Improving User QoE for Residential Broadband: Adaptive Traffic Management at the Network Edge

Felix Ming Fai Wong*, Carlee Joe-Wong*, Sangtae Ha†, Zhenming Liu*, Mung Chiang*
*Princeton University, Princeton, NJ, USA †University of Colorado, Boulder, CO, USA
{mwthree, cjoe, chiangm}@princeton.edu, sangtae.ha@colorado.edu, zhenming@cs.princeton.edu

***Abstract*—Recent increases in network traffic have led to severe congestion in broadband networks. We propose to mitigate this problem with a two-level edge-based solution that incentivizes users to moderate their bandwidth usage based on their actual needs. In the first level, home gateways are given QoE (quality of experience) credits that they can spend to receive more bandwidth at congested times; to ensure fairness, the credits are redistributed to other gateways after they are spent. We show that this scheme guarantees long-term fairness and maximizes users' total satisfaction at the equilibrium. In the second level, each gateway allocates bandwidth among its users and apps according to its own priorities. Gateways can thus customize their bandwidth allocation depending on individual preferences. We develop a prototype of this second-level allocation on commodity wireless routers. We then consider an example scenario and show by simulation and implementation results that our solution outperforms an equal bandwidth allocation, increasing users' overall utility and fairly allocating bandwidth across users.**

I. Introduction

A. Motivation: Demand in Broadband Networks

Recently, ISPs have seen a large, sustained increase in data usage, driven by popular streaming and cloud services such as Netflix and Dropbox [1]. While many strategies for managing this demand have been proposed [2], few works explicitly consider wired cable networks. Broadband providers have themselves considered two measures to manage congestion: content-agnostic bandwidth (network capacity) distribution based on usage history [3] and deep packet inspection to throttle "abusive" users, *e.g.,* those running BitTorrent. However, these solutions ignore users' true quality of experience (QoE), which introduces two key challenges:

Incentivizing responsible usage. Simply throttling heavy users does not always improve users' QoE, *e.g.,* when everyone overloads the network by streaming HD videos at the same time. Yet in today's networks, users have no incentive to moderate their bandwidth consumption only to the amount they actually need, at times when they need it.

Addressing different bandwidth needs. The same bandwidth rates can yield different QoE levels for different types of traffic. Yet users today cannot communicate their bandwidth needs to the ISP without telling ISPs which content they are consuming, violating their privacy and network neutrality. Consider, for instance, two neighbors who both use a substantial amount of bandwidth in the evening. One of them watches Netflix, and the other backs up large files from work.

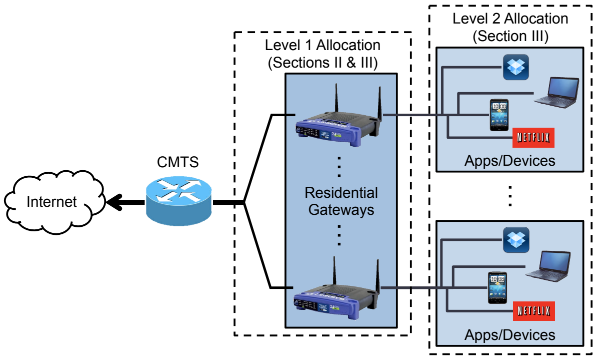


Fig. 1. Hierarchical edge-based bandwidth allocation.

The optimal solution would be to prioritize the first neighbor during the evening, and incentivize the second neighbor to back up his files a few hours later. Yet in today's networks, the broadband provider might simply give both equal bandwidth.

B. A Home Solution to a Home Problem

We argue that broadband providers can fully account for user QoE by pushing congestion management to the network edge at home gateways.[1] Thus, we *empower the user* to improve his or her own experience in using the network. We propose to allocate bandwidth in a two-level hierarchy, as shown in Figure 1. Bandwidth is first allocated among home gateways and then allocated locally among the users and devices connected to each gateway. While our solution can apply to any broadband network, we maintain a particular focus on cable networks, which have recently suffered from particularly egregious congestion [3].

Level 1: Our Level 1 solution distributes *QoE credits* [4] to the gateways. Each gateway uses its credits to "purchase" guaranteed bandwidth rates at congested times, giving it an incentive to moderate network usage due to its limited credit budget. We limit the total bandwidth demand to the network capacity by fixing the total number of credits in the system and recirculating credits to gateways as they are spent.

The ISP divides the day into a series of discrete time periods, *e.g.,* each lasting an hour, and designates some as "congested." At such times, traffic is divided into two classes: a first-tier class that gateways must purchase with credits and a second-tier class that requires no credits but is always of lower

[1]A gateway in this paper refers to the combination of a broadband modem and an in-home WiFi access point.

priority.[2] This scheme ensures that the network is fully utilized if there is sufficient demand, yet still encourages gateways to spend credits at different times.

At the start of the period, each gateway first decides how many of its credits to spend, *i.e.,* how much guaranteed bandwidth to purchase.[3] A central server in the ISP's network records the total credits spent by each gateway and redistributes the appropriate number of credits to each gateway in the next time period. Each gateway updates its budget by deducting the credits spent in the previous time period and adding the number of credits received. In the next time period, each gateway then knows its updated budget and can again choose how many credits to spend.

Level 2: Our Level 2 solution allows gateway users to prioritize different apps and automatically allocates bandwidth accordingly.[4] One user, for instance, might prioritize streaming music, while another might prioritize file transfers. The gateway then divides its purchased bandwidth among these apps according to their priorities. We focus on *elephant traffic*, which tends to be non-bursty and amenable to bandwidth throttling. Since the allocation runs locally at each gateway, the user has full control over these decisions.

Prioritizing different applications requires both automatically classifying sessions entering the gateway into different apps and enforcing rate limits for each app. While standard mechanisms are available for doing so in a router, they generally require static priority configurations and server-side support when prioritizing downlink traffic. We thus develop our own classification and rate limiting solutions, which run locally on each gateway, and prototype them in a modified home gateway router.

C. Practical Advantages

We show that our credit-based edge solution is both theoretically sound and practically deployable, by first analyzing gateways' credit spending behavior and then presenting a prototype implementation on a modified home gateway. Our solution has the following advantages for users and ISPs:

Fairness across gateways: Credits are circulated back to each gateway in a way that depends on other gateways' behavior. Over time, every gateway will be able to use a fair portion of the bandwidth, as gateways that spend a lot of credits in one time period will have fewer to spend later.

User-driven QoE optimization: Each gateway spends credits so as to maximize its own overall satisfaction or QoE, and is free to allocate this purchased bandwidth among its apps and devices. Our credit redistribution mechanism ensures that gateways' credit spending choices optimize the collective social welfare, *i.e., all* gateways' satisfaction, over time.

Session-level granularity: Within each gateway, users can allocate bandwidth to different apps according to their specific needs and priorities, ensuring that all apps have satisfactory QoE. More centralized solutions generally cannot optimize bandwidth at the session level, as this violates user privacy.

Privacy preservation: We design an algorithm that runs at individual gateways and utilizes *only the information on how the credits are circulated* to find the optimal spending. Thus, users need not reveal their individual QoE preferences to other gateways or to the ISP.

Scalability: A solution at the gateway level naturally allows for a distributed bandwidth allocation, since each gateway decides how to spend its credits and allocate its bandwidth. It can be easily deployed via modified home gateways and does not require substantive changes to ISPs' network architecture.

Incremental deployability: Since the number of total credits is fixed, we can incrementally deploy the solution by starting with a small number of credits and introducing more as more gateways begin to participate.

We introduce our credit redistribution algorithm in Section II and show that gateways' total satisfaction is maximized in equilibrium. In Section III, we present practical algorithms for each gateway to decide how many credits to spend and to distribute bandwidth among its apps and devices. We present our prototype implementation on a home gateway router in Section IV and show simulation and implementation results of an example scenario in Sections V and VI. We briefly discuss related works in Section VII before concluding in Section VIII.

II. Credit Distribution and Optimal Spending

In this section, we describe the bandwidth allocation at the higher level of Figure 1. We first describe our system of credits for purchasing bandwidth (Section II-A) and show that it satisfies several fairness properties. We then show that even if each gateway selfishly maximizes its own satisfaction, the total satisfaction across all gateways can be maximized (Section II-B). All proofs are in Appendices A–F.

A. Credit Distribution

We divide congested times of the day into discrete time periods, *e.g.,* of a half-hour duration, and allow gateways to "purchase" bandwidth in each period. The spent credits are redistributed at the end of each period. Users' credit budgets at the end of each day carry over into the next day, so our model is not affected by these time gaps in credit spending.

We suppose that a fixed number $B = \beta C$ of credits is shared by n different gateways, where C is the network capacity in Gbps and β an over-provisioning factor chosen by the ISP. We consider $T+1$ time periods indexed by $t = 0, 1, \ldots, T$, *e.g.,* $T+1$ periods per week. We use b_{it} to denote the budget, *i.e.,* number of credits owned, of gateway i at time t, and we suppose that the total credits are initially distributed equally across gateways, *i.e.,* $b_{i0} = B/n$ for all i. For brevity, in the remainder of the paper we use "budget" to mean "credit budget," or the number of credits available to the gateway at a given time. We use x_{it} to denote the number of credits used

[2]By spending credits, users essentially sort their first-class traffic into customized classes defined by the different guaranteed bandwidths purchased.

[3]In practice, an automated agent acting on behalf of the gateway's users makes this decision, with possible manual overrides.

[4]Should explicit prioritization prove too complex for average users, we can introduce default priorities for different types of apps and devices.

by gateway i in time period t. We then update each gateway i's budget as

$$b_{i,t+1} = b_{it} - x_{it} + \frac{1}{n-1}\sum_{j\neq i} x_{jt}, \tag{1}$$

where we sum over all gateways j except gateway i. Each gateway i is constrained by $0 \leq x_{it} \leq b_{it}$: it cannot spend negative credits, and the number of credits spent cannot exceed its budget. This credit redistribution scheme conserves the total number of credits for all times t:

Lemma 1: At any time t, the number of credits distributed among gateways is fixed, *i.e.,* $\sum_{i=1}^{n} b_{it} = B = \beta C$.

Since users cannot spend more than their budgets, their total bandwidth purchases are therefore limited to at most βC.

Heavy gateways are prevented from hogging the network, as a large x_{jt} (i.e., large usage by gateway j at time t) simply means that the other gateways $i \neq j$ will receive larger budgets in the time interval $t+1$. This natural fluctuation in credit budgets enforces a form of *fairness across gateways*. In fact, if this redistribution leads back to a previous budget allocation, all gateways spend the same number of credits:

Lemma 2: Suppose that for some times s and t, $b_{is} = b_{it}$ for all gateways i, *e.g.,* $s = 0$ and $b_{it} = B/n$. Then each gateway spends the same number of credits between times s and t: for all gateways i and j, $\sum_{\tau=s}^{t-1} x_{i\tau} = \sum_{\tau=s}^{t-1} x_{j\tau}$.

Using this result, we can more generally bound the difference in the number of credits gateways can spend:

Proposition 1: At any time t, for any two gateways i and j, $\left|\sum_{s=0}^{t} x_{is} - \sum_{s=0}^{t} x_{js}\right| \leq B(n-1)/n$. Thus, the time-averaged difference in spending

$$\lim_{t\to\infty} \frac{1}{t}\left|\sum_{s=0}^{t} x_{is} - \sum_{s=0}^{t} x_{js}\right| \leq \lim_{t\to\infty} \frac{B(n-1)}{nt} = 0. \tag{2}$$

Over time, fairness is enforced in the sense that all gateways can spend approximately the same number of credits.

Though these fairness results limit heavy gateways' hogging the network, gateways with less usage may conversely "hoard" credits, hurting other gateways' budgets. To limit hoarding, we cap each gateway's budget at a maximal value of $\overline{B}$, with $B/n < \overline{B} \leq B$.[5] For instance, the ISP might choose $\overline{B} = \frac{B}{n-m+1}$, where m is the minimum number of gateways on the network at any given time. The $n-m$ inactive gateways at that time can then hoard at most $B(n-m)/(n-m+1)$ credits, letting active gateways use the remaining $B/(n-m+1)$ credits.

To enforce this budget cap, the excess budget $\left(b_{it} - x_{it} + \sum_{j\neq i}\frac{x_{jt}}{n-1}\right) - \overline{B}$ of any gateway i exceeding the cap is evenly distributed among all gateways below the cap. Should these credits push any gateway over the cap, the resulting excess is evenly redistributed to the remaining gateways until all budgets are below the cap. Since $\overline{B} > B/n$ and we reallocate to fewer gateways after each iteration, this process converges after at most $n-1$ iterations. We expect that users will rarely reach the budget cap, as even without the cap, no single gateway can hoard all available credits:

Proposition 2: Let $\alpha = (n-2)/(n-1)$ and suppose that a given gateway i uses at least ϵ bandwidth every p periods, where $B/n > \epsilon \geq 0$ and p may denote, *e.g.,* one day. Then at any time t, gateway i's budget

$$b_{it} \leq \frac{B}{n}\alpha^{t+1} + B\left(1-\alpha^{t+1}\right) - \epsilon\left(\frac{\alpha^p - \alpha^{p\left(1+\left\lfloor\frac{t+1}{p}\right\rfloor\right)}}{1-\alpha^p}\right)$$
$$\to B - \frac{\epsilon\alpha^p}{1-\alpha^p} \tag{3}$$

as $t\to\infty$. Thus, if $\epsilon > 0$, $b_{it} < B$. Moreover, at any fixed time t, at most one gateway can have a budget of zero credits.

For instance, if a gateway spends ϵ credits at each time, then as $t\to\infty$, $b_{it} \leq B - \epsilon(n-2)$: if ϵ is relatively large, a gateway hoards fewer credits, since these are redistributed among others once spent. Conversely, a gateway that spends very little can asymptotically hoard almost B credits. More broadly, if a number m of gateways are inactive in a network for a certain number of time periods s, then we can bound the number of credits these m gateways accumulate:

Proposition 3: Suppose that m gateways are inactive from times 0 to $s-1$ (*i.e.,* for s periods). Then the number of credits that these gateways can accumulate by time s is given by

$$\sum_{i=1}^{m} b_{is} - b_{i0} \leq \left(1 - \left(\frac{n-2}{n-1}\right)^s\right)\left(B - \sum_{i=1}^{m} b_{i0}\right) \tag{4}$$

where we index the inactive gateways by $i = 1, 2, \ldots, m$.

B. Optimal Credit Spending

Given the above credit distribution scheme, each gateway must decide how many credits to spend in each period. To formalize this mathematically, let U_{it} denote gateway i's utility as a function of the guaranteed bandwidth x_{it} in time interval t. Though gateways may increase their utilities with second-tier traffic, we do not consider this traffic in our formulation. Second-tier bandwidth is difficult to predict: gateways could only obtain historical information on its availability by regularly sending such traffic, which they are unlikely to do.

We consider a finite time horizon T, *e.g.,* one week, since the utility functions cannot be reliably known far into the future. Each gateway i then optimizes its total utility from the current time s to $s+T$:

$$\max_{x_{it}} \sum_{t=s}^{s+T} U_{it}\left(x_{it}\right), \text{ s.t. } 0 \leq x_{it} \leq b_{it},\ \forall t. \tag{5}$$

Here the budgets b_{it} are calculated using the credit redistribution scheme (1), with appropriate adjustments to enforce the budget limit $\overline{B}$. For ease of analysis, we do not model these budget caps here. In practice, the ISP can cap gateways' budgets for each time period during the credit redistribution.

[5] If $\overline{B} = B/n$, then we would have $b_{it} = B/n$ for all gateways i at all times t. We therefore take $\overline{B} > B/n$ to ensure that there is a feasible set of budgets $\{b_{it}\}$ with each $b_{it} \leq \overline{B}$.

We first note that the budget expressions (1) can be used to rewrite the inequality $x_{it} \leq b_{it}$ as the linear function

$$\sum_{\tau=s}^{t} x_{i\tau} - \sum_{j \neq i} \sum_{\tau=s}^{t-1} \frac{x_{j\tau}}{n-1} \leq b_{i0}. \tag{6}$$

Thus, if the U_{it} are concave functions, then given the amount spent by other gateways $x_{j\tau}$, (5) is a convex optimization problem with linear constraints.[6]

Since each gateway chooses its own x_{it} to solve (5), these joint optimization problems may be viewed in a game-theoretic sense: each gateway is making a decision that affects the utilities of other gateways. From this perspective, the game has a Nash equilibrium at the social optimum:

Proposition 4: Consider the global optimization problem

$$\max_{x_{it}} \sum_{i=1}^{n} \sum_{t=s}^{s+T} U_{it}\left(x_{it}\right), \ \text{ s.t. } 0 \leq x_{it} \leq b_{it}, \ \forall i, t \tag{7}$$

with the credit redistribution (1) and strictly concave U_{it}. Then an optimal solution $\{x_{it}^*\}$ to (7) is a Nash equilibrium.

While Prop. 4's result is encouraging from a system standpoint, in practice this Nash equilibrium may never be achieved. Since the gateways do not know each others' utility functions, they do not know how many credits will be spent and redistributed at future times, making the future credit budgets unknown parameters in each gateway's optimization problem. These must be estimated based on historical observations, which we discuss in the next section.

III. An Online Bandwidth Allocation Algorithm

We now consider a gateway's actions at both levels of bandwidth allocation. We first give an algorithm to decide credit spending (Level 1) and then show how the purchased bandwidth can be divided at the gateway (Level 2). Using Algorithm 1, each gateway iteratively estimates the future credits redistributed, chooses how many credits to spend, prioritizes apps, and updates its credit estimates. We assume the gateway's automated agent knows its users' utility functions.

A. Estimating Other Gateways' Spending

To be consistent with (5)'s finite time horizon, we suppose that gateways employ a *sliding window* optimization. At any given time s, gateway i chooses rates for the next T periods $s, \ldots, s+T-1$ so as to maximize its utility for those periods. At time $s+1$, the gateway updates its estimates of future credits redistributed and optimizes over the next T periods, etc.

We use *scenario optimization* to estimate the number of credits each gateway will receive in the future.[7] Scenario optimization considers a finite set S_i of possible scenarios for each gateway i, associating each scenario $\sigma \in S_i$ with a probability π_σ that it will take place. Computing the credit

[6]The assumption of concavity, *i.e.,* $U_{it}''(x_{it}) < 0$, may be justified with the economic principle of diminishing marginal utility as bandwidth increases.

[7]This technique is often used in finance to solve optimization problems with stochastic constraints that are hard to predict, *e.g.,* market dynamics [5].

Algorithm 1 Gateway spending decisions.

```
$s \leftarrow 1$ {$s$ tracks the current time.}
while $s > 0$ do
    if $s > 1$ then
        Update estimate of future amounts redistributed using Algorithm 2.
    end if
    Calculate $\sum_{j \neq i} x_{jt}/(n-1)$ for $t = s, \ldots, s+T-1$.
    Solve (5) with budget constraints (9) given $\sum_{j \neq i} x_{jt}/(n-1)$.
    Choose the application priorities $\mu_k$ by solving (10).
    $s \leftarrow s + 1$
end while
```

redistribution and optimal spending x_{it} for each σ then yields a probability distribution of the possible credits spent. In our case, a "scenario" is a set of utility functions $\{U_{jt}\}$ for the other gateways. We parameterize these scenarios by noting that gateways' utilities depend on the application used, *e.g.,* streaming versus downloading files. We consider K different applications and define $u_k(x)$ as the (pre-determined) utility from an application of type k (*e.g.,* $k = 1$ corresponds to streaming, $k = 2$ to file downloads, etc.). We thus take

$$U_{jt} = \gamma_{jt} \sum_{k=1}^{K} p_{jt}^k u_k \tag{8}$$

for each gateway j, where γ_{jt} is a scaling factor specified by individual gateways. The variable p_{jt}^k denotes the (estimated) probability that gateway j optimizes its usage with the utility function u_k, *e.g.,* if app k is used the most at time t.

With this utility definition, we can define a scenario σ by the coefficients $\gamma_{jt}(\sigma)$ and $p_{jt}^k(\sigma)$ of gateways' utility functions. Since gateway i cannot distinguish between other gateways, it need only estimate their behavior in aggregate. These gateways can be thus viewed as one "gateway" j by adding their utility functions and budget constraints. Gateway j then maximizes

$$U_{jt} = \sum_{t=1}^{T} \gamma_{jt}(\sigma) \sum_{k=1}^{K} p_{jt}^k(\sigma) u_k$$

subject to the budget constraints $0 \leq x_{jt} \leq b_{jt}$, where the coefficients $\gamma_{jt}(\sigma) p_{jt}^k(\sigma)$ represent the added coefficients for all gateways $\neq i$. Since gateway i cannot know the accuracy of its or gateway j's estimates of future usage, for the purpose of estimation we assume that both gateways' future usage estimates are correct. Thus, following Prop. 4, all gateways choose their usage so as to maximize the collective utility $\sum_t \left(U_{jt} + U_{it}\right)$ subject to the budget constraints. This optimization may be solved to calculate the credits $\sum_{j \neq i} x_{jt}(\sigma)/(n-1)$ redistributed to user i at each time t in scenario σ.

To improve our credit estimates, at each time t we update the scenario probabilities π_σ by comparing the observed number of credits redistributed at time $t-1$, denoted by $\sum_{j \neq i} \overline{x}_{j,t-1}/(n-1)$, with the estimated amount redistributed $\sum_{j \neq i} x_{j,t-1}(\sigma)/(n-1)$ for each $\sigma \in S_i$. We suppose that gateways' behavior is sufficiently periodic (*e.g.,* over $T =$ one week) for the π_σ at times t and $t+T$ to be the same.

Algorithm 2 Estimating credit redistribution.

```
$s \leftarrow 1$ {$s$ tracks the current time.}
while $s > 0$ do
    for all gateways $i = 1, \ldots, n$ do {this loop may be run in parallel}
        Choose scenarios $S_i$.
        for each scenario $\sigma \in S$ do
            Calculate the predicted amount redistributed $\sum_{j \neq i} x_{jt}(\sigma)/(n-1)$ for $t = s, \ldots, s+T-1$, assuming other gateways know $x_{it}$ for all $t$.
            if $s > 1$ then
                Update probability $\pi_\sigma$ using Bayes' Rule.
            end if
        end for
    end for
end while
```

We use $P\left(\sum_{j \neq i} \overline{x}_{j,t-1} = \sum_{j \neq i} x_{j,t-1}(\sigma)\right)$ to denote the probability that, given $\sum_{j \neq i} \overline{x}_{j,t-1}/(n-1)$ credits redistributed to gateway i, gateways $\neq i$ use scenario σ's utility function at time t. We find these probabilities with the L_2 discrepancy between estimated and observed credits redistributed:

$$P\left(\sum_{j \neq i} \overline{x}_{j,t-1} = \sum_{j \neq i} x_{j,t-1}(\sigma)\right) = \frac{1}{|S_i| - 1}\left(1 - \frac{\left(\sum_{j \neq i} \overline{x}_{j,t-1} - \sum_{j \neq i} x_{j,t-1}(\sigma)\right)^2}{\sum_{l=1}^{|S_i|}\left(\sum_{j \neq i} \overline{x}_{j,t-1} - \sum_{j \neq i} x_{j,t-1}(l)\right)^2}\right).$$

We then update the scenario probabilities π_σ using Bayes' rule and use the new π_σ in Algorithm 2.

B. Online Spending Decisions and App Prioritization

Algorithm 1 shows how the credits spent in different scenarios are incorporated into choosing a gateway's rates x_{it} and application priorities. Each gateway constrains its spending depending on the estimated redistributed credits: for instance, a conservative gateway might choose the x_{it} so that the budget constraints $0 \leq x_{it} \leq b_{it}$ hold for all scenarios. In the discussion below, we suppose that gateways constrain the x_{it} so that (6) holds in expectation:

$$\sum_{\tau=s}^{t} x_{i\tau} - \sum_{\sigma \in S_i} \pi_\sigma \left(\sum_{j \neq i} \sum_{\tau=s}^{t-1} \frac{x_{j\tau}(\sigma)}{n-1}\right) \leq b_{i0}. \qquad (9)$$

We also constrain $b_{it} \leq \overline{B}$, *i.e.*, the expected budget at a given time cannot exceed the budget cap: users would rather spend more credits to remain under the budget cap than be forced to redistribute excess credits to other gateways.

Each gateway can further improve its own experience with its Level 2 allocation: dividing the purchased bandwidth among its apps. It does so by assigning priorities to different devices and applications, so that higher priority apps receive more bandwidth. Since users cannot be expected to manually specify priorities in each time period, we introduce an automated algorithm that leverages the gateway's known utility functions (8) to optimally set application priorities.

We consider the K application categories in (8) and use μ_k to represent each category k's priority. Since the applications

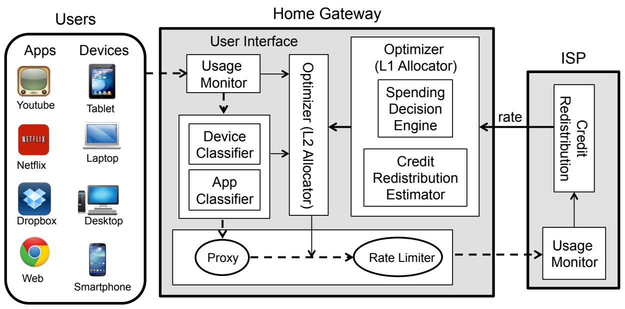


Fig. 2. System architecture. Dashed lines represent traffic flow, and solid lines represent rate and credit information.

active at a given time may change during a period, *e.g.*, if a user starts or stops watching a video, we define an app's priority in relative terms: for any apps k_1 and k_2, $\mu_{k_1}/\mu_{k_2} = y_{k_1}/y_{k_2}$, where y_k is the bandwidth allocated to application k and $\sum_k y_k = x_{it}$, ensuring that all the purchased bandwidth is used. We normalize the priorities to sum to 1: $\sum_k \mu_k = 1$.

Since it is difficult to predict which apps will be active at a given time, we choose the app priorities μ_k according to a "worst-case scenario," in which all apps are simultaneously active. In this case, each app k receives $y_k = \mu_k x_{it}$ bandwidth, and we choose the μ_k to maximize total utility:

$$\max_{\mu_k} \sum_{j=1}^{K} u_k\left(\mu_k x_{it}\right), \text{ s.t. } \sum_{k=1}^{m} \mu_k = 1. \qquad (10)$$

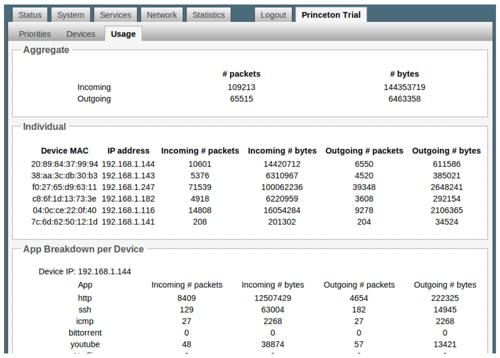
Since each function u_k is assumed to be concave and the constraint is linear in the μ_k, (10) is a convex optimization problem and may be solved rapidly with standard methods.

IV. Design and Implementation

Figure 2 summarizes the architecture of our system. It consists of four modules: 1) When traffic goes through the gateway for forwarding, it is passed to a device and application classifier to identify the traffic type and priority. 2) All traffic is redirected through a proxy process that forwards traffic between client devices and the Internet. The data forwarding rate is determined by the optimizer (L2 Allocator) in each gateway by considering app priorities and is enforced by a rate limiter. 3) The bandwidth (credit spending) for each gateway is computed by the optimizer (L1 Allocator). 4) A user can access the gateway through a web interface to view its usage (at aggregate or joint device-app levels) and update its preferences, *i.e.*, when to spend more credits and traffic priorities, so as to adjust the optimizer's decisions. Screenshots of the user interface are shown in Figures 3(a) and 3(b).

We implement our system in a commodity wireless router, a Cisco E2100L with an Atheros 9130 MIPS-based 400MHz processor, 64MB memory, and 8MB flash storage (Figure 4). We replaced the factory default firmware with OpenWrt, a Linux distribution commonly used for embedded devices. The implementation poses two significant challenges:

Traffic and device classification: Standard approaches for classifying traffic from different devices include port-based protocol detection and OS fingerprinting. However, different apps can run on the same protocol, *e.g.*, videos streamed in



Aggregate

	# packets	# bytes
Incoming	109213	144353719
Outgoing	65515	6463358

Individual

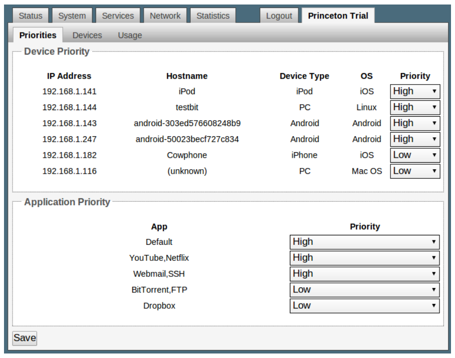
Device MAC	IP address	Incoming # packets	Incoming # bytes	Outgoing # packets	Outgoing # bytes
20:89:84:37:99:94	192.168.1.144	10601	14420712	6550	611586
38:aa:3c:db:30:b3	192.168.1.143	5376	6310967	4520	385021
f0:27:65:d9:63:11	192.168.1.247	71539	100062236	39348	2648241
c8:6f:1d:13:73:3e	192.168.1.182	4918	6220959	3608	292154
04:0c:ce:22:0f:40	192.168.1.116	14808	16054284	9278	2106365
7c:6d:62:50:12:1d	192.168.1.141	208	201302	204	34524

App Breakdown per Device

Device IP: 192.168.1.144

App	Incoming # packets	Incoming # bytes	Outgoing # packets	Outgoing # bytes
http	8409	12507429	4654	222325
ssh	129	63004	182	14945
icmp	27	2268	27	2268
bittorrent	0	0	0	0
youtube	48	38874	57	13421

(a) Usage tracking.



Device Priority

IP Address	Hostname	Device Type	OS	Priority
192.168.1.141	iPod	iPod	iOS	High
192.168.1.144	testbit	PC	Linux	High
192.168.1.143	android-303ed576608248b9	Android	Android	High
192.168.1.247	android-50023becf727c834	Android	Android	High
192.168.1.182	Cowphone	iPhone	iOS	Low
192.168.1.116	(unknown)	PC	Mac OS	Low

Application Priority

App	Priority
Default	High
YouTube,Netflix	High
Webmail,SSH	High
BitTorrent,FTP	Low
Dropbox	Low

(b) Traffic priorities and device/OS classification.

Fig. 3. Screenshots of the web interface.

Fig. 4. Cisco E2100L board.

HTTP, and many device types run on the same OS, *e.g.,* most smartphones run on a variant of Linux. Moreover, home gateways have only limited computational resources, but both these approaches require significant computational overhead.

Rate limiting and prioritization: We can limit a session's bandwidth rate by directly setting its TCP advertised window size [6], [7]. However, doing so requires knowing each connection's RTT and the number of active connections, both of which can be difficult to estimate in practice.

A. *Traffic and Device Classification*

To build a low-overhead classifier, we integrate a kernel-level `netfilter` module that inspects the first several packets of a connection for application matching. If a match is found, the classifier marks the connection with a mark to be queried at userspace by our proxy processes through `netlink`.

Our classifier module performs traffic classification *above* layer 7, *i.e.,* it can differentiate YouTube and Netflix, through a combination of content matching, byte tracking and protocol fingerprinting. We classify devices and OSes by using the same module to monitor HTTP traffic and inspect user-agent header strings for device information. This approach is practically effective due to the prevalence of devices using HTTP traffic.

B. *Rate Limiting Engine*

Our goal is to: 1) enforce an aggregate rate limit over multiple connections, and 2) enforce prioritization, *i.e.,* which gets higher bandwidth, among the connections given the aggregate limit. In this paper we only consider throttling *incoming* traffic, because the other direction can be easily and accurately done using standard token bucket-based traffic shaping tools.

Transparent Proxy. During the establishment of a connection between a client device and a server, it is intercepted at the gateway and redirected to the proxy process running in the gateway. Then the proxy establishes a new connection to the server on behalf of the client and forwards traffic between the two (proxy-server and client-proxy) connections. We use the Linux `splice()` function to achieve zero-copying, *i.e.,* all data are handled in kernel space.

Implicit Receive Window Control. TCP's flow control mechanism allows the receiver of a connection to advertise a receive window to the sender so that incoming traffic does not overwhelm the receiver's buffer. While originally set to match the available receiver buffer space, the receive window can be artificially set to limit bandwidth using the relation $\texttt{cwnd} = \texttt{rate} \times \texttt{RTT}$: given a maximum `rate` and measured round trip time (`RTT`), the receive window can be set to no greater than $\texttt{rate} \times \texttt{RTT}$. We opt for an *adaptive* approach such that the proxy does not need to know the `RTT` or compute the exact window size.

To illustrate our approach we consider a one connection case. As data from the server arrive at the proxy, they are queued at the proxy's receive buffer until the proxy issues a `recv()` on the proxy-server socket to process and clear them (at the same time the proxy issues a `send()` on the client-proxy socket to forward the data to the client). Note that if we modulate the frequency and the size of `recv()`'s, we modulate the size of the receive buffer and effectively the sending rate.

More specifically, we consider the model in Figure 5: the queue is the proxy's receive buffer, B is the receive buffer size,[8] and at time t, $F(t)$ is the fill rate (sending rate, which the proxy cannot directly control), $D(t)$ is the drain rate (how frequent the proxy issues `recv()`'s), $Q(t)$ is the queue length, and $W(t) = B - Q(t)$ is the advertised window size.

Suppose updates happen at intervals of Δt. The window update equation is then

$$W(t + \Delta t) = W(t) + \left[D(t) - F(t)\right]^{+} \Delta t \tag{11}$$

and taking a fluid approximation by setting $\Delta t \to 0$, we have

$$\dot{W}(t) = \left[D(t) - F(t)\right]^{+}. \tag{12}$$

[8]The receive buffer size can change with time due to Linux's buffer autotuning mechanism, but these changes do not affect our algorithm.

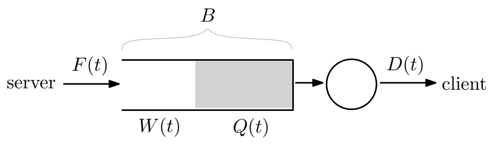


Fig. 5. Receive buffer model.

Our rate limiting goal is equivalent to getting $F(t) = R$ for large enough t through controlling $D(t)$. By setting $D(t) = R$ at all t, it is not difficult to verify from (12) that at equilibrium[9] we have $F^*(t) = R$ and $W^*(t) = R \times \texttt{RTT}$.

C. *Traffic Prioritization Engine*

When there are multiple connections the proxy spawns multiple threads such that each thread serves one connection, and we aim to limit the aggregate rate R over all connections. To allocate bandwidth fairly among the connections, we coordinate socket reads of these threads through a time division multiplexing scheme: using a thread mutex, we create a virtual time resource such that each socket read is associated with an exclusively held time slot of length proportional to the number of bytes read. Although more complicated socket read scheduling mechanisms can be considered, for simplicity we leave the scheduling to the operating system, and from experiments we observe that the time slots are shared fairly.

For traffic prioritization we assign a relative priority parameter $\alpha_i \in (0, 1]$ for every connection i such that for n *busy* connections, *i.e.,* each has a sufficiently large backlog, we want the sum of their rates R_i to be $\sum_{i=1}^{n} R_i = R$, and $R_i/R_j = \alpha_i/\alpha_j$ for $i, j = 1, \ldots, n$.

We achieve the desired prioritization through truncated reads. When the proxy issues a socket read, it needs to specify a maximum block size b to read (we set it as the page size of the processor architecture) and for a busy connection this limit b is always reached. If connection i is of lower priority with $\alpha_i < 1$, we truncate this block limit by setting it to be $\alpha_i b$. Since each access to a time slot is associated with a server socket read (equivalently, a client socket write) of $\alpha_i b$ bytes and time slots are fairly distributed across connections, the achieved client rate D_i (equivalent to R_i) scales with α_i.

By virtue of statistical multiplexing, our rate allocation mechanism does not require the number of busy connections n, which is difficult to track in practice; hence it can readily accommodate new connections. To accommodate bursty connections, the proxy first queries the receive buffer for the number of pending bytes. If it is above b then it does a truncated read as described above; otherwise it does not. The pseudocode of a proxy thread is shown in Algorithm 3.

Algorithm 3 Incoming rate control.

```
Input: $R$, $b$, $\alpha_i$,
  server_fd: socket of server connection,
  client_fd: socket of client connection,
  mutex: thread mutex shared by all connections
  while connection open do
    bytes_read = recv(server_fd, $b$)
    bytes_per_write = $\alpha_i \times$ bytes_read
    while not all bytes_read written to client do
      send(client_fd, bytes_per_write)
      lock(mutex)
          sleep(bytes_per_write/$R$)
      unlock(mutex)
    end while
  end while
```

V. Experimental Results

A. *Rate Limiting*

We compare our approach with the standard Linux traffic policing approach using the `tc` command with two different choices of the burst parameter. Two experiments are performed using `iperf`. In the first one, we fix network RTT to be 100ms and vary the rate limit from 1 to 15Mbps to observe the actual rate achieved. Figure 6(a) shows that our approach results in more accurate rate limiting (less than 4% error in each setting). While it appears that increasing the burst parameter helps in improving rate limiting accuracy, we note the values chosen are rather large (a typical value is 10k, while we use 50k and 200k) and may harm network stability. The sensitivity of the results of `tc` with respect to the parameters suggests the need for careful parameter tuning, which is undesirable given the diversity of network environments.

The first experiment hints that traffic policing, or using packet drops to signal the sender to reduce its rate, is too drastic as a rate control mechanism. Our second experiment confirms this observation. We fix the rate limit at 8Mbps and burst parameter at 50k, and vary network RTT from 20 to 100ms. Figures 6(b) and 6(c) show that `tc` results in significantly more packet retransmissions and higher jitter. This result shows that our approach is indeed more graceful in rate limiting.

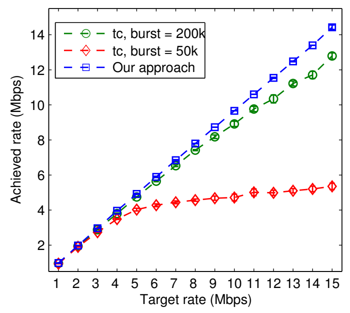
B. *Traffic Prioritization*

Consider a scenario with two users, one watching a 720p YouTube video stream and the other downloading a large file with `wget`, competing for a limited bandwidth of 2Mbps. We vary the priorities of the two types of traffic and observe the effect on video playback.

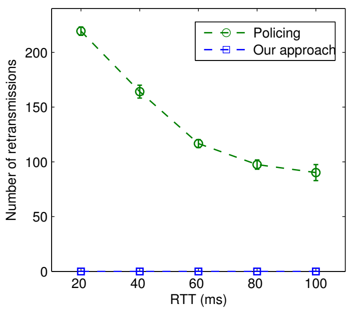
Let α_1 and α_2 be the priorities of YouTube and `wget` respectively. With α_1 fixed, we vary α_2 and measure the amount of video played over time.[10] Note there are two base cases: the case $\alpha_1/\alpha_2 = \infty$ corresponds to YouTube traffic without `wget` interference and is the best possible result we can expect; the case $\alpha_1/\alpha_2 = 1$ is equivalent to no prioritization. Figure 7 shows the results. When $\alpha_1/\alpha_2 > 1$, *i.e.,* YouTube has higher priority, playback performance (inversely related to the duration of pauses or flat regions in a curve) is strictly better than the no prioritization case. Also, performance improves with increasing α_1/α_2 ratio. Not only is our system able to do fine-grained traffic classification with two types of traffic

[9]Note that if we throttle a connection through TCP flow control, a static equilibrium can indeed be achieved because the rate is now limited by the receive window rather than self-induced congestion, *i.e.,* the usual sawtooth $W(t)$ time evolution no longer occurs.

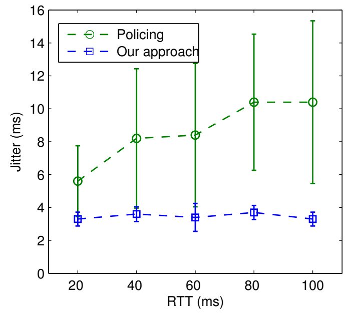
[10]We create a video-embedded webpage with a Javascript snippet that periodically queries the YouTube API for playback progress.



(a) Throughput.



(b) Retransmission counts.



(c) Jitter.

Fig. 6. Our rating limiting algorithm is more (a) accurate and (b, c) graceful than rate limiting through `tc`. We average all results over 10 runs, 60 seconds each, and show 95% confidence intervals.

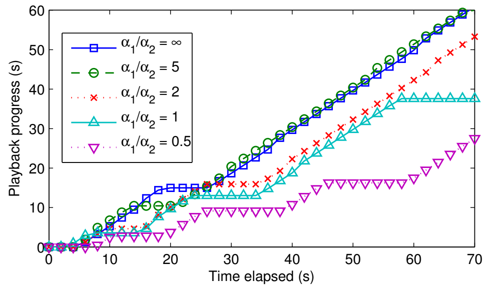


Fig. 7. YouTube playback performance improves as α_1/α_2 increases and YouTube receives higher prioritization over `wget`.

running under HTTP, but our traffic prioritization algorithm also produces noticeable improvement in user experience.

VI. Gateway Sharing Results

To demonstrate our sharing framework's efficacy, we consider sixteen gateways sharing a cable link. We compare our credit-based allocation to *equal sharing*, in which the ISP divides its capacity into slots with a minimally acceptable level of bandwidth, *e.g.,* 1 Mbps, and assigns them to gateways in a round-robin fashion until the network capacity is reached. This approach, which is similar to current practices in that gateways are all treated equally, risks inefficiency: gateways may gain little additional utility from the full bandwidth of their assigned slots, but cannot redistribute any excess bandwidth to gateways that would benefit more. Our credit-based approach addresses this disadvantage, and we show in our simulations that it significantly improves gateway utilities while enforcing a fair rate allocation.

After comparing the credit-based and equal sharing solutions, we evaluate our online algorithm for users' credit spending decisions (Algorithm 1 in Section III). We find that all gateways achieve near-optimal utilities despite their uncertain future budgets in the online case. The gateways actively save and spend credits at different times, resulting in a fair bandwidth allocation.

A. Gateway Utilities and Simulation Parameters

We suppose that credit-based sharing is enforced in the congested hours between 6pm and midnight, with half-hour timeslots. Users at each gateway are assumed to make their

TABLE I
Number of devices at each gateway.

Gateway	iPhones	Androids	Windows laptops	Mac laptops
1,4,9,13	1	1	1	1
2,6,10,14	2	0	2	1
3,7,11,15	1	1	1	2
4,8,12,16	2	0	1	1

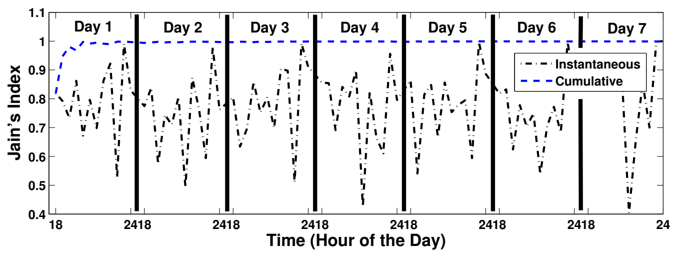
credit spending decisions based on their probability of using four types of applications: streaming, social networking, file downloads, and web browsing. We use the utility functions

$$u_1(x) = \frac{2(25x)^{1-\alpha_1}}{1-\alpha_1}$$
$$u_2(x) = \frac{(25x)^{1-\alpha_2}}{1-\alpha_2}$$
$$u_3(x) = \left(\frac{1}{\alpha_3 - 1} + \frac{(25x+1)^{1-\alpha_3}}{1-\alpha_3}\right)$$
$$u_4(x) = 15\left(\frac{1}{\alpha_4 - 1} + \frac{(25x+1)^{1-\alpha_4}}{1-\alpha_4}\right)$$

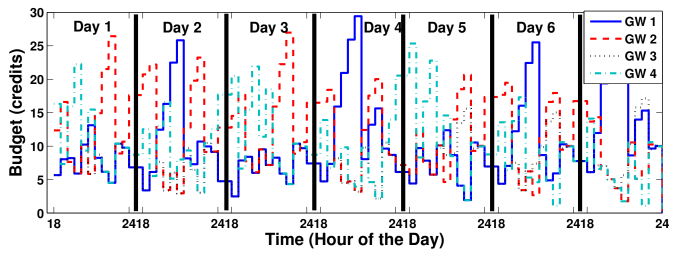
in (8) to respectively model the utility received from each application, where $(\alpha_1, \alpha_2, \alpha_3, \alpha_4) = (0.7, 0.5, 0.2, 3)$. The probabilities p_{it}^k of using each application are adapted from a recent measurement study of per-app usage over time for iOS, Android, Windows, and Mac smartphones and computers [8]. Table VI-A shows the devices at each gateway. We choose coefficients $\gamma_i(t)$ to be larger in the evening, as is consistent with observed data usage [8], and add random fluctuations to model period-to-period variations in each gateway's behavior.

We assume a budget of $B = 160$ total credits, with each credit representing 1Mbps.[11] The budget b_{it} for each gateway i is capped at 32 credits at any given time. In addition to the purchased bandwidth, we suppose that gateways send a random amount of traffic over the second tier, which is capped at the network capacity. We consider one week of credit redistributions and bandwidth allocations.

[11]Though 160 Mbps is a relatively small bottleneck bandwidth, we limit the number of users and link capacity in order to better illustrate the effect of QoE credit allocation on individual users.



(a) Jain's Index for gateway rates.



(b) Credit budgets for representative gateways.

Fig. 8. With our credit sharing scheme, all gateways (a) achieve comparable cumulative rates by (b) actively saving and spending credits at different times.

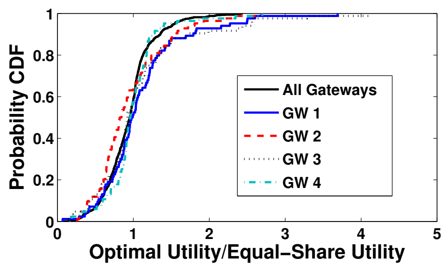


Fig. 9. With our credit sharing scheme, users achieve similar utility gains over equal sharing over one week.

B. Bandwidth Allocations

Globally Optimal Solution: We first compute the globally optimal rates, *i.e.,* those that maximize (7). To show that the overall rate allocation is fair, we compute Jain's Index over the gateways' rates, including second-tier traffic, at each time in Figure 8(a). Jain's Index is relatively low at some times, indicating a large variation in gateways' rates: some gateways use little bandwidth to save credits, while others spend a lot of credits to receive large rates. Yet if we compute the index for all gateways' *cumulative* usage over time, its value quickly converges to 1. The gateways receive comparable cumulative rates, consistent with the fairness property of Prop. 1.

The large variability in gateway allocations at a given time can be seen more clearly in Figure 8(b), which shows the budgets of four representative gateways over time. All four sometimes save credits to spend at other times. This flexibility causes total achieved utility to increase by 29.7% relative to equal sharing (allocating 10Mbps to each gateway at all times).

Figure 9 shows the cumulative density function (CDF) of the ratio of gateway utilities under credit allocation and equal sharing at different times. We plot the CDF over all gateways and times as well as the CDF over all times for each gateway shown in Figure 8(b). All of the CDFs are comparable, indicating that credit-based allocation benefits all gateways' utilities. While the gateways reduce their utility nearly half of the time, the utility more than doubles in some periods.

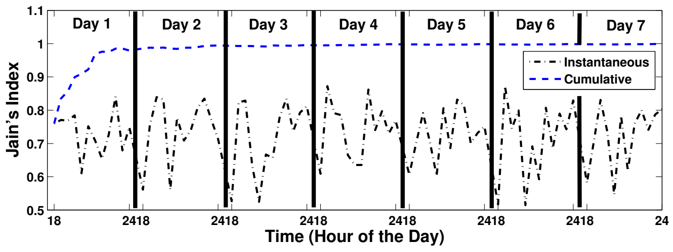
Online Solution: We next compare the globally optimal utilities with those obtained when the gateways follow Algorithm 1. To perform the credit estimation, we use four scenarios, in which all other gateways are assumed to use only streaming, only social networking, etc. Each gateway assumes (falsely) that the other gateways' $\gamma_i(t)$ coefficients are the same, and the probabilities π_σ of each scenario are initialized to be uniform. After learning the scenario distribution for only the simulation's first four days, the algorithm recovers most (84.7%) of the optimal utility for the remaining three days.

As with the optimal solution, at any given time gateways' rates can be very different: Jain's indices in Figure 10(a) for all gateways' usage at a given time can be quite low. However, all gateways achieve similar cumulative rates: Jain's index of the cumulative rates quickly converges to 1. Indeed, the budgets (and thus spending) of four representative gateways (Figure 10(b)) vary over time, as with the optimal solution (Figure 8(b)). Incentivizing gateways to delay some of their usage significantly improves users' overall satisfaction and utility.

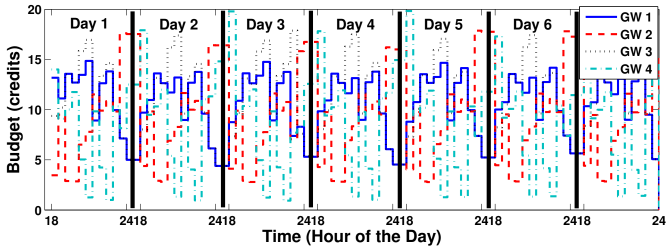
VII. Related Work

Using pricing to manage network congestion is a long-studied research area [2]. Our work differs in targeting broadband users on flat-fee service plans, prompting us to use a credit scheme instead of extra fees for prioritized access. Other credit-based schemes have been proposed for flow admission control [9] and for regulating access to higher-quality service [10], but these have remained theoretical proposals, due to users' reluctance to manually make complex token bidding decisions. We present a complete solution, from algorithms to implementation, for a specific problem of peak-hour broadband access. Moreover, our solution leverages user-specified QoE indicators to optimize traffic according to users' needs; while some works have introduced ways for users to give QoE feedback [11], [12], none of them have used this information to adjust bandwidth allocations.

From a systems perspective, there has been much recent work on developing smart home gateways with plain Linux/Windows or open-source router software such as OpenWrt. Smart home gateways have been used for network measurement [13], [14], providing intuitive interfaces for home network management [15], [16] and better QoS provisioning [17], [18]. One such gateway uses weighted fair queueing to allocate uplink traffic according to manual QoE feedback [19]. However, we are not aware of any work in coordinating bandwidth usage across households. We also develop our own incoming rate limiting tool, as off-the-shelf tools (*e.g.,* Linux `tc`) are insufficient for our application. The Congestion Manager (CM) project [20] shares similar goals of reducing congestion at the network edge, but we propose a QoE credit scheme to incentivize users to reduce usage, while CM



(a) Jain's Index for gateway rates.



(b) Credit budgets for representative gateways.

Fig. 10. Despite their uncertain future budgets, with our online algorithm gateways (a) achieve comparable cumulative rates by (b) saving and spending credits at different times over one week.

provides an API for applications to adapt to varying network conditions and requires sender-side support.

Receiver-side rate control is mostly done through explicitly controlling the receive window [21] or the receive socket buffer [22], *e.g.,* to implement low-priority transfers [23] and prioritize traffic [6], [7]. Our solution does not modify client devices or track the number of active connections and their RTTs. It also avoids interfering with Linux's buffer autotuning mechanism [24]. Our approach of implicit window control is similar to that of Trickle [25], but they serve different goals. Trickle is designed for non-root users to *voluntarily* rate limit their applications, while we aim to impose mandatory rate limits that are transparent to users.

VIII. Concluding Remarks

In this paper, we propose to solve peak-hour broadband network congestion problems by pushing congestion management to the network edge. We design incentive mechanisms to *empower users* to moderate their demand in a decentralized, personalized solution that respects user privacy and requires minimal support from ISP infrastructure and user devices.

Our solution performs a two-level bandwidth allocation: in Level 1, gateways purchase bandwidth on a shared link using QoE credits, and in Level 2 they divide the purchased bandwidth among their apps and devices. We show analytically that our credit distribution scheme yields a fair bandwidth allocation across gateways and describe our implementation of the bandwidth purchasing and app prioritization on commodity wireless routers. Our implementation can successfully enforce app priorities and increase users' satisfaction. Finally, we show in an example scenario that our algorithm's ability to adapt to users' QoE yields a fair bandwidth allocation that significantly improves user utility relative to a baseline equal-sharing scheme.

Acknowledgements

This work was supported by NSF CNS 1347234.

References

[1] Cisco Systems, "Cisco visual networking index: Forecast and methodology, 20132018," June 2014, http://tinyurl.com/VNI2014.

[2] S. Sen, C. Joe-Wong, S. Ha, and M. Chiang, "A survey of smart data pricing: Past proposals, current plans, and future trends," *ACM Computing Surveys*, vol. 46, no. 2, 2013.

[3] Comcast, "Frequently asked questions about network management," 2014, http://customer.comcast.com/Pages/FAQViewer.aspx?seoid=frequently-asked-questions-about-network-management#technique.

[4] F. Kelly, A. K. Maulloo, and D. H. K. Tan, "Rate control for communication networks: Shadow prices, proportional fairness, and stability," *Journal of the Operational Research Society*, vol. 49, pp. 237–252, 1998.

[5] A. Consiglio, F. Cocco, and S. A. Zenios, "Scenario optimization asset and liability modelling for individual investors," *Annals of Operations Research*, vol. 152, no. 1, pp. 167–191, 2007.

[6] N. T. Spring, M. Chesire, M. Berryman, V. Sahasranaman, T. Anderson, and B. Bershad, "Receiver based management of low bandwidth access links," in *Proc. IEEE INFOCOM*, 2000.

[7] Y. Im, C. Joe-Wong, S. Ha, S. Sen, T. T. Kwon, and M. Chiang, "AMUSE: Empowering users for cost-aware offloading with throughput-delay tradeoffs," in *Proc. IEEE INFOCOM*, 2013.

[8] J. Y. Chung, Y. Choi, B. Park, and J.-K. Hong, "Measurement analysis of mobile traffic in enterprise networks," in *Proc. APNOMS*, 2011.

[9] J. K. MacKie-Mason, L. Murphy, and J. Murphy, "Responsive pricing in the Internet," *Internet Economics*, pp. 279–303, 1995.

[10] D. Lee, J. Mo, J. Walrand, and J. Park, "A token pricing scheme for internet services," in *Economics of Converged, Internet-Based Networks*. Springer, 2011, pp. 26–37.

[11] J. S. Miller, A. Mondal, R. Potharaju, P. A. Dinda, and A. Kuzmanovic, "Understanding end-user perception of network problems," in *Proceedings of the first ACM SIGCOMM workshop on Measurements up the stack*. ACM, 2011, pp. 43–48.

[12] C.-C. Tu, K.-T. Chen, Y.-C. Chang, and C.-L. Lei, "Oneclick: A framework for capturing users network experiences," *Proceedings of ACM SIGCOMM 2008 (poster)*, 2008.

[13] S. Sundaresan, W. de Donato, N. Feamster, R. Teixeira, S. Crawford, and A. Pescapè, "Broadband Internet performance: A view from the gateway," in *Proc. ACM SIGCOMM*, 2011.

[14] A. Patro, S. Govindan, and S. Banerjee, "Observing home wireless experience through WiFi APs," in *Proc. ACM MobiCom*, 2013.

[15] R. Mortier, T. Rodden, P. Tolmie, T. Lodge, R. Spencer, A. crabtree, J. Sventek, and A. Koliousis, "Homework: Putting interaction into the infrastructure," in *Proc. ACM UIST*, 2012.

[16] J. Yang, W. Edwards, and D. Haslem, "Eden: Supporting home network management through interactive visual tools," in *Proc. ACM UIST*, 2010.

[17] C. E. Palazzi, M. Brunati, and M. Roccetti, "An OpenWRT solution for future wireless homes," in *Proc. IEEE ICME*, 2010.

[18] C. Gkantsidis, T. Karagiannis, P. Key, B. Radunovi, E. Raftopoulos, and D. Manjunath, "Traffic management and resource allocation in small wired/wireless networks," in *Proc. ACM CoNEXT*, 2009.

[19] J. S. Miller, J. R. Lange, and P. A. Dinda, "Emnet: Satisfying the individual user through empathic home networks," in *INFOCOM, 2010 Proceedings IEEE*. IEEE, 2010, pp. 1–9.

[20] H. Balakrishnan, H. S. Rahul, and S. Seshan, "An integrated congestion management architecture for Internet hosts," in *Proc. ACM SIGCOMM*, 1999.

[21] L. Kalampoukas, A. Varma, and K. K. Ramakrishnan, "Explicit window adaptation: A method to enhance TCP performance," in *Proc. IEEE INFOCOM*, 1998.

[22] J. Semke, J. Mahdavi, and M. Mathis, "Automatic TCP buffer tuning," in *Proc. ACM SIGCOMM*, 1998.

[23] P. Key, L. Massoulіè, and B. Wang, "Emulating low-priority transport at the application layer: A background transfer service," in *Proc. ACM SIGMETRICS/Performance*, 2004.

[24] M. Fisk and W.-c. Feng, "Dynamic right-sizing in TCP," in *Proc. LACSI Symposium*, 2001.
[25] M. A. Eriksen, "Trickle: A userland bandwidth shaper for Unix-like systems," in *Proc. USENIX Annual Technical Conference*, 2005.

APPENDIX A
PROOF OF LEMMA 1

We proceed by induction: at time $t = 0$, clearly the sum of gateways' budgets $\sum_i b_{i0} = B$ from the budget initialization. Supposing that $\sum_{i=1}^n b_{it} = B$ at time t, we then calculate

$$\sum_{i=1}^n b_{i,t+1} = \sum_{i=1}^n \left(b_{it} - x_{it} + \sum_{j\neq i} \frac{x_{jt}}{n-1} \right)$$
$$= B - \sum_{i=1}^n x_{it} + \sum_{i=1}^n \frac{(n-1)x_{it}}{n-1} = B.$$

APPENDIX B
PROOF OF LEMMA 2

We first note that (1) is equivalent to the statement that

$$b_{it} = b_{is} + \sum_{\tau=s}^{t-1} \left(-x_{i\tau} + \sum_{j\neq i} x_{j\tau}/(n-1) \right).$$

Then if $b_{is} = b_{it}$ for all gateways i, we obtain the system of equations

$$\sum_{\tau=s}^{t-1} x_{i\tau} = \sum_{\tau=s}^{t-1} \sum_{j\neq i} \frac{x_{j\tau}}{n-1}. \tag{13}$$

It suffices to show that (13) implies the proposition.

We proceed by induction on n. If $n = 2$, then clearly (13) is exactly our desired result, since $n-1 = 1$. We now suppose that the proposition holds for $n = m$ and show that it holds for $n = m + 1$. From (13), we have

$$\sum_{\tau=s}^{t-1} x_{1\tau} = \sum_{\tau=s}^{t-1} \sum_{j=2}^{n} \frac{x_{j\tau}}{n-1}.$$

Substituting this equality into (13) for $i > 1$, we have for all such i,

$$\sum_{\tau=s}^{t-1} x_{i\tau} = \sum_{\tau=s}^{t-1} \sum_{j=2}^{n} \frac{x_{j\tau}}{(n-1)^2} + \sum_{\tau=s}^{t-1} \sum_{j\neq i, j>1} \frac{x_{j\tau}}{n-1}.$$

Thus, we have upon rearranging that

$$\left(1 - \frac{1}{(n-1)^2}\right) \sum_{\tau=s}^{t-1} x_{i\tau} = \left(\frac{1}{(n-1)^2} + \frac{1}{n-1} \right) \sum_{\tau=s}^{t-1} \sum_{j\neq i, j>1} x_{j\tau}.$$

Simplifying, we obtain

$$\sum_{\tau=s}^{t-1} x_{i\tau} = \sum_{\tau=s}^{t-1} \sum_{j\neq i, j>1} \frac{x_{j\tau}}{n-2}$$

for all $i > 1$. By induction, this implies that $\sum_{\tau=s}^{t-1} x_{j\tau} = \sum_{\tau=s}^{t-1} x_{k\tau}$ for all $j, k > 1$, and the proposition follows upon solving for $\sum_{\tau=s}^{t-1} x_{1\tau}$.

APPENDIX C
PROOF OF PROPOSITION 1

We first show that given a distribution of budgets $\{b_{it}\}$ at a fixed time t, there exists a set of gateway spending decisions $\{x_{it}\}$ such that $b_{i,t+1} = B/n$ for all gateways i. Suppose that each gateway i spends $x_{it} = b_{it}(n-1)/n$ credits at time t. Then Lemma 1's budget conservation allows us to conclude that gateway i's budget at time $t+1$ is

$$b_{i,t+1} = b_{it} - \frac{b_{it}(n-1)}{n} + \sum_{j\neq i} \frac{b_{jt}(n-1)}{n(n-1)} = \sum_{i=1}^n \frac{b_{it}}{n} = \frac{B}{n}.$$

We now observe that since each $b_{i0} = B/n$, we can apply Lemma 2 to conclude that

$$\sum_{s=0}^{t+1} x_{is} = \sum_{s=0}^{t} x_{is} + \frac{b_{it}(n-1)}{n} = \sum_{s=0}^{t} x_{js} + \frac{b_{jt}(n-1)}{n} = \sum_{s=0}^{t+1} x_{js}$$

for all gateways i and j. We then rearrange this equation to find the first part of the proposition:

$$\left| \sum_{s=0}^{t} x_{is} - \sum_{s=0}^{t} x_{js} \right| = |b_{jt} - b_{it}| \frac{n-1}{n} \leq \frac{B(n-1)}{n}.$$

The time average follows immediately upon dividing by t and taking limits as s$t \to \infty$.

APPENDIX D
PROOF OF PROPOSITION 2

To prove the first part of the proposition, we note that if each $x_{it} = 0$, then (1) yields

$$b_{i,t+1} = b_{it} - x_{it} + \sum_{j\neq i} \frac{x_{jt}}{n-1} \leq \frac{B}{n-1} + b_{it}\frac{n-2}{n-1} - x_{it},$$

where the inequality comes from each gateway's budget constraint $\sum_{j\neq i} x_{jt} \leq \sum_{j\neq i} b_{jt} = B - b_{it}$. Thus, at time $t+1$, we have

$$b_{i,t+1} = \sum_{\tau=0}^{t} \left(\frac{B}{n-1} - x_{i\tau} \right) \left(\frac{n-2}{n-1} \right)^{\tau} + \frac{B}{n} \left(\frac{n-2}{n-1} \right)^{t+1}$$
$$\leq \frac{B}{n}\alpha^{t+1} + B\left(1 - \alpha^{t+1}\right) - \epsilon \sum_{\tau=1}^{\left\lfloor \frac{t+1}{p} \right\rfloor} \alpha^{p\tau}$$
$$= \frac{B}{n}\alpha^{t+1} + B\left(1 - \alpha^{t+1}\right) - \epsilon \left(\frac{\alpha^p - \alpha^{p\left(1 + \left\lfloor \frac{t+1}{p} \right\rfloor\right)}}{1 - \alpha^p} \right)$$

as desired, using the fact that $\sum_{\tau=s}^{s+n} x_{i\tau} \geq \epsilon$ at any time s. We obtain (3) by taking $t \to \infty$, substituting for $\alpha = \frac{n-2}{n-1}$, and simplifying.

To prove the second part of the proposition, suppose that gateways i and k both have zero budgets at time $t+1$, i.e., $b_{i,t+1} = b_{k,t+1} = 0$, but that $b_{it} > 0$. Since each $b_{i0} = B/n > 0$, such a time t must exist. But then from (1), $b_{i,t+1} = b_{it} - x_{it} + \sum_{j\neq i} x_{jt}/(n-1) = 0$, and since each $x_{jt} \geq 0$, we have $x_{it} = b_{it} > 0$. But then $b_{k,t+1} = b_{kt} - x_{kt} + \sum_{j\neq k} x_{jt}/(n-1)$, and since $x_{kt} \leq b_{kt}$, we have $b_{k,t+1} > 0$, which is a contradiction. Thus, at most one gateway can have zero budget in any given time period.

APPENDIX E
PROOF OF PROPOSITION 3

We first note that at each time $t < s$,

$$\begin{aligned}\sum_{i=1}^{m} b_{i,t+1} &\leq \sum_{i=1}^{m} b_{it} + \sum_{j>i} \frac{x_{it}}{n-1} \\ &\leq \sum_{i=1}^{m} b_{it} + \frac{B - \sum_{i=1}^{m} b_{it}}{n-1} \\ &= \frac{B}{n-1} + \left(\frac{n-2}{n-1}\right) \sum_{i=1}^{m} b_{it}.\end{aligned}$$

An inductive argument then shows that

$$\sum_{i=1}^{m} b_{i,s} \leq \frac{B}{n-1}\left(\sum_{\tau=0}^{s-1}\left(\frac{n-2}{n-1}\right)^{s}\right) + \left(\frac{n-2}{n-1}\right)^{s} \sum_{i=1}^{m} b_{i0}.$$

Expanding the sums and subtracting $\sum_{i=1}^{m} b_{i0}$ then yields the proposition.

APPENDIX F
PROOF OF PROPOSITION 4

Suppose that $\{x_{it}^*\}$ solve (7), and let λ_{it} denote the corresponding Lagrange multiplier for the constraint $0 \leq x_{it} \leq b_{it}$, with ν_{it} the multiplier for the constraint $x_{it} \geq 0$. Since the U_{it} are strictly concave, it suffices to show that these multipliers satisfy the Karush-Kuhn-Tucker conditions for (5), augmented by all gateways' constraints:

$$\max_{x_{it}} \sum_{t=s}^{s+T} U_{it}\left(x_{it}\right), \;\; \text{s.t. } 0 \leq x_{it} \leq b_{it}, \; \forall i, t.$$

Since the budget constraints $0 \leq x_{it} \leq b_{it}$ are identical to those of (7), it suffices to show that

$$\frac{dU_{it}}{dx_{it}} - \sum_{j=1}^{n} \sum_{\tau=t}^{s+T} \lambda_{i\tau} + \sum_{j\neq i} \sum_{\tau=t}^{s+T-1} \frac{\lambda_{j\tau}}{n-1} + \nu_{it} = 0, \qquad (14)$$

where we use (6) to sum over the appropriate multipliers $\lambda_{i\tau}$. However, this equation is just one of the KKT conditions for (7): the only change between (7) and (5) is the addition of utility terms $U_{jt}(x_{jt})$, which are additively decoupled from gateway i's spending decisions x_{it}. Thus, (14) must be satisfied by the x_{it}^* and multipliers λ_{it}, ν_{it}. Each gateway i is thus optimizing its own utility, given other gateways' credit spending decisions x_{jt}^*.