowTele-flows have joined. Each flow exits after 60 seconds. The link capacity is 1Gbps. Overall, the performance gain of FlowTele increases as more flows join. The gain peaks at time 105s with roughly 60 flows. The improvement then drops as the network becomes less congested.

Figure 12 explores how FlowTele interoperates with varying TCP CCAs and RTTs. Figure 12(a) shows results from different CCAs. ("xTraffic" represents cross traffic). We ran 80 FlowTele flows and 80 cross traffic flows with New Reno, Vegas, BBRv2, and Cubic.



(a) With user-generated cross traffic.



(b) With dynamic bottleneck capacities.

Figure 13: Aggregate user value gain achieved by FlowTele over GENI.

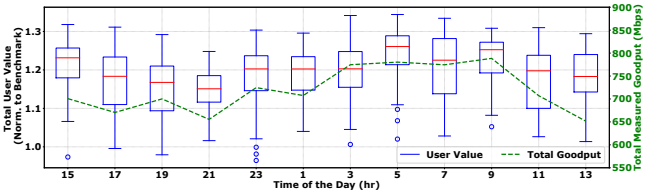
FlowTele's median gain is 30% when using Cubic as calibrator flows. When FlowTele flows and cross traffic use the same CCA, FlowTele uses 47–50% of the total bandwidth. Vegas and BBRv2 reflect lower median gains because they use delay as a congestion signal, reducing performance when competing against controlled flows.

In Figure 12(b), we consider flows with diverse RTTs. We divided the flows into 4 groups, and assign them RTTs from two sets: (10 ms, 15 ms, 20 ms, 25 ms) and (10 ms, 20 ms, 30 ms, 40 ms). We ran 80 FlowTele flows and 80 Cubic flows with 25% of them using each RTT. Increased RTT variance reduces FlowTele's performance gain, but the median gain is still 20%. Friendliness depends on the extent to which selected calibrator flows can represent the aggregate FlowTele flows.

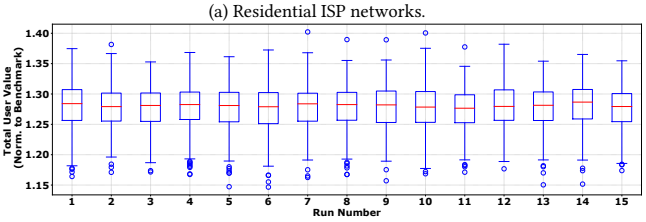
Over GENI [5]. We evaluate FlowTele on the GENI testbed [5]. First, as shown in Table 3 setting 2, we construct a path from Chicago InstaGENI to our machines in Illinois with up to 1Gbps capacity. We send 50 FlowTele flows and 50 Cubic flows across the path. We also add real user-generated cross traffic by redirecting 2 volunteers' Internet traffic through the Chicago InstaGENI using a proxy server. During the experiments, volunteers engaged in various activities, including file downloads, web searching, video streaming, and video conferencing, providing a total load of 50–100Mbps. We ran 15 runs and plot the result in Figure 13(a). The results suggest that even with very dynamic cross traffic (as shown by the varying total measured goodput), FlowTele still achieves significant performance gains, with median gains from 20–30%.

Further, we build a setting (the 3rd one in Table 3) from University of Vermont InstaGENI to University of Texas InstaGENI. Both the bottleneck and total capacity are dynamic and unknown *a priori*. Since we typically get only a few hundred Mbps using our GENI settings (default WAN and non-guaranteed peering links), we use 20 FlowTele flows and 20 TCP Cubic flows as cross traffic in this experiment. The results are shown in Figure 13(b). FlowTele again shows at least 20% gain in all 15 runs.

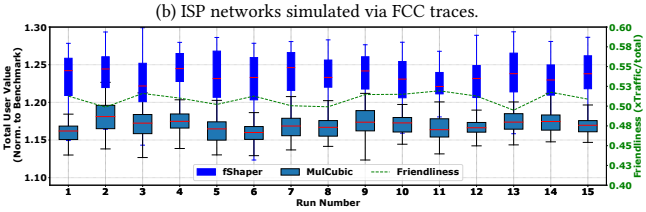
From Cloud to ISPs. We also created three settings (no. 4, 5, and 6 in Table 3) to evaluate FlowTele when the content providers are hosted by cloud VMs, and their traffic traverses ISP networks. First, we test FlowTele over a real ISP residential link with 1Gbps advertised downstream bandwidth (Table 3 setting 4). Since the cloud VMs on GCP have over 2Gbps capacity, the bottleneck is likely



(a) Residential ISP networks.



(b) ISP networks simulated via FCC traces.



(c) fShaper vs. MulCubic [45].

Figure 14: Aggregate user value gain achieved by FlowTele from Cloud to ISP networks.

located in the ISP (likely the last-mile). We use 100 FlowTele flows and 100 TCP Cubic flows of cross traffic. We ran the experiments every 2 hours for a day. The results are shown in Figure 14(a). On average, FlowTele achieves ~20% value gain throughout the day with dynamic background traffic.

In the second case of this setting (no. 5 in Table 3), we construct an ISP network using the realistic trace extracted from FCC Datasets [10]. In particular, following [65], we concatenate reports from each server-and-client pair to form 60 second blocks and group by ISP and geographical area. In total, we find 6 ISP networks as well as their served flows (details deferred to Table 4). In this part, we consider an ISP network with roughly 2Gbps capacity (serving 510 flows/users, where each user receives an advertised 16Mbps). We thus start 200 FlowTele-flows from the content providers on Digital Ocean (Table 3 setting 5) and replay the bottleneck capacity based on the bandwidth traces using the `tc` command. The results are shown in Figure 14(b).

In the final setting (no. 6 in Table 3), we compare FlowTele with an approach that uses MulCubic [45] to enforce the bandwidth allocations given by vAlloc. This demonstrates the fast convergence of fShaper. Given roughly 1Gbps bottleneck, we place 100 FlowTele flows and 100 TCP Cubic flows as cross-traffic. The results are shown in Figure 14(c). Overall, FlowTele achieves over 25% performance gain, which doubles the gain (roughly 12%) using MulCubic.

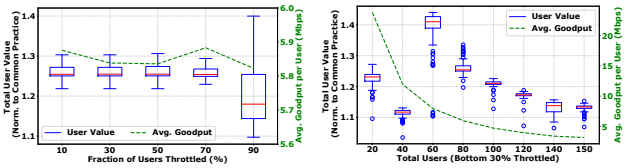
TCP Friendliness. For all settings in Table 3 with TCP Cubic cross-traffic, we report the bandwidth ratio of the aggregate FlowTele-controlled flows and cross flows in Table 5. FlowTele neither overshoots nor undershoots its fair bandwidth share.

ISP No.	Number of Flows	Total Bandwidth	Advertised Bandwidth	Average Bandwidth
A	10	25 Mbps	16 Mbps	2.5 Mbps
B	104	430 Mbps	8 Mbps	4.13 Mbps
C	288	930 Mbps	16 Mbps	3.23 Mbps
D	510	1976 Mbps	16 Mbps	3.87 Mbps
E	13	170 Mbps	14 Mbps	13.1 Mbps
F	34	590 Mbps	14 Mbps	17.35 Mbps

Table 4: Collected FCC flow traces [10].

Setting No. in Table 3	Number of Flows FlowTele / Cubic	Bandwidth Ratio Mean	Bandwidth Ratio s.t.d.
2	50 / 50	0.5527	0.0096
3	20 / 20	0.5123	0.0042
4	100 / 100	0.5001	0.0032
6	100 / 100	0.5137	0.0075

Table 5: FlowTele is TCP friendly to cross traffic.



(a) With varying fractions of users being throttled. (b) With varying number of flows.

Figure 15: Aggregate user value gain achieved by FlowTele compared with common practices [20, 60].

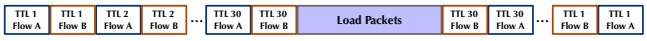


Figure 16: Measurement packet train for co-bottleneck detection, based on Pathneck [28].

6.2 Comparison with Current Practices

During the COVID-19 pandemic, some large-scale content providers (such as YouTube and Netflix) proactively throttle video quality [20, 60] to 480p to reduce the bandwidth pressure on downstream Internet links. We compare FlowTele with such practices using the first setting in Table 3. First, we evaluate different fractions of lower-valued users throttled to 480p. We set up 80 FlowTele flows, competing with 80 TCP Cubic flows of cross traffic, over the bottleneck link with 1Gbps capacity. The results are shown in Figure 15(a). FlowTele achieves roughly 25% aggregate value gain compared with this common practice. When 90% users are throttled, FlowTele's performance gain is smaller (15%) because the amount of bandwidth saved by throttling most users allows the remaining (high-value) users to all have very high video qualities (*e.g.*, 2K), leaving fewer potential optimizations for FlowTele.

We also vary the number of flows while fixing the percentage of throttled users at 30%. The results are shown in Figure 15(b). The overall performance gains are similar to Figure 11(b), showing that FlowTele outperforms static value-based throttling across a wide range of per-flow bandwidths.

6.3 Co-Bottleneck Detection

FlowTele is designed to work with any system for detecting co-bottlenecked flows. We demonstrate one design by extending Pathneck [28] to perform co-bottleneck detection between *pairs* of flows. For a single flow, Pathneck sends a packet train that contains measurement packets, load packets, and more measurement packets. The idea of Pathneck is that the load packets will buffer more at the bottleneck than at other points in the path. By measuring the time at which measurement packets induce ICMP Time Exceeded replies, Pathneck localizes the bottleneck. We create Pathneck-MF by extending Pathneck so it sends multiple sets of measurement packets (both at the beginning and the end of the packet train). For example, for two flows A and B, we create a packet train by interleaving measurement packets for A and B, then sending load packets (which could be packets for A or B), and finally interleaving measurement packets for A and B, as shown in Figure 16. When Pathneck detects the same bottleneck for flow A and flow B, we call these two flows co-bottlenecked.

No.	Type	Users on GENI (# of Nodes in a Site, Total BW, Site Name)	Content Provider BW (Mbps)	Precision	Recall
1	One downstream Bottleneck	(2, 100 , NW); (2, 200, UW)	500	1.00	0.95
2	Two downstream Bottlenecks	(2, 100, NW); (1, 100, UW); (2, 100, MI)	500	0.81	0.75
3	Content Provider Bottleneck	(2, 100, NW); (2, 200, UW)	250	1.00	0.99
4	Content Provider Bottleneck	(2, 100, NW); (1, 100, UW); (2, 100, MI)	250	1.00	1.00
5	No co-bottleneck	(1, 100, NW); (1, 100, UW)	250	0.95	0.99
6	No co-bottleneck	(1, 100, NW); (1, 100, UW); (1, 100, MI)	500	0.95	1.00

Table 6: The network settings used in co-bottleneck detection. NW, UW, MI stand for Northwestern, Wisconsin, Michigan InstaGENI, respectively.

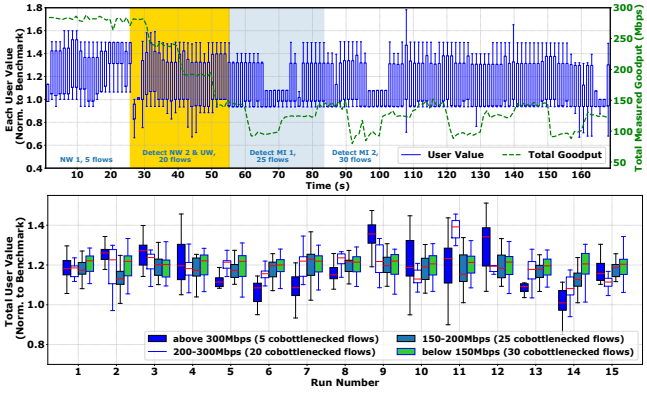


Figure 17: Per-user and aggregate-user value with dynamic co-bottlenecked flows detected by Pathneck-MF.

Detection Accuracy. We construct a controlled environment to verify Pathneck-MF. To gain ground-truth on bottleneck location, we use a content provider located in the Midwest and control the next-hop capacity using `tc`, and send measurement packet trains to multiple nodes on three GENI sites (NW, UW, and MI). We explored all source-destination pairs using `iperf` and found that at NW and MI, the two same-site nodes share a bottleneck link of 100Mbps. Each node at UW has a dedicated 100Mbps link. We consider three cases (summarized in Table 6): (i) co-bottlenecked on the downstream path (#1, 2), (ii) co-bottlenecked at the content provider (#3, 4), and (iii) no co-bottleneck links (#5, 6). We send 20 probes for each scenario, compare the results with ground truth, and present results in Table 6. Overall, Pathneck-MF demonstrates satisfactory precision and recall.

Dynamic Bottlenecks. We construct a multiple-bottleneck environment (the 2nd setting in Table 6), and run 10 FlowTele flows

CPU Usage	10 flows	20 flows	50 flows	100 flows	200 flows
fShaper	0.014%	0.032%	0.031%	0.155%	0.168%
vAlloc	0.640%	0.702%	0.732%	0.579%	0.481%
qAlloc	0.635%	0.673%	0.695%	0.659%	0.639%
Pathneck-MF	0.233%				

Table 7: CPU overhead of FlowTele on EPYC 7B12.

at each GENI site, 30 flows in total. Both NW and MI sites have two nodes, and we split flows equally between them. The content provider initially has 500Mbps upstream capacity. We dynamically change the capacity to create moving bottlenecks. In particular, initially, only the 5 flows sent to the same node at NW are co-bottlenecked at their last hop. As we reduce the upstream capacity, the bottleneck links move up. As a result, Pathneck-MF detects more co-bottlenecked flows. Eventually, all 30 flows are co-bottlenecked at the content provider's outbound link. We plot these dynamics in Figure 17: the top half is the real-time user values in specific one and the bottom half is the aggregated user values in 15 runs. It is clear that FlowTele achieves consistent gains (20% on average) given dynamic co-bottlenecked flows.

CPU Overhead. To measure CPU overhead, we use a GCP VM with 64-core, 128-thread AMD EPYC 7B12 CPUs to run FlowTele. Our VM had 4 threads and 4GB memory and sent traffic to our clients located in Illinois. We used `sar` to measure the difference between CPU utilizations after adding Pathneck-MF, fShaper, vAlloc, and qAlloc. Table 7 shows that vAlloc and qAlloc took about 0.63% of the CPU, whereas the overhead of fShaper depends on the number of flows, using 0.0025%–0.005% of the CPU per flow.

7 ETHICAL CONSIDERATIONS

Ethical Network Testing. Before releasing our traffic on the broad Internet, we ran several experiments on internal networks that demonstrated the friendliness of our traffic across a range of bandwidths and ratios of controlled traffic to cross-traffic. We also initially constrained our traffic to limited rates while we gained confidence in the friendliness of fShaper in real environments. When we used real cross traffic that we could observe in § 6.1, we did not gather any sensitive information about that cross traffic, including destination information, packet sizes, or packet patterns, and the cross-traffic was the normal traffic initiated by two of this paper's co-authors, both of whom designed the experiment and understood the setup and risks.

Ethics of Bandwidth Reallocation. While many network management tools can be used for unethical purposes, we believe that the examples illustrated here are, in many practical cases, ethical uses of bandwidth reallocation. We explore how FlowTele approaches this balancing in light of four ethical principles [62]. FlowTele does not address the principles of autonomy and beneficence, as those relate to the operation of the content provider (*e.g.,* user interface, information collection, and content selection) rather than the bandwidths selected for their delivery. FlowTele conforms with the principle of nonmaleficence by taking a fair share as compared to competing flows, while the allocators aim to promote some form of the principle of justice. Several forms of the principle of justice have been stated [62], such as "to each person an equal share" (as TCP attempts to provide), "to each person according to need" (a form of which qAlloc attempts to provide), and "to each person according to contribution" (a form of which vAlloc attempts to provide). The relative merits of each principle of justice is a philosophical question beyond the scope of this paper.

8 RELATED WORK

Traffic Shaping on Remote Bottlenecks. In general, network traffic shaping is done locally; for instance, ISPs shape their traffic to reduce peak utilization [42]. When network flows are bottlenecked on remote links, flow senders have limited shaping capabilities. Although it is possible for senders to enforce various queuing or throttling mechanism in private WAN [26, 30, 33] or datacenter networks [3, 23, 38], fine-grained shaping in open and decentralized networks is still challenging. Prior work [37, 39, 40] proposed the concept of destination-driven policies which allows senders to reallocate bandwidth across flows at Cloud/ISP middleboxes (aiming at defending against DDoS attacks). FlowTele, however, does not require intermediate deployment.

Transport Variant. Several transport protocols have recently been proposed, such as BBR [7] BBRv2 [8], PCC [16], QUIC [35], DCTCP [4] in data center networks, and Coupled Congestion Control [14, 47, 68] in multi-path networks. As discussed in § 5, fShaper is not another TCP variant. Instead, it is cross-flow bandwidth reallocator compatible with the underlying transport protocol.

Video Quality of Experience. Many studies aim to improve the QoE, including [2, 15, 29, 45, 56, 57, 65, 68]. For instance, a recent art Minerva [45], designed based on MPC [65], optimizes flow rates for QoE fairness among video streams. FlowTele can be instantiated with qAlloc (detailed in § 4) to optimize QoE fairness as well. We implement Minerva on our emulation platform, as described in § D.1, and FlowTele outperforms Minerva.

9 CONCLUSION

In this paper, we present FlowTele, the first system on which content providers can build to remotely shape traffic on Internet-scale networks in accordance with certain prioritization metrics chosen by the providers. FlowTele is designed with multiple components. First, we developed fAllocator, the brain that optimizes content-provider selected metrics by computing cross-flow bandwidth allocations. We demonstrate two fAllocator optimizers: vAlloc to maximize aggregate user value and qAlloc to minimize user QoE unfairness. Next, we design fShaper, which takes bandwidth allocations from fAllocator and enforces the target bandwidth share among flows. We implemented a prototype of FlowTele with about 9900 lines of code, and substantially evaluate FlowTele's components individually and together over various network settings.

ACKNOWLEDGMENTS

We thank our shepherd, Peter Steenkiste, and our anonymous reviewers for their thoughtful comments and numerous suggestions for improvement. This work was supported in part by NSF grant CNS-1717313, Google Cloud Research Credits award GCP203123924, and Taiwan (R.O.C.) MOE 108 GSSA scholarship. Zhuotao Liu (zhuotaoliu@tsinghua.edu.cn) and Yih-Chun Hu (yihchun@illinois.edu) are corresponding authors.

REFERENCES

[1] Aditya Akella, Srinivasan Seshan, and Anees Shaikh. An empirical evaluation of wide-area internet bottlenecks. In *ACM IMC*, pages 101–114, 2003.

[2] Zahaib Akhtar, Yun Seong Nam, Ramesh Govindan, Sanjay Rao, Jessica Chen, Ethan Katz-Bassett, Bruno Ribeiro, Jibin Zhan, and Hui Zhang. Oboe: auto-tuning video ABR algorithms to network conditions. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, pages 44–58, 2018.

[3] Wei Bai, Li Chen, Kai Chen, Dongsu Han, Chen Tian, and Hao Wang. Information-agnostic flow scheduling for commodity data centers. In *USENIX NSDI*, 2015.

[4] Stephen Bensley, Dave Thaler, Praveen Balasubramanian, Lars Eggert, and Glenn Judd. Data Center TCP (DCTCP): TCP Congestion Control for Data Centers. RFC 8257, October 2017.

[5] Mark Berman, Jeffrey S Chase, Lawrence Landweber, Akihiro Nakao, Max Ott, Dipankar Raychaudhuri, Robert Ricci, and Ivan Seskar. GENI: A federated testbed for innovative network experiments. *Computer Networks*, 2014.

[6] Khadija Bouraqia, Essaid Sabir, Mohamed Sadik, and Latif Ladid. Quality of experience for streaming services: Measurements, challenges and insights. *IEEE Access*, 8:13341–13361, 2020.

[7] Neal Cardwell, Yuchung Cheng, C Stephen Gunn, Soheil Hassas Yeganeh, and Van Jacobson. BBR: Congestion-based congestion control. *Queue*, 14(5):20–53, 2016.

[8] Neal Cardwell, Yuchung Cheng, S Hassas Yeganeh, Ian Swett, Victor Vasiliev, Priyaranjan Jha, Yousuk Seung, Matt Mathis, and Van Jacobson. Bbrv2: A model-based congestion control. In *Presentation in ICCRG at IETF 104th meeting*, 2019.

[9] Cisco. Study: Complete visual networking index (VNI) forecast. https://www.cisco.com/c/en_au/solutions/service-provider/visual-networking-index-vni/index.html, accessed in Jan 2020.

[10] Federal Communications Commission. Measuring fixed broadband - tenth report, 2021 (accessed Sep 8, 2021).

[11] ADG CREATIVE. By 2021, 82% of consumer internet traffic will be video, 2020 (accessed April 2021).

[12] Jon Crowcroft and Philippe Oechslin. Differentiated end-to-end internet services using a weighted proportional fair sharing tcp. *ACM SIGCOMM Computer Communication Review*, 28(3):53–69, 1998.

[13] Cam Cullen. Over 43% of the internet is consumed by Netflix, Google, Amazon, Facebook, Microsoft, and Apple: Global Internet Phenomena Spotlight. https://www.sandvine.com/blog/netflix-vs.-google-vs.-amazon-vs.-facebook-vs.-microsoft-vs.-apple-traffic-share-of-internet-brands-global-internet-phenomena-spotlight, Accessed on 2020.

[14] Quentin De Coninck and Olivier Bonaventure. Multipath quic: Design and evaluation. In *ACM CoNext*, 2017.

[15] Florin Dobrian, Vyas Sekar, Asad Awan, Ion Stoica, Dilip Joseph, Aditya Ganjam, Jibin Zhan, and Hui Zhang. Understanding the impact of video quality on user engagement. In *Proceedings of the ACM SIGCOMM 2011 Conference*, SIGCOMM '11, page 362–373, New York, NY, USA, 2011. Association for Computing Machinery.

[16] Mo Dong, Qingxi Li, Doron Zarchy, P Brighten Godfrey, and Michael Schapira. PCC: Re-architecting congestion control for consistent high performance. In *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15)*, pages 395–408, 2015.

[17] Simone Ferlin, Özgü Alay, Thomas Dreibholz, David A Hayes, and Michael Welzl. Revisiting congestion control for multipath TCP with shared bottleneck detection. In *IEEE INFOCOM*, 2016.

[18] Sally Floyd, Mark Handley, Jitendra Padhye, and Jörg Widmer. Equation-based congestion control for unicast applications. In *Proceedings of the conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM 2000)*, pages 43–56, August 2000.

[19] Noelle Garnier. Facebook's "transparency center" confirms what "shadow banned' accounts already knew, September 2021. https://nrb.org/articles/facebooks-transparency-center-confirms-what-shadow-banned-accounts-already-knew-2/. Accessed January 17, 2022.

[20] Hadas Gold. Netflix and YouTube are slowing down in Europe to keep the internet from breaking. CNN Business, March 2020. https://www.cnn.com/2020/03/19/tech/netflix-internet-overload-eu/index.html. Accessed June 20, 2022.

[21] Google. Google video quality report. https://www.google.com/get/videoqualityreport/. Accessed January 2020.

[22] Google. Edge nodes (google global cache, or ggc), 2021 (accessed Sep 8, 2021).

[23] Matthew P Grosvenor, Malte Schwarzkopf, Ionel Gog, Robert NM Watson, Andrew W Moore, Steven Hand, and Jon Crowcroft. Queues don't matter when you can JUMP! In *USENI NSDI*, 2015.

[24] Khaled Harfoush, Azer Bestavros, and John Byers. Measuring bottleneck bandwidth of targeted path segments. In *IEEE INFOCOM*, 2003.

[25] Sofiane Hassayoun, Janardhan Iyengar, and David Ros. Dynamic window coupling for multipath congestion control. In *IEEE ICNP*, 2011.

[26] Chi-Yao Hong, Subhasree Mandal, Mohammad Al-Fares, Min Zhu, Richard Alimi, et al. B4 and after: managing hierarchy, partitioning, and asymmetry for availability and scale in google's software-defined WAN. In *ACM SIGCOMM*, 2018.

[27] Tobias Hoßfeld, Lea Skorin-Kapov, Poul E Heegaard, and Martin Varela. Definition of qoe fairness in shared systems. *IEEE Communications Letters*, 21(1):184–187, 2016.

[28] Ningning Hu, Li Li, Zhuoqing Morley Mao, Peter Steenkiste, and Jia Wang. Locating Internet bottlenecks: Algorithms, measurements, and implications. *ACM SIGCOMM Computer Communication Review*, 34(4):41–54, 2004.

[29] Te-Yuan Huang, Ramesh Johari, Nick McKeown, Matthew Trunnell, and Mark Watson. A buffer-based approach to rate adaptation: Evidence from a large video streaming service. In *ACM SIGCOMM*, 2014.

[30] Sushant Jain, Alok Kumar, Subhasree Mandal, Joon Ong, Leon Poutievski, et al. B4: Experience with a globally-deployed software defined WAN. *ACM SIGCOMM*, 2013.

[31] Peter Kafka. YouTube 'demonetization,' explained for normals. Vox., September 2016. https://www.vox.com/2016/9/4/12795214/youtube-demonetization-explainer. Accessed January 17, 2022.

[32] Min Sik Kim, Taekhyun Kim, Yong-June Shin, Simon S Lam, and Edward J Powers. A wavelet-based approach to detect shared congestion. *IEEE/ACM Transactions on Networking*, 2008.

[33] Alok Kumar, Sushant Jain, Uday Naik, Anand Raghuraman, Nikhil Kasinadhuni, Enrique Cauich Zermeno, C Stephen Gunn, Jing Ai, Björn Carlin, Mihai Amarandei-Stavila, et al. BwE: Flexible, hierarchical bandwidth allocation for WAN distributed computing. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, pages 1–14, 2015.

[34] Kevin Lai and Mary Baker. Nettimer: A tool for measuring bottleneck link bandwidth. In *USITS*, volume 1, pages 11–11, 2001.

[35] Adam Langley, Alistair Riddoch, Alyssa Wilk, Antonio Vicente, Charles Krasic, Dan Zhang, Fan Yang, Fedor Kouranov, Ian Swett, Janardhan Iyengar, et al. The QUIC transport protocol: Design and internet-scale deployment. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, pages 183–196, 2017.

[36] Zhi Li, Anne Aaron, Ioannis Katsavounidis, Anush Moorthy, and Megha Manohara. Toward a practical perceptual video quality metric. *The Netflix Tech Blog*, 6, 2016.

[37] Zhuotao Liu, Yuan Cao, Min Zhu, and Wei Ge. Umbrella: Enabling ISPs to offer readily deployable and privacy-preserving DDoS prevention services. *IEEE Transactions on Information Forensics and Security,* 2018.

[38] Zhuotao Liu, Kai Chen, Haitao Wu, Shuihai Hu, Yih-Chun Hu, Yi Wang, and Gong Zhang. Enabling work-conserving bandwidth guarantees for multi-tenant datacenters via dynamic tenant-queue binding. In *IEEE INFOCOM*, 2018.

[39] Zhuotao Liu, Hao Jin, Yih-Chun Hu, and Michael Bailey. MiddlePolice: Toward enforcing destination-defined policies in the middle of the Internet. In *ACM CCS*, 2016.

[40] Zhuotao Liu, Hao Jin, Yih-Chun Hu, and Michael Bailey. Practical proactive DDoS-attack mitigation via endpoint-driven in-network traffic control. *IEEE/ACM Transactions on Networking*, 2018.

[41] Michael J Magazine and Maw-Sheng Chern. A note on approximation schemes for multidimensional knapsack problems. *Mathematics of Operations Research*, 9(2):244–247, 1984.

[42] Massimiliano Marcon, Marcel Dischinger, Krishna P Gummadi, and Amin Vahdat. The local and global effects of traffic shaping in the internet. In *IEEE COMSNETS*, 2011.

[43] Matthew Mathis, Jeffrey Semke, Jamshid Mahdavi, and Teunis Ott. The macroscopic behavior of the TCP congestion avoidance algorithm. *ACM SIGCOMM*, 1997.

[44] MiniMatters. The best video length for different videos on YouTube. https://www.minimatters.com/youtube-best-video-length/. Accessed Jan 2020.

[45] Vikram Nathan, Vibhaalakshmi Sivaraman, Ravichandra Addanki, Mehrdad Khani, Prateesh Goyal, and Mohammad Alizadeh. End-to-end transport for video qoe fairness. In *Proceedings of the ACM Special Interest Group on Data Communication*, SIGCOMM '19, page 408–423, New York, NY, USA, 2019. Association for Computing Machinery.

[46] Gabriel Nicholas. Shadowbanning is big tech's big problem. *The Atlantic*, April 2022. https://www.theatlantic.com/technology/archive/2022/04/social-media-shadowbans-tiktok-twitter/629702/. Accessed June 20, 2022.

[47] Costin Raiciu, Mark Handley, and Damon Wischik. Coupled congestion control for multipath transport protocols. Technical report, 2011.

[48] Rakesh Rao Ramachandra Rao, Steve Göring, Werner Robitza, Bernhard Feiten, and Alexander Raake. Avt-vqdb-uhd-1: A large scale video quality database for uhd-1. In *2019 IEEE International Symposium on Multimedia (ISM)*, pages 17–177, 2019.

[49] Vinay Joseph Ribeiro, Rudolf H Riedi, Richard G Baraniuk, Jiri Navratil, and Les Cottrell. pathchirp: Efficient available bandwidth estimation for network paths. In *Passive and active measurement workshop*, 2003.

[50] Giorgio Roffo and Alessandro Vinciarelli. Personality in computational advertising: A benchmark. In *International Workshop on Emotions and Personality in Personalized Systems at ACM RecSys (EMPIRE 2016)*, 2016.

[51] Dan Rubenstein, Jim Kurose, and Don Towsley. Detecting shared congestion of flows via end-to-end measurement. *IEEE/ACM Transactions On Networking*, 2002.

[52] Stefan Saroiu, P Krishna Gummadi, and Steven D Gribble. Sprobe: A fast technique for measuring bottleneck bandwidth in uncooperative environments. In *IEEE INFOCOM*, 2002.
[53] Brandon Schlinker, Hyojeong Kim, Timothy Cui, Ethan Katz-Bassett, Harsha V Madhyastha, Italo Cunha, James Quinn, Saif Hasan, Petr Lapukhov, and Hongyi Zeng. Engineering egress with edge fabric: Steering oceans of content to the world. In *ACM SIGCOMM*, pages 418–431, 2017.
[54] Arjun Singh, Joon Ong, Amit Agarwal, Glen Anderson, Ashby Armistead, Roy Bannon, Seb Boving, Gaurav Desai, Bob Felderman, Paulie Germano, et al. Jupiter rising: A decade of clos topologies and centralized control in google's datacenter network. *ACM SIGCOMM CCR*, 2015.
[55] Prabhakant Sinha and Andris A Zoltners. The multiple-choice knapsack problem. *Operations Research*, 27(3):503–515, 1979.
[56] Kevin Spiteri, Rahul Urgaonkar, and Ramesh K Sitaraman. Bola: Near-optimal bitrate adaptation for online videos. *IEEE/ACM Transactions on Networking*, 2020.
[57] Yi Sun, Xiaoqi Yin, Junchen Jiang, Vyas Sekar, Fuyuan Lin, Nanshu Wang, Tao Liu, and Bruno Sinopoli. CS2P: Improving video bitrate selection and adaptation with data-driven throughput prediction. In *ACM SIGCOMM*, 2016.
[58] Mark Taylor. Verizon's accidental mea culpa. Archived at https://ecfsapi.fcc.gov/file/7521706256.pdf, July 2014. Accessed on March 2020.
[59] Telecommunication Standardization Sector of ITU. Vocabulary for performance, quality of service and quality of experience. ITU-T Rec. P.10/G.100 (11/2017), 2017.
[60] Catherine Thorbecke. YouTube throttling streaming quality globally as coronavirus forces people indoors. ABC News, March 2020. https://abcnews.go.com/Technology/netflix-youtube-throttle-streaming-quality-europe-coronavirus-forces/story?id=69754458. Accessed June 20, 2022.
[61] Linus Tech Tips. Reacting to our most PROFITABLE videos! YouTube, June 2022. https://www.youtube.com/watch?v=Rh5hL47z2us&t=326s. Accessed June 20, 2022.
[62] B. Varkey. Principles of clinical ethics and their application to practice. *Medical Principles and Practice*, 30:17–28, 2021.
[63] Wenjia Wei, Yansen Wang, Kaiping Xue, David SL Wei, Jiangping Han, and Peilin Hong. Shared bottleneck detection based on congestion interval variance measurement. *IEEE Communications Letters*, 2018.
[64] Kok-Kiong Yap, Murtaza Motiwala, Jeremy Rahe, Steve Padgett, Matthew Holliman, Gary Baldus, Marcus Hines, Taeeun Kim, Ashok Narayanan, Ankur Jain, et al. Taking the edge off with espresso: Scale, reliability and programmability for global internet peering. In *ACM SIGCOMM*, 2017.
[65] Xiaoqi Yin, Abhishek Jindal, Vyas Sekar, and Bruno Sinopoli. A control-theoretic approach for dynamic adaptive video streaming over HTTP. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, pages 325–338, 2015.
[66] YouTube. YouTube recommended upload encoding settings. https://support.google.com/youtube/answer/1722171?hl=en#zippy=%2Cbitrate. Accessed October 18, 2022.
[67] YouTube. Recommended upload encoding settings. https://support.google.com/youtube/answer/1722171?hl=en, Accessed Jan 2020.
[68] Zhilong Zheng, Yunfei Ma, Yanmei Liu, Furong Yang, Zhenyu Li, , et al. Xlink: Qoe-driven multi-path quic transport in large-scale video services. In *ACM SIGCOMM*, 2021.

A QALLOC SOLVER

In this section, we present the detailed solver design to solve the optimization problem defined in § 4.2. To solve the problem, qAlloc needs to determine the target bitrate and the allocated bandwidth for the next chunk. A simple approach is to choose both $R_{1,i}$ and $\mathcal{B}_i$ directly at every control interval. However, there are two issues with such an approach.

First, the bitrate $R_{1,i}$ can not be switched to until current chunk ends, and the end of control interval does not necessarily coincide with the end of the chunk duration T_c. To more rapidly accomodate changes in available bandwidth, a small control interval provides bandwidth allocations that better reflect available bandwidth. However, since quality switches negatively affect QoE, frequent bitrate decisions are not ideal, especially when it is possible to directly control the bandwidth. Thus, we often choose a control interval smaller than T_c, allowing bandwidth to be adjusted more frequently than bitrate. Second, the chunk boundaries of all the streams may not be synchronized, and the time at which the new bitrates take effects can be quite different. Thus, it is not ideal to always make a centralized decision of $(R_{1,i}, \mathcal{B}_i)$ for asynchronous flows.

To address this limitation, qAlloc switches between two modes for each flow. In **Free Mode**, qAlloc is free to select both $R_{1,i}$ and $\mathcal{B}_i$ for flow i; whereas in **Constrained Mode**, qAlloc is constrained by the previous chosen $R_{1,i}$ and can only select bandwidth $\mathcal{B}_i$ for flow i. Free mode ends as soon as a different target bitrate is chosen (at the boundaries of control intervals), and constrained mode ends when the flow switches to the target bitrate (at the boundaries of chunks). With these two modes, the switch of bitrate and bandwidth are decoupled, and qAlloc is able to focus on the bandwidth allocation for a specific target bitrate without changing bitrate too often. Additionally, the mode information is provided to the centralized controller so qAlloc knows whether to allocate $\mathcal{B}_i$ only or jointly $(R_{1,i}, \mathcal{B}_i)$, which helps balance short term and long term optimization performance.

When a new, higher bitrate is chosen, we keep two chunks of buffer and discard the rest, to improve user-perceived quality more quickly without excessively increasing the risk of buffering; when a lower bitrate is chosen, we keep the entire buffer to avoid possible rebuffering.

We summarize the qAlloc optimization solver in Algorithm 2. At its core, the solver aims to find the largest achievable QoE given the bandwidth limit. To facilitate the exploration, it computes a qtable for each flow i that describes the lowest-bandwidth approach from any given start state ((R_0, ℓ) in Free mode and (R_0, ℓ, R_p) in Constrained mode) to achieve any given QoE, by exploring all possible quantized allocations of bitrate (from YouTube resolutions [67]) and bandwidth (from 1Mbps to 100Mbps). Different flows typically have different qtable since their available bandwidth and MOS-to-bitrate functions (*i.e.,* the $q_c()$ in Equation (2)) vary. Given the qtable, the solver performs a binary search over all possible QoEs to determine the highest achievable QoE, upon which it returns the bitrate $R_{1,i}$ and bandwidth $\mathcal{B}_i$ as the allocation for the next video chunk.

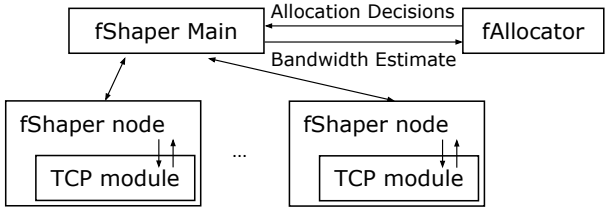


Figure 18: The architecture of fShaper.

B DETAILED DESIGNS CHOICES IN FSHAPER

B.1 fShaper Parameters

Each server inside a cluster (or a CDN) contains a fShaper node, to which the server reports RTT and packets-in-flight for each flow from `tcp_info` to *fShaper Main* every 10 milliseconds, as shown in Figure 18. The number of packets-in-flight is estimated using

```
packets-in-flight = tcp_info.tcpi_unacked
                  − tcp_info.tcpi_sacked
                  − tcp_info.tcpi_lost
                  + tcp_info.tcpi_retrans,
```

Func: qAlloc Optimization Problem:
maximize $\left\{\min_i \ \text{QoE}_{i,\text{avg}}^{K_T}\right\}$
subject to $\sum_i \mathcal{B}_i \le \mathcal{B}_{\text{total}}$ *// Constraints are simplified here*

Func: qAlloc Solver: *// Perform per-QoE allocation for flow i*
for each flow i **do** $\text{qtable}_i \leftarrow \text{BuildTable}(i)$
while $Q_{\min} < Q_{\max} - \epsilon$ *// Seek the maximum achievable QoE* **do**
  $Q \leftarrow (Q_{\min} + Q_{\max})/2$
  for *each flow i* **do**
    if a pending bitrate allocation $R_{\text{p},i}$ exists **then**
      // Constrained mode: using $R_{p,i}$ as $R_{1,i}$
      $(R_{1,i}, \mathcal{B}_i) \leftarrow \text{LookUp}(\text{qtable}_i^{\text{c}}, (R_{0,i}, \ell_i, R_{\text{p},i}, Q))$
    else *// Free mode: choose both birates and bandwidth*
      $(R_{1,i}, \mathcal{B}_i) \leftarrow \text{LookUp}(\text{qtable}_i^{\text{f}}, (R_{0,i}, \ell_i, Q))$
  if $\sum_i \mathcal{B}_i \le \mathcal{B}_{total}$ **then**
    $Q_{\min} \leftarrow Q$ *// Achievable*
    save each $(R_{1,i}, \mathcal{B}_i)$
  else $Q_{\max} \leftarrow Q$ *// Unachievable*
return most recently saved $(R_{1,i}, \mathcal{B}_i)$ as allocation for flow i

Func: BuildTable(i) *// Compute separate qtable for each flow*
for *each start state* $(R_0, \ell) \in \{BR, BUF\}$ **do**
  for *each allocation* $(R_1, \mathcal{B}) \in \{BR, BW\}$ *in order of non-increasing* $\mathcal{B}$ **do**
    // Affected by the flow's available bandwidth
    Compute expected QoE Q using Equation (2)
    $\text{qtable}_i^{\text{f}}(R_0, \ell, Q) \leftarrow (R_1, \mathcal{B})$ *// Free mode*
for *each start state* $(R_0, \ell, R_p) \in \{BR, BUF, BR\}$ **do**
  for *each allocation* $\mathcal{B} \in \{BW\}$ *in descending order* **do**
    Compute expected QoE Q using Equation (2)
    $\text{qtable}_i^{\text{c}}(R_0, \ell, R_{\text{p}}, Q) \leftarrow (R_{\text{p}}, \mathcal{B})$ *// Constrained mode*

Algorithm 2: The QoE Optimization Algorithm in qAlloc

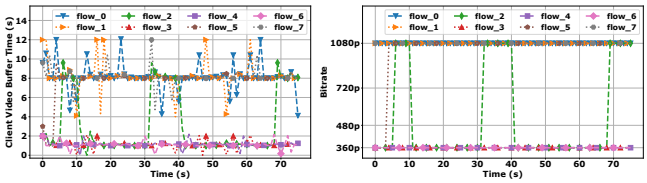
as in `tcp_packets_in_flight()`. This 100 Hz sample rate provides 10 samples of RTT and packets-in-flight per adjustment interval, which fShaper averages. fShaper Main then aggregates the estimated bandwidths and fAllocator allocates the bandwidth. We modify the TCP kernel module so that our fShaper node overwrites cwnd with the value allocated by fShaper Main in the function `tcp_congestion_ops->cong_control()`.

B.2 fAllocator with Discrete Video Rates

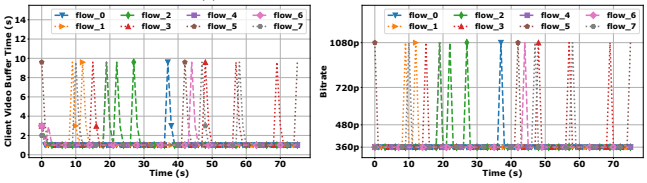
In a video environment, fAllocator chooses for each flow a (encoding bandwidth, client video buffer) pair, calculates the required bandwidth for each flow, and provides those bandwidth targets to fShaper. The determination of required bandwidth depends on the current client video buffer, which can be reported directly by the client, or the server can determine from the number of bytes acknowledged by the client, in combination with a client report of playback position and speed.

Given a discrete bitrate, the fAllocator-selected rate will either result in increased buffer, no buffer size change, or decreased buffer. For a single-flow, the fAllocator might choose a higher encoding rate for the next chunk once the buffer has been sufficiently filled, thus providing a long-term average closer to the allocated rate. In the multi-flow scenario, flows with sufficient buffer can lend their bandwidth fair-share to recipient flows, increasing the encoding rate that such recipient flows can sustainably stream. In unbuffered applications with discrete rates, FlowTele likely allows for only limited improvements, but even with small buffers (our evaluations do not select buffer sizes exceeding 20 seconds), applications can show significant improvements even when the set of possible rates are discrete.

Figure 19(a) shows the adaptation of vAlloc in practice. We randomly selected 8 flows from the 80 FlowTele flows and plot their selected resolution (encoding rate) and buffer time. In this run, the system would be application-limited if all flows were sent at 360p, and would have excessive buffering if all flows were sent at 720p. vAlloc changes both resolution and buffer time to improve the performance of higher-value flows when conditions allow.



(a) vAlloc with controlled flows.



(b) vAlloc with calibrator flows.

Figure 19: vAlloc allocations over time.

B.3 fAllocator for Video with Calibrator Flows

A calibrator flow is one regulated by a one-flow instance of fAllocator, so that the video encoding rate is selected on a per-chunk basis by fAllocator, while the cwnd is determined using a standard CCA. fAllocator uses historical goodput to determine target resolution and client video buffer time, and as client video buffer time increases or decreases, fAllocator can switch a flow between two encoding rates so that the long-term average bandwidth is slightly above the long-term average encoding rate. An example of this behavior is shown in Figure 19(b). Future work can include a more complete design for fAllocator for such scenarios, since significant changes in video encoding rate can result in lower QoE.

C STANDALONE EVALUATIONS

In this section, we report the performance gains of the standalone qAlloc and fShaper in various settings.

C.1 Standalone qAlloc Evaluations

We evaluate the standalone qAlloc using our simulated platform, which consists of emulated network links, virtual video-streaming clients and a sender, qAlloc, and a QoE monitor to measure QoE metrics. The sender and virtual clients are connected via a shared link and multiple end-user links (each end-user link connecting with one client). To emulate an end-user link, we generate the real-time link bandwidth based on the Markov model described in § 3.2. We enforce a total bandwidth limit on the shared link to control

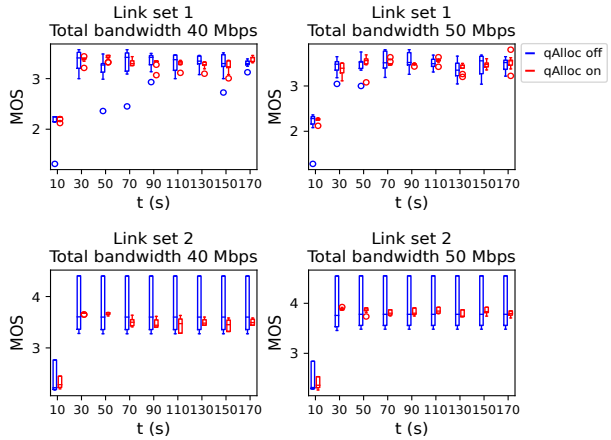


Figure 20: qAlloc results for Group One evaluation.

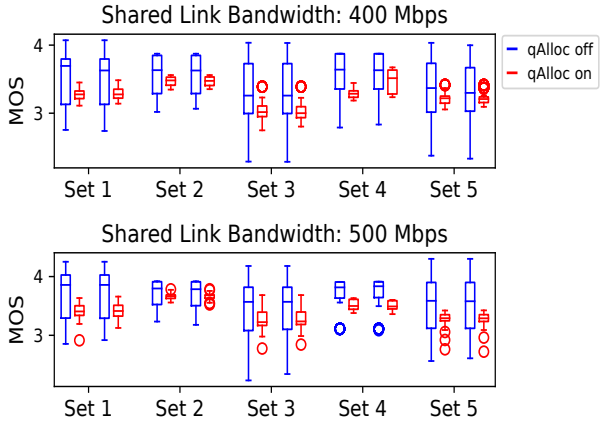


Figure 21: qAlloc results for Group Two evaluation.

Experiment Set	Number of End-User Links	Advertised Bandwidth	Bandwidth s.t.d.	Video No.
1	3	10 Mbps	0.5 Mbps	8
	4	10 Mbps	1 Mbps	8
	3	10 Mbps	2 Mbps	8
2	3	10 Mbps	1 Mbps	3
	4	10 Mbps	1 Mbps	5
	3	10 Mbps	1 Mbps	7

Table 8: Experimental setting for Group One.

Set	Number of End-User Links	Advertised Bandwidth	Bandwidth Standard Deviation
1 ~ 5	100	Uniform [10, 30] Mbps	Uniform [0, 0.2] × advertised_bandwidth

Table 9: Experimental setting for Group Two.

whether the shared link is the bottleneck or not. Each virtual client updates its buffer length based the current video bitrate and its actual achievable bandwidth, emulating video playback, stalling, and video quality changes.

To demonstrate the ability of qAlloc to improve QoE fairness by better allocating the total bandwidth, we compare QoE performance with and without qAlloc allocating the shared link bandwidth. To ensure that the comparison focuses on bandwidth allocation, the algorithm for choosing resolution remains the same (*i.e.,* both scenarios use the solver in Algorithm 2 to decide the resolution in each evaluation interval). When qAlloc does not control bandwidth allocations for the shared link, we give each flow its fair share over the shared link. We conduct the following two groups of experiments to evaluate qAlloc.

Group One. In this group, we choose two sets of end-user link and videos listed in Table 8. Both sets have 10 end-user links (*i.e.,* virtual clients), but each set has different bandwidth settings. In the first set, the bandwidth variances are different; in the second set, videos are different. We evaluate two bandwidth values for the shared link: 40Mbps and 50Mbps, and the per-flow fair share are 4Mbps and 5Mbps, both making the shared link the co-bottleneck for qAlloc to take effect. We run the simulations for 180 seconds and plot the QoEs for each 20 second interval against time in Figure 20.

Our results show that, when the shared link bandwidth is 40Mbps, qAlloc improves min q_i by 21.9% and 6.9% on average for experiment set 1 and 2, respectively; when the shared link bandwidth is 50Mbps, qAlloc improves min q_i by 7.5% and 11.3%. In the first link set, the original QoE is unstable due to the bandwidth variation, and qAlloc successfully prevents the occasional low QoE of the some worst user links through the centralized control; in the second set, the original QoE has a large deviation caused by the different video type, and qAlloc decreases the range of QoEs, resulting in reduced unfairness. This also demonstrates that qAlloc can make all flows more equal in terms of QoE, which is exactly the objective of qAlloc. Overall, the results show qAlloc's ability to improve QoE fairness when there exists either user link bandwidth variation or diversity among the video types.

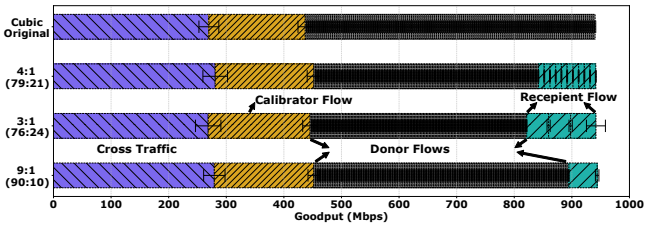
Group Two. To evaluate qAlloc in larger scenarios, we generated 100 end-user links with random advertised bandwidth, variance and video types, and repeat five sets of such experiment, as shown in Table 9. For each set of links, we perform two runs, sort the QoEs among flows, and average them across all intervals. We plot the resulting QoEs for each link set in Figure 21 with both 400Mbps and 500Mbps shared link bandwidth. qAlloc improves min q_i by 30.0% and 17.2%, respectively in the cases of 400Mbps and 500Mbps shared link bandwidth; and their QoE fairness of Hossfeld's definition [27] are improved by 18.8% and 17.1%. The performance here is significantly better than Group One as a wider range of advertised bandwidth, deviation as well as the video type are combined and thus provides more room for qAlloc to allocate the bitrate and the total bandwidth.

C.2 Standalone fShaper Evaluations

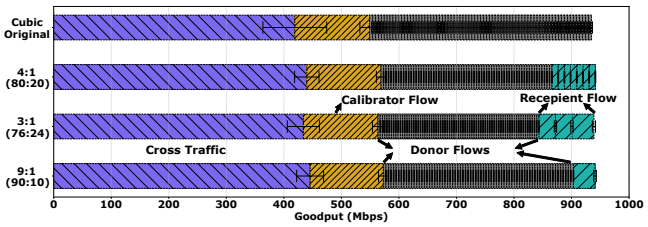
We evaluate the standalone fShaper over the Internet using various settings. The topology settings are shown in Table 2. All flows are co-bottlenecked at the last link.

Internal Weighted Fairness and External Friendliness. We repeat the experiments describd in § 3.4 using the Asia network setting of Table 2. One sight difference in the setting is we have 7 recipient flows (instead of 10) for *the 4:1 case* in the *minority scenario*. The results are plotted in Figure 22. Scenario-by-scenario results are given in Table 10. Overall, we see fShaper is effective in different network environments.

In addition, even with a rich model like TCP BBRv2 as cross traffic, fShaper can have the same level of external friendliness as



(a) Majority Scenario.



(b) Minority Scenario.

Figure 22: Evaluating fShaper for the Asia network setting. The y-axis plots the target weight allocations and the achieved allocations (enclosed in bracket) between the recipient flow and the *aggregate* donor flow.

Majority Scenario		Ratio: Self-Flows / Cross-Traffic	Ratio: donor / recipient	Among donor Flows	Among recipient Flows
the 4:1 case	Target	71.31 / 28.69 (all Cubic flows)	4 / 1	Equal Share	Equal Share
	Achieved	70.18 / 29.82	79.68 / 20.32	0.0046Mbps (Goodput s.t.d.)	0.0107
the 3:1 case	Target	71.31 / 28.69	3 / 1	Equal Share	Ratio: [40:30:30]
	Achieved	71.48 / 28.52	75.77 / 24.23	0043Mpbs	[36.52:31.64:31.82]
the 9:1 case	Target	71.31 / 28.69	9 / 1	Equal Share	One recipient
	Achieved	70.42 / 29.58	89.94 / 10.06	0.0049Mbps	One recipient
Minority Scenario		Ratio: Self-Flows / Cross-Traffic	Ratio: donor / recipient	Among donor Flows	Among recipient Flows
the 4:1 case	Target	44.82 / 55.17 (all Cubic flows)	4 / 1	Equal Share	Equal Share
	Achieved	46.73 / 53.27	79.78 / 20.22	0.0052Mbps	0.0026Mbps
the 3:1 case	Target	44.82 / 55.17	3 / 1	Equal Share	Ratio: [40:30:30]
	Achieved	46.22 / 53.78	74.28 / 25.72	0.0053Mbps	[40.00:30.01:29.99]
the 9:1 case	Target	44.82 / 55.17	9 / 1	Equal Share	One recipient
	Achieved	47.36 / 52.63	89.68 / 10.32	0.0051Mbps	One recipient

Table 10: Scenario-by-scenario fShaper results in the Asia network setting.

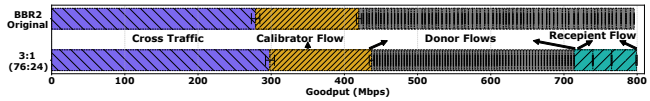


Figure 23: fShaper achieves the same level of external friendliness as TCP BBRv2 flows.

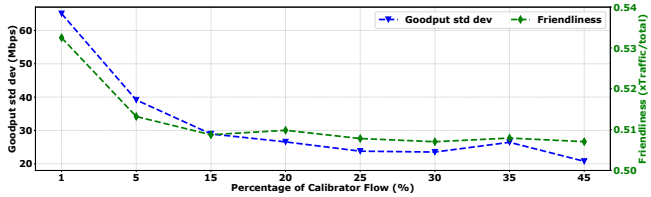


Figure 24: With varying percentage of calibrator flows.

TCP BBRv2 flows by using the same variant as calibrator flows, as shown in Figure 23.

Goodput Variance by Using Calibrator Flows. Suppose X is the goodput of each flows, so we have the variance of $V_{\text{ori}} = \mathcal{N}_{\text{total}} Var(X)$ for $\mathcal{N}_{\text{total}}$ flows. Now, since fShaper utilizes the goodput of $\mathcal{N}_{\text{calibrator}}$ calibrator flows and scales them by the factor of $\frac{\mathcal{N}_{\text{total}}}{\mathcal{N}_{\text{calibrator}}}$, we have:

$$\begin{aligned}\mathcal{N}_{\text{calibrator}} Var\left(\frac{\mathcal{N}_{\text{total}}}{\mathcal{N}_{\text{calibrator}}} X\right) &= \mathcal{N}_{\text{calibrator}} \frac{\mathcal{N}_{\text{total}}^2}{\mathcal{N}_{\text{calibrator}}^2} Var(X) \\ &= \frac{\mathcal{N}_{\text{total}}}{\mathcal{N}_{\text{calibrator}}} \mathcal{N}_{\text{total}} Var(X) \\ &= \frac{\mathcal{N}_{\text{total}}}{\mathcal{N}_{\text{calibrator}}} V_{\text{ori}}\end{aligned}$$

This means that we introduce a factor of $\frac{\mathcal{N}_{\text{total}}}{\mathcal{N}_{\text{calibrator}}}$ to the goodput variance compared with the original case. In addition, since $\frac{\mathcal{N}_{\text{total}}}{\mathcal{N}_{\text{calibrator}}} \geq 1$, using fewer calibrator flows causes higher goodput variance across controlled flows, but using more calibrator flows reduces the number of flows that can benefit from fAllocator. To evaluate this effect, we ran 80 FlowTele flows and 80 cross traffic flows traffic flows with Cubic using 1Gbps bandwidth and RTT 20 ms. We calculate the friendliness of the 80 FlowTele flows and the aggregate goodput variance experienced across FlowTele flows when measured as a time-series on 1 second intervals. These results, plotted in Figure 24, show that as we use fewer than 15% calibrator flows, goodput variance increases. We conservatively chose 25% of flows as calibrator to balance the goodput variance and performance gain of FlowTele.

Retransmission Rate. We further evaluate the retransmission rate of fShaper-controlled flows. As shown in Figure 25, these flows exhibit very similar retransmission rates as TCP Cubic flows. This indicates that although fShaper does not directly react to packet losses of individual non-calibrator flows, its periodic cwnd overwrites do not have result in excessive losses in controlled flows.

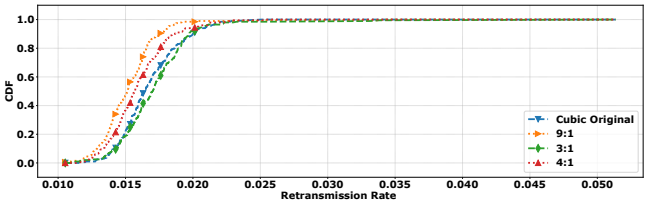


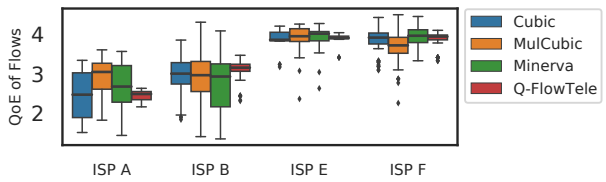
Figure 25: The retransmission rate in fShaper is stable.

D INTEGRATED EVALUATIONS

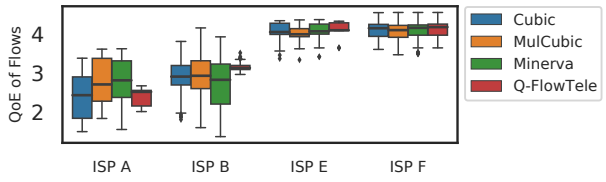
In this section, we provide the integrated evaluations for FlowTele when it uses qAlloc to optimize user experiences (which we call Q-FlowTele). We further report additional results for FlowTele when it uses vAlloc to maximize aggregate user value (which we call V-FlowTele in this section).

D.1 Q-FlowTele Evaluations

We demonstrate the performance of Q-FlowTele in real network and compare it with **MulCubic**, which uses Q-FlowTele but replaces fShaper with MulCubic, and **Minerva**, which implements Minerva's application layer algorithm [45]. To allow for a fair comparison, we model QoE using the same MOS as qAlloc instead of VMAF (though we use the same QoE chunk model), and we simulate ABR streaming for 10 chunks instead of 5. We set control interval



(a) Small user links



(b) Large user links

Figure 26: Comparing Q-FlowTele with prior arts.

of Q-FlowTele to 0.5 second, prediction horizon T to 20 seconds, and chunk duration T_c to 2 seconds.

We use the fifth network setting from Table 3, from the flow settings shown in Table 4, we use A, B, where flows are co-bottlenecked, and E, F, where flows are not co-bottlenecked. As link variation is not considered in [45], we consider two sets of user links: a "large user link" case where user links are set to 25Mbps to remove the effect of end-user link disparities, and a "small user link" case that replicates our original settings in Table 4. For each control scheme and flow setting, we conducted three runs, collected the average QoEs of each flow over time, and plotted their distribution in Figure 26.

We draw four main conclusions. First, regardless of the user link quality, Q-FlowTele has the best min-QoE as well as QoE fairness in the co-bottleneck networks, (*i.e.,* setting A and B), followed by Minerva, MulCubic, and Cubic. For instance, in setting A and B, Q-FlowTele improves min-QoE by 34.4% and 64.2%, respectively, in the "large user links" case, and improve QoE fairness by 27.1% and 21.2%, respectively. Minerva sometimes improves the median QoE, but generally fails to improve the QoE fairness. Possible reasons are: (i) Minerva performs worse with more flows (10 and 104 here); (ii) Minerva suffers significantly from the poor shaping ability of MulCubic, as shown by MulCubic's result. Second, MulCubic performs worse than Q-FlowTele, showing that MulCubic cannot achieve the fast convergence required by qAlloc. Third, the median QoE of Q-FlowTele is not as good as MulCubic and Minerva in setting A, because of the trade-off between min-QoE and the overall QoE in the optimizer (Q-FlowTele optimizes min-QoE). Finally, when the flows are not co-bottlenecked, all schemes perform similarly to Cubic, showing that Q-FlowTele, like Minerva and MulCubic, *do not reduce QoE performance when bandwidth is unconstrained.*

D.2 Additional Evaluations for V-FlowTele

In this part, we present additional evaluation results for V-FlowTele.

Bandwidth allocations and Shaping Effectiveness. We use the bottleneck capacity of ISP C trace (930Mbps) in Table 4. We sorted users by the value of showing them a single advertisement and grouped users into deciles; Figure 27 shows the goodput achieved by fShaper as compared to the bandwidth requested by vAlloc, grouped by these deciles. vAlloc provides significantly more bandwidth to the top three deciles, but provides sufficient bandwidth to the remaining users to continue to retain them. (*i.e.,* vAlloc's high-value users do not crowd out low-value users). fShaper goodput tracks these vAlloc allocations fairly closely.

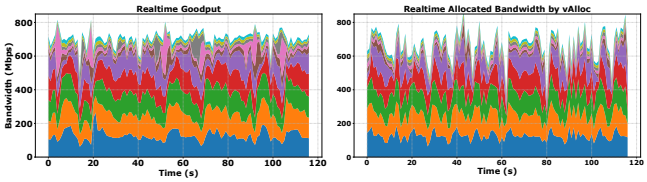


Figure 27: Realtime V-FlowTele allocated bandwidth and the actual achieved goodput, aggregated by decile.

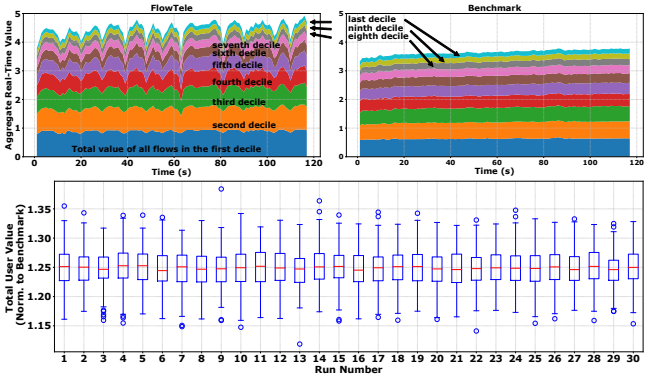
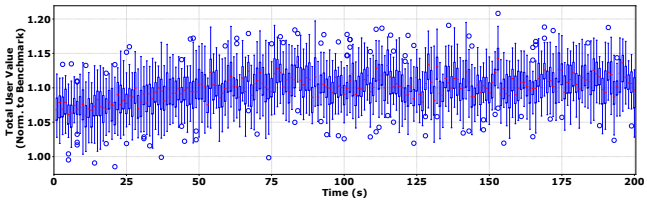
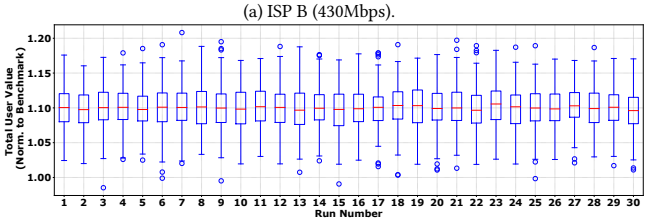


Figure 28: Realtime user values for V-FlowTele and unshaped (TCP) traffic, aggregated by decile (more-valuable users towards the bottom).



(a) ISP B (430Mbps).



(b) ISP B (430Mbps).

Figure 29: Aggregate user values achieved V-FlowTele, normalized to uncontrolled cases.

Realtime User Value Improvement. Figure 28 shows per-user real-time value with vAlloc and without any control (*i.e.,* TCP fairness), grouped by user-value decile. When controlled by V-FlowTele, total user value increases significantly. The bottom graph of Figure 28 shows that *at all times, all flows* have FlowTele-controlled

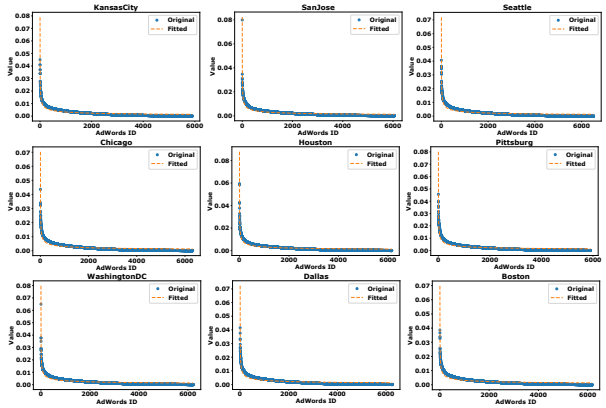


Figure 30: Additional advertisement values and their fitted curves of the Google Adwords we collected.

value at least 60% of their uncontrolled value, meaning that even the lowest-valued users experience moderate degredation of experience. Nonetheless, the top quartile of users experience user value increases of over 80%, allowing for significant improvements in total user value. Figure 29 shows the results using the ISP B (430Mbps) trace.

Fast Convergence. In Figure 14(c), we compared FlowTele with an approach that uses MulCubic [45] to enforce the bandwidth allocations given by vAlloc, demonstrating the fast convergence of fShaper. In this segment, we provide an intuitive estimate about MulCubic's convergence time to analytically support this experimental result. The expected cwnd of TCP Cubic is

$$\mathbb{E}\{\text{cwnd}_{cubic}\} = \left(\frac{3+\beta}{4(1-\beta)}\right)^{\frac{1}{4}} \left(\frac{\text{RTT}}{p}\right)^{\frac{3}{4}},$$

where p is loss probability. MulCubic assumes a deterministic loss model where the number of packets between two successive packet losses is always $\frac{1}{p}$. MulCubic adjusts β to reach the target cwnd. Suppose it takes on average n packet losses to converge to the target cwnd, then the total time of convergence is $T_{total} = \frac{n*p*1500}{B}$ (for MTU-sized packet), where B is the available bandwidth. Take $n = 1, p = 1000$ (loss rate $1e^{-3}$), $B = 2$Mbps, we have $T_{total} = 6$ seconds. However, fShaper directly overwrites and fixes the cwnd, which converges in one RTT (10–30ms). Thus, enforcing vAlloc decisions using fShaper delivers high performance gain.

E CUSTOMIZING USER MODEL IN VALLOC

In this section, we describe a method for content providers to learn their per-user models from a base model such as the one we use in § 3.2. At a very high level, our method enables a content provider to learn a shifted version of the base-model p_{wl} that is most aligned with the user history collected by the provider. Towards this end, we formulate the following optimization problem to determine the most desirable shift amount x:

$$\min_x \; -\log\Big(\prod_{\text{buf}} (\hat{p}_{\text{buf}} + x)^{\text{Leave}_{\text{buf}}} (1 - \hat{p}_{\text{buf}} - x)^{\text{TotalTime}_{\text{buf}}}\Big)$$

where $\hat{p}_{\text{buf}}$ is the p_{wl} for the given (bitrate, buffer) in the base model. Since we consider the p_{wl} curve for every bitrate as being independent, we solve the above optimization problem for one fixed bitrate in one training round, and repeat the process for every bitrate. In one training round, for every (bitrate, buf) configuration, we count how many seconds a user spent watching with the configuration during a long period of time (we use 180 simulated hours in our experiments). Each time the user leaves, we record the latest configuration. Then we count the total watch time ($\text{TotalTime}_{\text{buf}}$) and total number of leaves ($\text{Leave}_{\text{buf}}$). *Thus, the optimization is to find a shifted p_{wl} curve such that the probability of observing such watching history is maximized.* We clarify that the content providers can execute the optimization either on a per-user basis or in aggregate, allowing the providers to customize user models on different-sized subsets of users.

To avoid getting trapped in local optimum, where a partially-incorrect model results in repeated choices of a suboptimal resolution, we further design an exploration mechanism. Specifically, each time we obtain a new p_{wl} model, we will also determine a bitrate that needs to be explored during the next training round. One exploration session is a short internal we insert in a training round using a specific bitrate different from the bitrate used in the training round, which would otherwise be the one chosen by vAlloc. The exploration bitrate value is chosen to maximize the following distance:

$$\text{distance} = \sum_{\text{buf}} \frac{\hat{p}_{\text{buf}} - p_{\text{buf}}}{\sqrt{\frac{\hat{p}_{\text{buf}}(1-\hat{p}_{\text{buf}})}{n}}}$$

where p_{buf} and $\hat{p}_{\text{buf}}$ are the shifted p_{wl} model in the current and previous training round, and n is the number of samples from the curve. A large distance means that the two consecutive p_{wl} models have significant differences, indicating that our training may be stuck between two local optima. To gather the data necessary to break out of these local optima, we force the next optimization round to use the exploration bitrate.

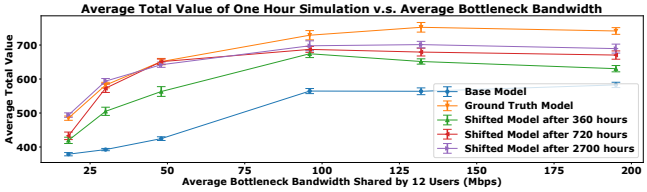
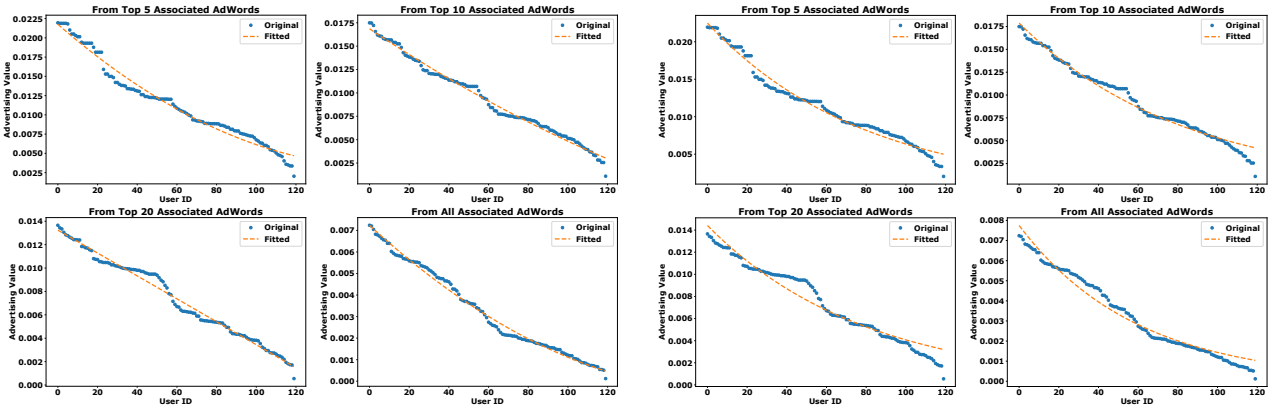


Figure 32: The total user value for customized user model. By executing the optimization process, the content providers can gradually approach the "unknown" ground truth even starting from a "bad" base user model.

We evaluate the effectiveness of our method by showing that even if a content provider starts with a user model that is very far from the ground-truth model, the content provider can gradually improve its user model. Specifically, we create several example ground truth models by randomly shifting the base p_{wl} curve and then scaling it with a factor chosen from Uniform(0.9, 1.1). Then we run our optimization process for multiple training rounds and collect the total user value for several shifted p_{wl} models. The results are plotted in Figure 32. Our results show that as we learn more about each user, the total user value for the customized user



(a) Fitting using quadratic distributions.

(b) Fitting using exponential distributions.

Figure 31: Fitting results for the value distributions of our user profiles synthesized based on the Facebook and Google datasets. Overall, the quadratic fitting shows the best coefficient of determination for this dataset.

model gradually approaches the ground truth. We are able to recover 91.4% of the optimal value (*i.e.,* using the ground truth model) after 720 simulated hours, which improves to be 94.0% after 2700 simulated hours. More crucially, our method works better in the lower bandwidth region where vAlloc is likely to show the most improvement. When the average bottleneck is about 40% of "threshold value" (where increasing bandwidth will not improve total value), our method achieves 98.6% and 99.9% of the optimal user value, after 720 and 2700 simulated hours, respectively. Thus, even if a content provider starts with a base model that is inconsistent with its actual user base, it can still gradually converge the true model by analyzing user watching histories.

F USER VALUE DISTRIBUTION STUDY

This section presents additional results on user value distributions omitted from the body of the paper due to space constraints. Figure 30 shows the value distributions of Google AdWords we collected in nine additional cites. Figure 31 plots the quadratic and exponential fitting results for our synthesized user profiles based on Facebook and Google datasets for different advertising strategies. The quadratic fitting shows the best coefficient of determination for this dataset, so we use it as one of our representative value distributions for vAlloc and integrated evaluation.

G PUBLIC POLICY IMPLICATIONS

Beyond the scope of this paper are any *public policy* implications; for example, the application of ordinary corporate income tax, or a specially-designed traffic-shaping tax, could provide for a socially-equitable distribution of the additional profits attributable to these shaping techniques. In particular, if high-quality video delivery is generally reduced in one group and increased in a second group, the proceeds of a traffic-shaping tax can provide benefits to the first group that are, from their perspective, *even more meaningful* than an equal probability of high-quality video delivery, such as reduced Internet prices or increased community services, thus making revenue-based shaping a win-win for all groups of consumers. This paper focuses on the *network mechanisms* that can enable an ecosystem combining technical approaches, policy decisions, and social programs that enable these mutually beneficial outcomes.