Computer Sciences Department

RouteBazaar: An Economic Framework for Flexible Routing

Holly Esquivel
Chitra Muthukrishnan
Feng Niu
Shuchi Chawla
Aditya Akella

Technical Report #1654

April 2009

UNIVERSITY OF WISCONSIN MADISON

RouteBazaar: An Economic Framework for Flexible Routing

Holly Esquivel, Chitra Muthukrishnan, Feng Niu, Shuchi Chawla, Aditya Akella
University of Wisconsin - Madison

ABSTRACT

The Internet's routing protocol provides users a single end-to-end route that is not guaranteed to be available or to meet user requirements. Our paper addresses this rigidity using an economically grounded approach that appeals both to users and to service providers. We propose a framework called RouteBazaar in which service providers announce links connecting different parts of the Internet, along with dynamically changing prices associated with using the links. All topology information is exposed to end-users. Users simply use routes that best match their cost-performance requirements. The appeal to users is that they can always obtain a route of their choice as long as they are willing to pay for it. The appeal to providers is that our framework offers them means for monetary gains as long as they are willing to offer greater visibility into their topology and greater flexibility in user routes. We present both centralized and distributed versions of this framework. Both versions are designed to supplement, not supplant, BGP. We design a variety of algorithms to ensure robustness and stability of either framework. We study the framework using a simplified game-theoretic model and also conduct large scale simulations of its performance in practice using real and synthetic network topologies. We find that despite allowing flexibility to both users and service providers, our system operates at a stable and close-to-optimal state.

1. INTRODUCTION

The BGP routing protocol forms the backbone of the Internet today. It enables end-users to communicate with different corners of the Internet and has played a key role in the growth of the network. Unfortunately, BGP suffers from key drawbacks that constrain the Internet and the applications running on it in various ways [26, 3, 22, 10, 29]. A key constraint of BGP is that it offers end-users very limited visibility into network topology and routes, providing users exactly one policy-constrained path per destination. There is no guarantee that the single route that BGP provides will meet an end-user's requirements. Most importantly, the route is not guaranteed to be available [3].

Approaches that have been proposed to address these challenges and improve end-user experience, such as overlay routing [27, 3] and multihoming [2, 4], are either undesirable in practice or inadequate [2]. The flexibility offered by overlay routing to end-users has undesirable interactions with ISP policies and traffic engineering objectives. Multihoming can offer better performance than single BGP paths, but it still cannot guarantee meeting users' end-to-end requirements satisfactorily (e.g., routes are not guaranteed to be available).

We argue that overcoming the rigidity in BGP—i.e., the poor visibility into routes and no flexibility in routing—requires an alternate, economically-grounded approach. We believe that service providers will be willing to expose greater amounts of information to users, and/or offer them greater flexibility in routing, as long as this leads to suitable monetary compensation. Equivalently, users who have the means will spend money to buy the additional visibility and flexibility as long as their requirements are guaranteed to be met. Our work asks whether it is possible to develop a flexible and robust economic framework to match the desires of users with those of service providers.

A crucial requirement of the framework is that it attract participation from both users and service providers. Therefore, it should explicitly account for the local goals of both users and service providers, making *robustness to selfish behavior* a first class design requirement. This offers flexibility simultaneously to users and service providers, and fosters greater participation from both, without requiring any kind of global oversight. A secondary requirement is that the framework exist alongside BGP and other existing systems such as multihoming and overlay routing. This ensures that whenever user and service provider requirements are misaligned, users can fall back to the existing mechanisms.

We describe and study such a framework called RouteBazaar. Providers participating in RouteBazaar announce (virtual) links connecting different locations of the Internet, along with the price associated with routing over them. Crucially, providers have the flexibility of dynamically altering the link prices so as to control quality of their links and, more importantly, to attract traffic and maximize their revenue. Users of RouteBazaar will have complete visibility into the links and paths offered by service providers via the system. Crucially, users will always be able to find a route over RouteBazaar to the destination of their choice that best meets their constraints as long as they have the willingness to pay for it. A user's route in RouteBazaar could be composed of links from different providers (each with a different price). Each user settles payments with each individual provider along the paths she uses.

RouteBazaar encourages competition among service providers and makes it easy for newcomers to participate in our system. In particular, providers can use dynamic pricing as a tool to attract traffic from users who are not connected directly to

their links. Thus, third-party service providers can start offering services using our system even if they don't have a customer base like ISPs do today.

We describe two variants of RouteBazaar. In the *centralized* variant, users submit their requirements and service providers submit their topology information to a central arbiter. The arbiter simulates selfish behavior for both users and service providers and derives the *Nash equilibrium* prices for providers links and routes for users (some users may be denied a route). In the *distributed* variant, users and service providers constantly and directly interact with each other. We develop learning-based algorithms for service provider- and user-response. The former dictates how service providers should change prices to maximize revenue. The latter dictates how users should reroute traffic (in response to price changes) toward newer paths that best match their requirements. Our algorithms are designed to ensure stable operation in the face of modest churn in the system.

We evaluate our framework using both analysis and large scale simulations. We consider a stylized model of RouteBazaar in a game-theoretic setting and analyze its *price of anarchy*—the gap between the performance of the system when users and service providers are selfish, and the optimal performance achievable through global oversight. Previous work [14] has shown that the price of anarchy worsens as the disparity among users or the number of monopolistic service providers in the system increases. We prove that the price of anarchy improves considerably under mild assumptions on the users' values. We conduct simulations over several real and synthetic topologies. Our key finding is that, in most scenarios, the net performance derived by RouteBazaar users (in both the centralized and distributed versions) is just 25%-35% away from the best possible (i.e. where all service providers are altruistic and provide globally-optimal routes). We find that network topology, link capacities and the requirements of users have minor impact on the performance of RouteBazaar in practice. On the whole, our study shows that RouteBazaar is a viable system to supplement BGP and could address some of its key drawbacks in a way that appeals to both users and service providers.

2. RELATED WORK

Our RouteBazaar framework builds upon existing literature on the interplay between routing and economics. We discuss potential similarities and key differences next.

Economics-based systems. The RouteBazaar framework bears some similarity to the network "nanopayment" systems Bill-Pay [15] and A la carte [16]. In these clean-slate architectures proposed as alternatives to BGP, ISPs announce a list of service levels and corresponding prices, which apply at the packet level. Based on knowledge of topology and link prices, users compute paths they wish to use and insert source routes along with nanopayments into each packet they send. The ISPs may be able to alter their prices, although neither system specifies what approach ISPs should adopt. The nanopayments are used to pay each on-path ISP for the service it offered to the packet. Because these are packet-level system they impose significant overhead on the service provider infrastructure. They also need significant changes to applications and to TCP/IP stacks. Users in these systems may not find a path whose price matches their willingness to pay; neither system addresses how to accommodate them. Finally, unlike our framework, neither system has been evaluated (theoretically or otherwise) to test for practical viability.

A few other systems have proposed broker-based architectures for routing and quality of service. Bandwidth brokers [35] were proposed originally in the context provisioning QoS paths in a single service provider network. More recent proposals extend the bandwidth broker framework across multiple ISPs [20, 31]. Thus, they enable users to buy quality-of-service across multi-ISP paths. Both systems are designed to work alongside BGP, similar to our approach. The MINT proposal [31] uses a centralized mediator to run a continuous double auction in order to match user requests (e.g. for bandwidth between two points on the Internet) against ISP bids (e.g. price per unit bandwidth per link). A similar mediator is used in [20]. The design of these approaches is similar our centralized framework at a high level and they share the same broad goal as us, but there are key differences. In particular, neither approach takes the selfish goals of users and service providers into account. Service providers cannot directly compete against each other in these systems. Its also not clear if users of the system will have their requirements met in the optimal fashion (e.g. it is not clear if the systems provide users the least expensive paths for the level of service they want). Because our framework explicitly takes selfishness into account it addresses both of these issues. Thus, we believe that it fosters greater participation from both users and service providers. Another key difference is that, unlike our approach, it is not clear if these architectures are amenable to a distributed implementation.

Our framework also draws from prior work on congestion-based pricing for the Internet such as the smart-markets proposal [21]. In these proposals, network users pay for a service whose price is set dynamically based on the individual offers of the users. While these proposals apply to a single service provider network, ours applies to multiple service providers that could be interconnected arbitrarily.

Fine-grained route selection. Without directly addressing the economic aspects of the problem, several papers have proposed routing systems that allow users greater flexibility in selecting routes. In Pathlet routing [17], users perform source routing over "pathlets", where each pathlet is a virtual link between points in the Internet (e.g. between PoPs in the same ISP). NIRA [34] and MIRO [33] allow users to construct AS level source routes to achieve various traffic engineering objectives and economic goals. Both distributed and centralized versions of our system leverage these proposals to facilitate forwarding.

Analytical studies. A number of recent papers have theo-

retically analyzed the economic aspects of bandwidth pricing under various models (see, e.g., [19, 1, 5, 24, 14]). Among these the model considered by Chawla and Roughgarden [14] is the most relevant to our setting. Chawla and Roughgarden analyze the price of anarchy of a bandwidth pricing game in which service providers pick prices to maximize their revenue and users pick paths to maximize their utilities. The price of anarchy is the worst gap between the performance of an equilibrium (stable state) in this game to the optimal performance that may be achieved through centralized control. They show that the price of anarchy for this bandwidth pricing game can be large when there is a large disparity between the values of different users and the network contains many "monopolistic" service providers. In Section 5 we expand on these results and show that under mild and natural assumptions on the distribution of users' values in the network, the price of anarchy improves considerably. Our simulations in Section 6 support these findings.

3. OVERVIEW OF RouteBazaar

We start by briefly stating the architectural features of our framework and comparing against BGP routing.

(1) Separating route visibility from economics. Our system is designed to separate payment and economics from visibility into network topology. Specifically, economic constraints of service providers in our system should not limit users' knowledge of, and their ability to potentially use, different paths in the network. In BGP routing, providers implement a variety of economics-driven import and export policies on route announcements that filter out topology and route information before it reaches end-users.

(2) Optimal routes for users who are willing to pay. Users of the system should be able to route to their corresponding destinations using our system as long as they have the means to pay for the route provided by our system. Thus, only a user's willingness to pay determines whether or not she will find a route with our system. Today, end-networks who are willing to pay can afford to buy connectivity from one or more large ISPs with global reach. While such ISPs have greater reachability to various parts of the Internet, merely connecting to them does not guarantee that an end-to-end route will be available no matter how much a user pays.

In order to accommodate user constraints well and to encourage participation, we additionally require that the routes provided by our system be *optimal* from the user's perspective. In other words, there should be no cheaper route in the system that also meets the users constraints.

(3) Service provider flexibility. In order to encourage participation from service providers, our system should similarly offer them flexibility as well. We require that service provider constraints be accommodated by letting them set prices for the services they offer any way they want in order to attract traffic and boost revenue. Because of the hop-by-hop nature of Internet routing, today, a service provider's constraints are specified implicitly in the form of a collection of long-term bi-lateral relationships with neighboring networks. The binding, bi-lateral nature of these contracts constrains service providers from offering a rich variety of flexibly-priced services to users who transit their networks.

Our system allows service providers to vary prices for their services at finer time-scales. More importantly, the price imposed by a service provider is charged to all traffic routed over its links, irrespective of where it originates. Users who route atop our system pay each on-path service provider for the services they used. Thus our framework allows a service provider to attract a user who may not be directly connected to it to route over its links and potentially boost its revenues further. In contrast, ISPs today can only charge customers who are directly connected to them.

We present a high-level description of our RouteBazaar framework next. Design details and specific algorithms that we use to enable the above features are discussed in §4.

Assumptions. Our description of the RouteBazaar framework in this section makes a few assumptions.

The service offered by each service provider in our system is providing a route between two points in the Internet at some cost per unit bandwidth consumed on the path between the points. We use the term connectivity service providers or CSPs to describe the service providers.

Each user of the system wishes to use the system to route to a specific destination. Someone wishing to route to multiple destinations could be represented by a collection of end-users each wishing to route to one of the destinations. Users may enter and leave the system at any time. But we assume that this happens at a *very low rate*.

At any given time, some of the potential users of the system may be denied a route. This is because the system may be unable to provide routes with prices that match the user's willingness to pay and other requirements of the user. We assume that all such users fall back to using default BGP routes.

We assume that a source routing-based system such as Pathlet routing [17], MIRO [33] or NIRA [34] is available for users to route traffic along the paths specified by the system. The specific approach does not matter as long as it offers scalable support for source routing.

3.1 The RouteBazaar Framework

We now describe the two sets of participants in our framework, the service providers and users, and their behavior.

Connectivity service providers: Our system is composed of multiple connectivity service providers (CSPs) that offer to connect different Internet *locations*. Each location is akin to a point-of-presence today and each link is a virtual connection of a fixed maximum capacity between the locations. Each CSP can own multiple virtual links of different capacities. Each CSP announces a price per unit bandwidth for the links it owns. The link could have different prices. We assume that the CSP link graph connects users to their destinations.

Users: We envision the system to be used by medium to large end-networks, including enterprises (single or multi-

campus), universities, data centers, and even regional Internet service providers that wish to provide their customers robust transit across the wide-area network. Each such *user* E wishes to purchase some amount of bandwidth between its own location and the location of some destination D.

An user attaches some *value per unit demand* for its traffic. This reflects the user's willingness to pay per unit traffic to the destination D. We model this using a *demand-value* vector, $\langle B, V \rangle_{E,D}$. An entry $\langle b_i, v_i \rangle$ in this vector indicates that the user E is willing to pay at least v_i per unit traffic for b_i units of traffic to the destination D.

We assume that the users' values are local to the users, and provided to our system as inputs. Depending on a user's local requirements, the demand-value vectors to a destination may change slowly over time based, for example, on the time of the day or day of the week.

CSP and user behavior: Each CSP participating in RouteBazaar is *revenue-maximizing* in that it sets prices so that the net revenue derived from the collection of links owned is maximal. In particular, CSPs may change their link prices over time, especially when the current prices are deemed suboptimal given the price choices of other CSPs in the system. We assume that the CSPs are unaware of the values of the users and cannot distinguish between the users. The only feedback available to them from the system is the amount of traffic willing to use their links at their current prices.

A CSP's revenue per link is the amount of user demand routed on the link times the price per unit demand. However, whenever the load on link exceeds it capacity, we assume that the CSP derives no revenue from the link.

We assume that users select routes in a *utility-maximizing* fashion: given the prices of each CSP link, the routes employed by a user are such that the user's *utility*—the difference between the value derived by the user and the price it pays—is maximized on a per destination basis.

How a user uses the bandwidth provided by our system is purely based on local policies. A user may apply a local traffic classification algorithm and use its outcome to determine which traffic to route using the bandwidth allocated to it by the system. The rest of the user's traffic is routed in a best-effort manner using current BGP-based paths.

We now present two flavors of RouteBazaar, one based on a centralized arbiter and the other completely distributed.

3.2 Centralized RouteBazaar Architecture

In the centralized variant of RouteBazaar, we assume the presence of a logically central, neutral *arbiter service* with whom both users and CSPs communicate. The arbiter service coordinates the actions and choices of users and CSPs and tries to accommodate their requirements and goals.

The CSPs provide as input to the arbiter the list of links they own, the locations that the links connect, and the capacity of the links. Users provide as input their demands and the value per unit demand. Recall that we assume users to arrive or leave slowly over time. In our description of the centralized architecture below, and of the distributed architecture in the next subsection, we assume for simplicity that the set of users (and CSPs) remains fixed and does not change. We discuss how to accommodate low-rate churn in Sections 4 and 6.

The arbiter's role is to *simulate the behavior* of both the users and the CSPs (as per the behavior outlined in §3.1 above) and find a stable state for the system that is acceptable to all parties. It runs the simulation until an *equilibrium* is reached.

The arbiter determines the following:

1. The prices that CSPs should charge for their links.
2. Which users get to route their demand over the CSP network and those who must route using default BGP routes.
3. For the former set of users in #2 above:
   a. The routes to the corresponding destinations over the CSP network that the users must use.
   b. The split-up of users' demands across these routes.

The arbiter communicates this information to the CSPs and users, who can then choose to follow or ignore the arbiter's advice in picking their respective strategies.

In order for the arbiter's solution to be stable and acceptable to CSPs and users, it must form a *Nash equilibrium* [30]. At equilibrium the following properties are satisfied:

- No CSP has the incentive to unilaterally deviate from the current offered price in order to boost its revenue. Furthermore, no CSP observes a load > 1 on its links.
- All users employ utility-maximizing paths and have no incentive to route over alternate paths. If there exists a path whose price is less than a user's value per unit demand, then all of the user's demand is routed using RouteBazaar. If the price of the least-cost path equals the user's value for the traffic, the user's traffic may be split between RouteBazaar and regular BGP routes.

Flexible global policies: A feature of the centralized approach is that it allows the arbiter to impose flexible policies in order to try to ensure some *global properties* are satisfied, while ensuring that the local, selfish goals of the various player (CSPs and users) are also honored. For instance, one set of policies could ensure that, at equilibrium, a certain minimum amount of demand from each geographic region is guaranteed routes over the CSP network, irrespective of the values of the users located at each geographic location. This introduces some level of fairness into the system. Another set of policies could ensure that the net value derived across all users who end up using the system is maximized. While this can ensure global optimality in terms of net value, this policy could result in unfairness. A final set of policies could enforce some combination of fairness and value-optimality.

3.3 Distributed RouteBazaar Architecture

At a high level, the distributed architecture works in roughly the same fashion as the centralized approach. As before, CSPs can adjust their prices to boost their revenue. Users select routes over the collection of CSPs to maximize their utility. However, there are three key aspects that differentiate the distributed and the centralized architectures.

(1) **Dissemination.** In the centralized approach, the arbiter coordinates CSP pricing and user route selection and suggests a globally stable solution to all parties. In the decentralized approach, however, special mechanisms are needed to facilitate this. In particular, link prices should be communicated to the participating users to enable them to select routes. Two alternatives are possible here:

- CSPs employ a link-state based routing algorithm to gather this information. Link state updates could be created on a regular basis, or on a triggered basis, whenever a CSP updates its link price in an attempt to boost revenue. Network-wide link-state updates are sent to the users as well.
- CSPs register the current prices with, and send all updates to, a central database. Users query the database for current CSP-level topology and link prices at regular intervals.

In our design (§4), we adopt the latter of the two approaches because of its simplicity and because it offers more consistent network views to CSPs and users.

(2) **No global policies.** In the distributed approach, the actions of the different players are completely uncoordinated. This means that global desirable properties such as optimal net value or fairness cannot be guaranteed. The advantage of the decentralized mechanism, however, is that it offers flexibility to CSPs and users in implementing local algorithms, rather than a central arbiter simulating their selfish behavior based on assumptions about their objectives.

(3) **Constant online operation.** In the centralized approach, the arbiter simulates CSP and end-user behavior until an equilibrium is deemed to have been reached. CSPs and end-users only use the parameters corresponding to the equilibrium. No equivalent arrangement can be made in the distributed case. Instead, we assume the distributed version of the system to be in a constant state of flux with prices changes and users alternating their paths. However, we propose a suite of CSP price change and user response algorithms in §4 which ensure that the system operates in a stable state with only minor changes in link loads and prices.

3.4 Deployment Path and Role of BGP

Before providing the details of either variant, we briefly outline how we envision RouteBazaar to be adopted in practice. We envision the RouteBazaar system to start out being using in a centralized fashion on a small scale with a few early adopters. We envision large end-networks and small ISPs with mission critical information to be active early users of this system. Third party overlay service providers, as well as current tier-1 and tier-2 ISPs could offer connectivity within the RouteBazaar system. As the scale grows, we envision the system to transition into a distributed set-up, perhaps with a much larger user and CSP-base. We envision the set of CSPs to be rich and diverse, composed of a wide variety of providers of different sizes and reach. Similarly, we also envision users with different demand-value distributions to coexist and use the system in parallel.

Since RouteBazaar runs the risk of not being able to route all user requests, we feel that the underlying BGP routing will continue to play a crucial role through the evolutions because it is the fallback option.

4. DESIGN DETAILS

We now expand on the algorithms used by various participants in RouteBazaar to arrive at a stable state for the system. Towards the end of the section we present a discussion of the performance metrics that can be used to evaluate the system, as well as factors that may influence these metrics. For ease of exposition, we describe the distributed architecture first. Again, for ease of exposition, we assume that the set of CSPs and users in either case is fixed and address the issue of accommodating slow churn in §4.5.

4.1 Distributed Architecture: CSP Behavior

The goal of every CSP is to price its links so as to maximize its revenue given the other CSPs' and users' choices. Since the CSP has incomplete information about others' strategies and moreover strategies change dynamically, the CSP faces an online optimization problem. Each CSP therefore employs a learning algorithm to determine the best strategy. The learning algorithm successively tries different prices at different iterations and gradually converges to a near-optimal one. The goal of the algorithm is to minimize the "regret" of the CSP, which is the difference between the optimal average revenue achievable through a single price and the average revenue obtained by the algorithm over all the iterations.

Below we present two iterative regret-minimizing learning algorithms. The first, the *epsilon-decreasing explore-exploit* algorithm [32], is a simple and natural learning algorithm that is proven to minimize regret in a static environment (that is, if strategies of other CSPs and users are picked from an unchanging probability distribution, and do not adapt to this CSP's strategy). The second, the *hedge-bandit* algorithm [13, 6], provably minimizes regret even in worst-case dynamic online settings, and is therefore (from a theoretical viewpoint) most appropriate for our setting. A crucial aspect of both algorithms is their use of *history*. The algorithms track historical information on the net revenue derived at various prices to determine optimal price choices at each iteration.

We now describe the two algorithms. In the *epsilon-decreasing explore-exploit* algorithm, at every iteration the CSP either "explores" the space of prices by picking a price uniformly at random, or "exploits" by picking a price that has historically obtained the most revenue. At iteration t, an explore is performed with probability ϵ_t and an exploit is performed with probability $1-\epsilon_t$. In the beginning, to bootstrap, ϵ_t is set to a high value but as more and more history is accumulated it decreases with t. Specifically, we use $\epsilon_t = \min(N/t \log t, 1)$ for some constant N that depends on the size of the network.

In order to pick the best-price-so-far in an exploit step, the CSP maintains an exponentially weighted moving average (EWMA) of the revenue obtained by each potential price. After a certain large number of iterations, this approach would

Parameters: k, the number of different prices; $N = k^2$; α, the weighting parameter for EWMA.
Variables: Randomness parameter $\epsilon_t = \min(N/t \log t, 1)$; for every potential price i, a revenue estimate π_i.
Initialization: For every price i, $\pi_i = 0$.

At every iteration t do:
1. With probability ϵ_t, pick a price i uniformly at random and report it.
2. Otherwise (i.e. with probability $1 - \epsilon_t$), pick the price i with the maximum π_i and report it.
3. Let the revenue obtained at the current step be X. Update the revenue estimates for i as follows: $\pi_i = \alpha\pi_i + (1 - \alpha)X$.

Figure 1: Algorithm *epsilon-decreasing explore-exploit*—Regret-minimizing learning algorithm for CSPs

Parameters: k, the number of different prices; L = the capacity of the link times maximum price; randomness parameter $\epsilon = 0.01$ and weight $\delta = 0.01$.
Variables: Weights w_i and probabilities p_i for every potential price i. $W = \sum_i w_i$, and $p_i = w_i/W$.
Initialization: For every price i, $w_i = 1$ and $p_i = 1/k$. $W = k$.

At every iteration do:
1. With probability ϵ, pick a price i uniformly at random and report it.
2. Otherwise (i.e. with probability $1 - \epsilon$), pick a price i randomly from the distribution p and report it.
3. Let the revenue obtained at this step be X. Update w_i as follows: $w_i = w_i e^{\delta X/LP}$ where $P = \epsilon/k + (1 - \epsilon)p_i$. Other weights stay the same.
4. Update the probability vector p by setting $p_j = w_j/W$ for every j, where W is the new sum of all the weights.

Figure 2: Algorithm *Hedge-Bandit*—Regret-minimizing learning algorithm for CSPs

give more weight to newer knowledge gained and less to historical data compared to a simple average. This is especially important in a dynamic environment where demands come and go, because a price that was attractive historically may become unattractive over time. The EWMA approach allows the algorithm to adapt quickly to such changes. We present this algorithm formally in Figure 1.

The *hedge-bandit* algorithm also "explores" at every iteration with an ϵ_t probability. However, in this algorithm ϵ_t continues to remain at a constant value and does not decrease over time. The crucial difference between the two algorithms is in their "exploit" step. Instead of picking the best-in-history price at every exploit step, Hedge-bandit picks a price from a probability distribution that assigns high probability to prices with high revenues and low probability to other prices. The probability associated with prices that consistently perform poorly decreases exponentially over time. Therefore, Hedge-bandit quickly converges to good prices while not entirely disregarding prices that perform moderately. This allows it to adapt quickly to changes in the system such as the arrival or departure of demands. We present this algorithm formally in Figure 2.

4.2 Distributed Architecture: User Behavior

As described earlier, the goal of every user is to maximize the utility it derives from routing its traffic – this is the difference between the value it obtains from routing its traffic and the price it pays to the CSPs. Therefore, every user distributes its traffic over least cost paths between its source and destination, and uses the default BGP path for traffic with value less than the price on the least cost paths. In what follows, we use the term "user" to denote all traffic from some source to some destination with the same per-unit-flow value.

Parameters: Selfishness parameter ϵ_s; granularity parameter δ.
Variables: H, a list of paths used in the previous iteration; L, a list of currently least-cost paths; f_P, flow on path P; r, the amount of flow to be redistributed at any iteration.
Initialization: $H = \emptyset$.

At every iteration t do:
1. Construct L by finding all least-cost paths.
2. Initialize $r = 0$. For every path P in $H \setminus L$, increment r by $\epsilon_s f_P$ if $f_P \geq \delta$ and f_P otherwise; Set $f_P = f_P - r$.
3. Let $f_{\text{tot}} = \sum_{P \in L} \max(f_P, \delta)$. For every path P in L, set $f_P = f_P + r/f_{\text{tot}} \max(f_P, \delta)$.

Figure 3: Distributed flow-update algorithm for users

In the basic version of the user algorithm, each user maintains a list of the paths that it currently uses, as well as a list of currently least-cost paths. At every iteration the user removes all traffic from paths that are not currently least-cost, and spreads this traffic across least-cost paths in proportion to the traffic already carried by them. Additionally, the user treats the default BGP path as another available source-destination path in the system with price equal to the user's per-unit-flow value. When the price on the least-cost path is smaller than the user's value, it sends some fraction of its flow on the default BGP path. This algorithm is described in Figure 3.

We also study a smoother version of this flow-update algorithm with the goal of understanding whether slow updates to users' flow lead to better convergence properties. Similar algorithms [8, 7] have been employed to solve flow problems in distributed settings and have been theoretically shown to have good convergence times. In our setting in particular, because the system is dynamic, just like the CSPs, users may benefit from using historical data to determine good paths. In the smoother version, once again each user gradually moves its traffic from paths used in the previous iteration to those that currently charge the least price. In particular, each user removes an ϵ_s fraction of its flow from non-least-cost paths and distributes it over least-cost paths.

The parameter ϵ_s characterizes user selfishness — $\epsilon_s = 1$ corresponds to the basic "selfish" version where users always only use least-cost paths, while $\epsilon_s < 1$ corresponds to the "smooth" version where users give weight to historically good paths and occasionally route over non-least-cost paths. While on the one hand the smooth version is more robust to sudden random fluctuations in CSP prices, on the other hand, it may adversely affect performance because CSPs get slower feedback to their price changes. In §6 we study the impact of ϵ_s on the performance of the system.

4.3 Centralized Architecture

The goal of the central arbiter is to compute a "good"

$$\max \sum_u \sum_{P \in \mathcal{P}_u} v_u f_{P,u} \quad \text{subject to}$$

$$\sum_u \sum_{P \in \mathcal{P}_u, P \ni e} f_{P,u} \leq c_e \quad \forall e$$

$$\sum_{P \in \mathcal{P}_u} f_{P,u} \leq d_u \quad \forall u$$

u :	user with source s_u, destination d_u, value v_u, demand d_u
$\mathcal{P}_u$:	the set of all shortest paths for user u
$f_{P,u}$:	the flow of user u on path P
e :	edges with capacities c_e

Figure 4: Value maximizing LP for the central arbiter

equilibrium for the system. It does so through an iterative approach, by simulating CSP behavior and user behavior at each step. Specifically at each iteration it first picks prices for each CSP and then flow paths for each user, until some termination condition is met. We now elaborate on these two aspects of the centralized architecture.

Simulating CSP behavior: The central arbiter simulates the system as a repeated game among the CSPs. In any repeated game if each player employs a regret minimizing learning algorithm for picking its strategy, then the game converges to a correlated equilibrium [18, 12]. That is, no player has incentive to deviate from its strategy if other players continue to follow their strategies. The central arbiter therefore uses one of the two algorithms, *epsilon-decreasing explore-exploit* and *hedge-bandit*, described in § 4.1.

Simulating user behavior: At every iteration given prices for the CSPs, the central arbiter simulates user behavior by routing the flow of each user along the least cost path. Despite this constraint of following the best response, the arbiter has a lot of flexibility in routing flow because some source-destination pairs may have multiple least cost paths between them. Moreover, for some users the cost of the least cost path may be exactly equal to their value in which case routing their flow over this path or over the default BGP path brings equal utility to them. The arbiter may use this flexibility to implement any desirable social objective. Here we focus on the objective of maximizing the total social value of the system.

To achieve this objective, the arbiter solves the following max-value flow problem: It first determines for every user a list of all least cost paths between the corresponding source-destination pair. It then determines the amount of flow to be sent by the user along every such least cost path while honoring capacity constraints on edges and maximizing total value of the flow routed. This problem can be set up as a linear program (Fig 4) and solved using standard LP solvers.

While the solution to the LP maximizes the value routed subject to capacity constraints, it may not follow best response for every user. This is because if for some user the cost of the least cost path is strictly less than the user's value, it is in the best interest of the user to route its entire flow, while the LP may only route a fraction of the flow in order to satisfy capacity constraints. In order to rectify this, for every user with value strictly larger than the cost of the least cost path, the arbiter finds an arbitrary least cost path and routes the entire remaining flow of the user along this path.

We now elaborate on how the arbiter computes a list of all shortest paths for a user in order to set up the aforementioned LP. The arbiter first computes all-pairs-shortest-path distances given the CSP prices. It then determines for every source, the set of edges that lies on some shortest path starting at that source. An edge $(u \rightarrow v)$ lies on a shortest path starting at source s if and only if $\text{dist}(s, v) = \text{dist}(s, u) + \text{price}(u \rightarrow v)$. The arbiter then runs a depth-first search (DFS) starting from the source and only following edges that lie on shortest paths. Every time it encounters the destination in this DFS, it outputs the current path from the source to the destination. This algorithm takes time proportional to the number of shortest paths between a source-destination pair. We find in our simulations that the number of distinct shortest paths between any source-destination pair is at least one order of magnitude smaller than the number of edges in the graph. Therefore this algorithm is efficient in practice.

Termination condition: The arbiter could use a variety of tests to determine when to halt the simulation of the interaction between users and CSPs. In general, a good test is to check if the system is at a state where the total utility as perceived by both CSPs and users does not improve any further. The arbiter should also check for other conditions that the system should meet at equilibrium (§3.2). In our implementation of the arbiter, we check to see if in addition to the above conditions, over a certain large number I of iterations, the net utility in the system does not seem to improve or change significantly. Specifically, we check if the difference between the 95th and the 5th percentile net utilities is less than a small fraction (say 5%) of the 95th percentile.

4.4 Evaluating RouteBazaar Performance

A natural metric for measuring the performance of RouteBazaar is to quantify the total value it offers to the participants (CSPs and users). Define the *social value* of a state of the system to be the total utility of all the agents, or, the total value obtained by all the users, minus the prices paid by the users, plus the revenues (prices) earned by all the CSPs. Since prices are endogenous to the game this is equivalent to the total value obtained by all the users. Throughout this paper we use this metric to understand how well our system performs. In particular, we compare it against the optimal or the maximum possible social value achievable through global oversight while satisfying capacity constraints on edges. In the next section we discuss various factors that influence this metric and analytically study how the system behaves in various situations. Several other key metrics can also be used to evaluate the system; we discuss these in §6.

4.5 Churn

Users of RouteBazaar enter or leave the system only occasionally. RouteBazaar is designed to accommodate slow churn. In the centralized framework, we require that users

enter or leave the system *at fixed times*, e.g. at the beginning of a two hour period. Such churn is not so much an issue for the centralized framework because the arbiter can simply recompute the routes and link prices from scratch. Note that addition of new users may affect whether or not some current users can continue using routing over RouteBazaar. We require that existing users of the system check with the arbiter at the beginning of every time interval. If denied, they stop using the system (they may submit a new request). In the common case, users who have low willingness to pay are likely to be more affected by churn in the system, and we consider this to be an acceptable operating mode.

The distributed framework allows users to enter or leave at any time, but the system is designed for the user-set to change slowly. When single users are added to the system over time, few (if any) CSPs are likely to be effected, because small changes in demand do not cause immediate CSP response. In the rare case that multiple users join simultaneously, links could be overwhelmed and both the users routing over the links and the CSPs who own the links are dissatisfied. In such situations, the faster the system reconverges to a stable set of prices the better it is overall. We evaluate this situation in §6.

5. UNDERSTANDING THE FRAMEWORK

Since each CSP in our system is interested in maximizing its own revenue, the system performance in terms of this total benefit may not be optimal. In this section we perform a theoretical study of the gap between the optimal system performance and the performance at equilibrium.

A theoretical model: We analyze the following "pricing game". We are given a directed network (graph) with capacitated edges representing CSPs and a value-demand curve for each pair of nodes in the graph representing users. The game has two stages. In the first stage of the game each CSP or edge picks a price. In the second stage each user picks paths between its source and destination to send its traffic. We assume that users can split their traffic into infinitesimally small chunks, and spread it across multiple paths, or send fractional amounts of traffic. A given state in a game (in this case consisting of a set of prices and traffic pattern) is called a *Nash equilibrium* if no participant (user or CSP) wants to change its strategy unilaterally so as to improve its own utility. In the pricing game users are price-takers, that is, they merely follow a best response to the prices set by CSPs, and the responses of different users are decoupled from each other. Therefore, given the first stage strategies, the second stage strategies always form a Nash equilibrium, and the dynamics of the system is determined primarily by the first stage game.

We evaluate Nash equilibria in the pricing game by comparing the social value at an equilibrium to the maximum possible social value achievable while satisfying capacity constraints on edges. Recall that the social value for any state is defined as the total value of all the users in the system. The ratio of the worst value over all Nash equilibria to the optimal value is called the *Price of Anarchy (POA)* of the game. POA is always at least 1, and small values indicate good system behavior. In particular, a POA of 1 indicates that every Nash equilibrium is optimal in terms of social value.

Monopolies and the price of anarchy: Chawla and Roughgarden [14] showed that the price of anarchy of the pricing game depends primarily on the existence and number of "monopolies" in the network, and also on the distribution of traffic. A CSP is called a monopoly if there exist source-destination pairs such that all paths from the source to the destination go through the CSP. Specifically Chawla and Roughgarden proved the following:

Theorem 1 ([14]) *When all the demands in the network have the same source and destination, the POA of the pricing game is* 1 *in the absence of monopolies, at most* $\log L$ *when there is one monopoly, and can be unbounded otherwise. When the instance contains* $k > 1$ *monopolies, there always exists one equilibrium with social value at least an* $O(L^k)$ *fraction of the optimal. Here* L *is the ratio of the maximum per-unit-demand value to the minimum per-unit-demand value.*

This theorem shows that networks that contain great disparity among users in terms of per-unit-demand values, or that contain long source-destination paths (leading to a large k), can have poor equilibria. Furthermore, it suggests that eliminating monopolies from a network may be sufficient to ensure good performance. Unfortunately the latter is not true in networks with multiple sources of traffic. Specifically, by making certain parts of a network congested, it is possible to create so-called "virtual monopolies" in the network that lead to poor performance. A link is called a virtual monopoly if there exists a source-destination pair such that all of the flow routed from the source to the destination is carried by the link (because all alternate routes are too expensive). Chawla and Roughgarden further proved the following theorem about networks with multiple sources. Here the sparsity of the network is a measure of congestion in the network and is defined (in single-destination instances) as the maximum fraction of each demand that can simultaneously be routed in the network while satisfying link capacities.

Theorem 2 ([14]) *The price of anarchy in a network with multiple sources can be unbounded. When all users have a common destination (but different sources) and identical value-demand curves, and the graph is a so-called "traffic spreader", there exists a Nash equilibrium with social value at least a* $1/\alpha$ *fraction of the optimal social value, where* α *is the sparsity of the network.*

Monotone Hazard Rate distributions: As mentioned earlier, the above results indicate that network performance gets worse as the disparity between different users in terms of value per-unit-demand increases. These negative results are quite pessimistic in that the worst bounds arise from unnatural value-demand curves that are unlikely to occur in practice. We now show that when value-demand curves satisfy the "monotone hazard rate (MHR)" condition, the price of anarchy of a network improves considerably and does not

depend on the value disparity. That is, all Nash equilibria in the system are good. The MHR condition, defined below, is widely used in game theory and economics to characterize commonly occurring value distributions (see, for example, [11, 9]). Most natural distributions such as the uniform, normal, exponential, power-law (for exponents greater than one), Laplace, chi-square, etc. satisfy this condition [9].

Definition 1 (The Monotone Hazard Rate (MHR) condition) *For a given user and a value v, let $F(v)$ denote the fraction of traffic for which the user is willing to pay a per-unit-demand value of at most v. Let f denote the derivative of the function F. The MHR condition states that the "hazard rate" of the value-demand curve, $h(v) = \frac{f(v)}{1-F(v)}$, is a monotonically non-decreasing function of v.*

Theorem 3 *Consider an instance of the pricing game in which all users have the same source and destination. If the aggregate value-demand curve for the system satisfies the MHR condition and the network contains k monopolies, the POA of the game is at most e^k.*

PROOF. As in the definition above, let $F(v)$ denote the total fraction of traffic with per-unit-demand value at most v. Consider an arbitrary Nash equilibrium for the game. If the network is saturated under this equilibrium it already operates optimally. Otherwise, we note that all non-monopolies in the network charge a price of 0. (We omit an argument for the sake of brevity.) Now consider the monopoly charging the maximum price at this equilibrium; say that the price is p^*. Let $P + p^*$ be the total price charged by all the monopolies (this is the price paid by all the users that are routed). Then $p^* \geq (P + p^*)/k$. If this monopoly changes its price to p, its revenue is given by $p(1 - F(P + p))$. Since the system is at equilibrium and so p^* is the price maximizing this revenue, upon differentiating the function we get:

$$1 - F(P + p^*) - p^* f(P + p^*) = 0$$

That is, $h(P') = 1/p^* \leq k/P'$ where $P' = P + p^*$.

Now, by the MHR condition, h is a non-decreasing function, therefore, $h(t) \leq k/P'$ for all $t \leq P'$. Finally, we note that by definition, $F(t) = 1 - \exp(\int_0^t h(x)dx)$. Therefore, $F(P') = 1 - \exp(\int_0^{P'} h(x)dx) \leq 1 - \exp(k)$.

The total traffic admitted by this equilibrium is therefore at least an e^k fraction of all traffic. Therefore, the total social value of this equilibrium is at least an e^k fraction of the total value of all the users, and so also at least an e^k fraction of the optimal social value. □

The result above focuses on single-source single-sink networks containing monopolies. However it is also relevant to more general settings of multiple-source multiple-sink networks without monopolies because, as described earlier, such instances can contain virtual monopolies. In a companion paper currently under submission we show that both the upper and lower bound given above extend to multiple-source single-sink networks when all users have identical value-demand curves that satisfy the MHR condition.

Topology	Nodes, Edges
Rocketfuel-based	50, 100
Power law	46, 93
Random	50, 93
Regular	50, 100

Table 1: Details of topologies used in this study. For Random graphs, the probability of each edge is 0.08. In the regular graph, each node has a degree of 4.

These results show that the performance of the system degrades as the value disparity between different users increases, as congestion increases (equivalently sparsity decreases), or as the number of hops in source-sink paths increase (leading to a larger number of virtual monopolies). However, if all demand-value curves satisfy the MHR condition defined above, the system performs well at *any* equilibrium.

6. RouteBazaar EVALUATION

We conduct a variety of simulations to study the performance of RouteBazaar in realistic settings. We study several CSP interconnection topologies that may arise in practical instantiations of RouteBazaar. We also experiment with a few different distributions of user demand-value vectors. Our primary goal is to understand the global performance of the system (and compare against the theoretical results outlined above). Our key metric of interest is the social value derived by the system relative to the optimal social value, that we also refer to as "efficiency", under the different settings. This metric measures the ability of users to obtain as much benefit as possible from the system while allowing the CSPs to derive reasonable revenue, and indicates whether the system operates to its full potential. Note that this metric is different from POA because POA is a measure of efficiency of the worst Nash equilibrium; in general, when multiple equilibria exist, our system may converge to a good equilibrium. We also examine lower-level details of the system, e.g. the performance of various CSP and end-user algorithms designed for system robustness and stability, convergence behavior and re-convergence under churn, CSP revenues, link loads, link price distributions etc. A final goal is to compare the centralized and distributed variants of our system. The centralized approach is able to ensure sound global properties and our goal is to understand how close a completely decentralized approach can come to achieving such properties.

6.1 Experimental Setup

Topologies. We conduct experiments using several different CSP interconnection topologies. Our baseline topology is one derived from the ISP topologies discovered in [28]. We extracted a subset of 50 highly populated tier-1 PoPs in North America and the edges between them (100 edges in all).

We generated three other synthetic topologies: Power-law random graphs, Erdos-Renyi random graphs and regular graphs. The details of these graphs are shown in Table 1.

In all the graphs described above, every bi-directional edge is replaced by two individual links allowing traffic to flow

Algorithm	Parameters
Explore-Exploit (EE)	$\alpha = .95, N = 10000$
Hedge-Bandit (HB)	$\epsilon = .01, \delta = .01$

Table 2: Parameters for learning algorithms.

in opposite directions. Link capacities are drawn at random from one of two values—25 or 100 units. We force links flowing in opposite directions between a pair of nodes to be the same capacity. We also experimented with other link capacity distributions and our high level observations remain qualitatively similar in those settings.

Link costs. Each link is assigned an initial cost generated randomly between 0.1 units and 5 at a granularity of 0.1 units. The monetary value of "units" is arbitrary and would be determined by real demands in practice. CSPs adjust their link costs according to the algorithms presented in §4. We ensure that link costs stay in the range $[0.1, 5.0]$.

Sources and sinks. We use a multi-source multi-sink setting throughout our evaluation. Sources of traffic, i.e., users in RouteBazaar, could be located at any node in the above topology. However, the sinks are located at the 16 most highly populated nodes, where we assume that node population is proportional to degree (when degrees are similar, we pick 16 nodes at random).

User demands. To determine the amount of demand in the system, we use the gravity model: the net demand between a pair of nodes is proportional to the population of the corresponding nodes. The total number of users in the system is proportional to the net demand and each user has a single unit of demand to one of the 16 destinations.

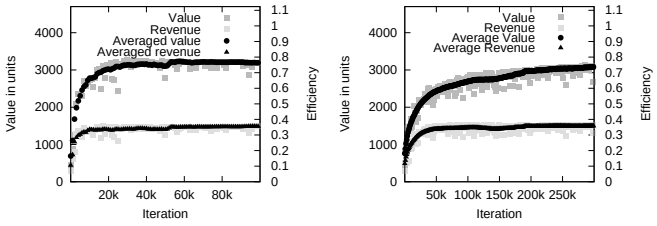
User values. We study a simplistic setting where all users have a demand-value vector containing a single entry, with demand set to unity. Each user is assigned an associated value with the unit demand it imposes. The values are drawn from the same range as link costs, $[0.1, 5.0]$ units.

We experiment with several value distributions. The first is a Zipfian distribution with an exponent of $\alpha = 1$. This represents a typical income model and satisfies the MHR condition. The second set of values is drawn uniformly at random from the range and also satisfies the MHR condition. The third is a bimodal distribution where there is significant disparity in user values: they are drawn from the ranges [0.1,1] or [4.5 and 5], with 90% of values originating from the first interval. This does not satisfy the MHR condition.

Parameters. The algorithms we proposed in §4 each use a set of parameters. We conducted a variety of simulations to understand how to select these parameters. Without going into the details and in the interest of brevity we briefly list values selected for key parameters in Table 2. These parameters ensure that the algorithms offer the fast convergence to equilibrium and good overall system performance in all scenarios we consider.

6.2 Centralized RouteBazaar

Our first set of simulations considers the centralized RouteBazaar framework under different settings. We first show that the system converges and is efficient in practice, and then



(a) EE Algorithm (b) HB Algorithm

Figure 5: Convergence: Rocketfuel topology, with Zipfian value distribution.

Topology	Value Distribution	Convergence	Efficiency
Rocketfuel	Zipfian	52,000	0.78
Rocketfuel	Uniform	53,000	0.70
Rocketfuel	Bimodal	53,000	0.75
Random	Zipfian	50,000	0.74
Random	Uniform	50,000	0.74
Random	Bimodal	50,000	0.79
Regular	Zipfian	52,000	0.68
Regular	Uniform	52,000	0.66
Regular	Bimodal	53,000	0.75
Power law	Zipfian	50,000	0.68
Power law	Uniform	50,000	0.67
Power law	Bimodal	50,000	0.73

Table 3: Centralized framework, with CSPs using EE.

compare the two CSP learning algorithms.

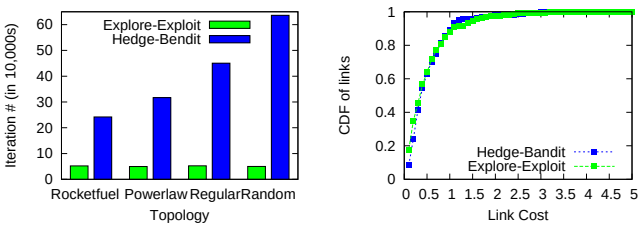
Convergence. Figure 5 shows the evolution over time of social value and total revenue at the central arbiter in the centralized RouteBazaar system for one of our experiments. We present these results for both CSP learning algorithms—EE (epsilon-decreasing explore-exploit) and HB (hedge-bandit). Note that both the social value and revenue curves flatten out after 50,000 iterations for EE (Fig 5(a)) and 200000 iterations for HB (Fig 5(b)). This indicates that the arbiter's simulation of user and CSP interaction reaches a stable state over time.

Using the termination condition described in §4.3 the arbiter declares convergence and outputs link prices and user routes at around 52,000 and 230,000 iterations for the two algorithms, respectively. In the rest of this section we refer to the state of the system at these points of time as the equilibrium (although it may not correspond to an actual Nash equilibrium), and say that the system has converged. We tried several other scenarios and observed convergence in all of them — the iteration at which convergence was observed is shown in column 3 of Table 3 for the EE algorithm.

Note that the two CSP learning algorithms differ in their convergence speed. We revisit this issue later in this section.

Efficiency. We now discuss a key measure of the overall performance of centralized RouteBazaar, the system efficiency, as defined earlier. The higher the efficiency, the closer the system operates to its capacity.

Table 3 shows the efficiency of the system upon convergence for different topologies and user value distributions. We assume that the EE algorithm is used to emulate CSP learning. In general, we note that despite allowing users and CSPs to interact selfishly with each other, our system arrives at a reasonably efficient outcome in most of the situations we studied. Although not 100% efficient, the centralized



(a) Convergence Time (b) Distribution of Link Prices

Figure 6: Comparing EE and HB: Zipfian user values over different topologies

RouteBazaar framework is able to perform at 60%-80% efficiency. We note that system performance at equilibrium is significantly better than indicated by the worst-case analysis. In particular, value distributions that were shown to result in undesirable outcomes by prior analytical results—e.g. the bimodal distribution—still result in fairly efficient outcomes in the situations we have analyzed.

Comparing CSP Learning Algorithms. We now compare the two CSP algorithms on various topologies. The results of the comparison are shown in Figure 6, where we study the time taken for the system to converge and the link prices derived by either algorithm. Recall that while HB has been theoretically shown to converge in arbitrary settings [13, 6], no such result is known for EE.

Several points are worth noting: Figure 6(b) shows that the algorithms result in roughly the same settings of link prices at equilibrium. We also found that the two algorithms differ negligibly in system efficiency at equilibrium (specifics not shown for brevity). However, Figure 6(a) indicates that HB takes a lot longer (4-10 times slower) to converge to an equilibrium in practice. In the HB algorithm, link prices could change more abruptly and more often until convergence is reached because the algorithm explores mediocre prices in addition to the best ones. In contrast, EE forces link prices to be more stable and slowly evolving. The same reason also contributes to HB converging slower on topologies with greater potential for route diversity (e.g. compare Regular against Power law in Figure 6(a)).

On the whole, it is unclear which algorithm is more suitable in practice. On the one hand, HB is designed for the worst case and offers theoretically sound properties, and hence may be more appealing under a wide spectrum of situations. On the other hand, the EE algorithm seems to work well in most practical situations anyway. Because of its simplicity and reasonable performance we expect that EE will be adopted in practical implementations.

6.3 Distributed RouteBazaar

We now evaluate the distributed variant of the RouteBazaar framework. Note that CSPs continue to use one of the two learning algorithms, EE and HB, in the distributed setting as well. We start with examining the impact of user response in this framework on system performance.

Understanding User Response. In the distributed setting, users employ the flow update algorithm described in Figure 3 to adapt to changes in prices at the CSPs. The algorithm gradually moves an ϵ_s amount of the users' traffic from currently-used sub-optimal paths to the new best priced paths. We study how the choice of ϵ_s impacts system performance and in particular if allowing users to be "fully selfish" ($\epsilon_s = 1$) leads to poor outcomes.

CSP Price Algorithm	ϵ_s	Efficiency	Convergence
EE	0.1	0.39	73,000
EE	0.5	0.55	91,000
EE	1	0.69	52,000
HB	0.1	0.19	100,000
HB	1	0.61	150,000

Table 4: The impact of ϵ_s in the distributed case; Rocketfuel topology and Zipfian value distribution

Topology	Convergence	Efficiency	Efficiency - centralized
Rocketfuel	52,000	0.69	0.78
Random	59,000	0.62	0.74
Power law	88,000	0.61	0.68
Regular	52,000	0.59	0.68

Table 5: Efficiency of the distributed approach for Zipfian value distribution.

Table 4 summarizes our findings for both the CSP learning algorithms (EE and HB). Surprisingly, we find that using large ϵ_s leads to robust performance—allowing users to be selfish is not bad from a global viewpoint. On the other hand, constraining users from reacting selfishly (i.e. using small ϵ_s) seems to lead to inferior system-wide performance at equilibrium. This is because when users use small ϵ_s, CSPs get very slow feedback about the effect of their price changes. CSPs naïvely believe that the price changes they made caused only a small increase/decrease in the amount of traffic they carried and the revenue it produced. This "incorrect" revenue "pollutes" their history, causing sub-optimal prices to be charged and the system to converge to a worse equilibrium.

The results are consistent across both CSP learning algorithms and across other scenarios (i.e. network topologies and user value distributions — not shown for brevity).

Comparison with the Centralized Variant. A key difference between the centralized and distributed variants is that the central arbiter in the former can optimize global objectives to the extent allowed by CSP and user constraints. In particular, when the system admits multiple stable states, the central arbiter can suggest the best one to all the participants, in particular the one that maximizes social value. Such optimizations cannot be performed in the distributed framework. A natural question is to examine the extent to which the distributed framework suffers due to the lack of this global optimization. Our evaluation, summarized in Table 5, indicates that the distributed approach performs nearly as well as the centralized one. In particular, the difference between the distributed and centralized variants in terms of efficiency is just 0.07 to 0.12. Thus, the distributed approach in practice can offer reasonable system-wide performance despite the lack of a global perspective.

Recall that the distributed approach is designed to run in a continuous online fashion and is always in a state of flux. To

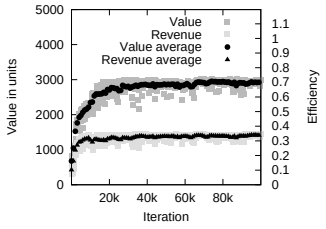
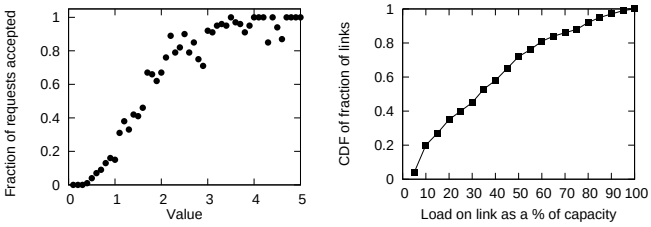


Figure 7: Distributed Convergence: Rocketfuel topology, Zipfian user values and CSPs using the EE algorithm



(a) Users Utilizing the System (b) CSP Link Utilizations

Figure 8: Distributed RouteBazaar **at equilibrium**

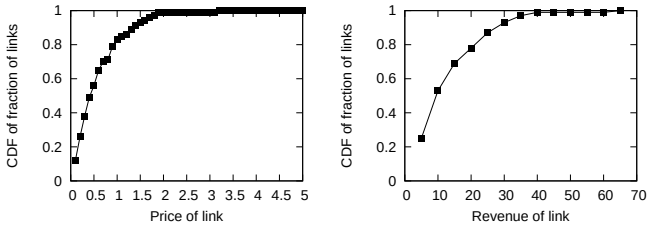
understand if the system oscillates, we applied a variety of tests. We monitored the change in the net social value of the system and net CSP revenues over time in various settings. In particular, we applied the centralized termination condition (Section 4.3) to the distributed simulation state to examine if the latter even converges and if so how the convergence rate compares to the centralized approach. Our observations are summarized in column 3 of Table 5. The distributed framework achieves convergence in all cases. (Note that iterations to convergence mean different things in the context of the centralized and distributed approach, because the latter involves direct user-CSP interaction.)

For completeness, in Figure 7, we show the evolution of the distributed framework over multiple iterations when CSPs use the EE learning algorithm and users employ $\epsilon_s = 1$ in their flow update algorithms. We find that the social value of the system reaches a steady state in about 50,000 iterations. Net CSP revenue is also stable beyond this point.

6.4 RouteBazaar at Equilibrium

RouteBazaar is designed to attract participation from users and CSPs. We now examine the status of RouteBazaar at equilibrium to check if this design requirement is met satisfactorily. We consider the distributed approach and take a snapshot of the system after a certain large number of iterations. We use the Rocketfuel topology with Zipfian value distributions, the EE algorithm for CSPs and assume selfish user response ($\epsilon_s = 1$ for user flow update). Our observations for the centralized approach are qualitatively similar.

In Figure 8(a), we show the fraction of users that are allocated routes at each value for the distributed framework. We see that very few users with low values (< 1 unit) get to route using the system. In contrast, almost all users with high values (> 3 units) route using the system. We note that some users with high values are denied service—this could happen because of where they are located and how many other high-



(a) Link Price (b) CSP Revenue

Figure 9: Distributed RouteBazaar**: CSPs at equilibrium**

value users they are competing against. In general, the system shows the expected behavior, allowing users who value the routes the most to be able to use RouteBazaar.

In Figure 8(b), we show the utilizations of links owned by different CSPs. A small fraction of the links (20%) have link utilizations below 10%. About 30% of the links have 50% or higher utilization. This suggests that most CSPs are able to attract enough traffic onto their links using RouteBazaar. In Figure 9 we show the link prices and revenues of CSPs. We find that in an overwhelming fraction of the cases, the equilibrium prices are 2 units or lower (Figure 9(a)). From (Figure 9(b)) we see that revenues are fairly evenly distributed across the CSPs and there is no significant skew. Roughly 45% of the revenues are in the 10-40 unit range. The reasonable distribution of revenues and link utilizations indicate that service providers would be willing to take part on our RouteBazaar framework. Since users with moderate to high values are able to route in RouteBazaar, our system would be able to attract enough users as well.

6.5 Churn

In this section, we discuss the performance of RouteBazaar when users enter and leave. Our discussion in §4.5 argues that slow churn can be accommodated in both the centralized and the distributed frameworks. In this section, we evaluate the rare situation where the system faces unexpected churn. We focus on the distributed setting.

To test the flexibility of handling "extreme" churn, we ran the rocketfuel topology with Zipfian user values with the EE CSP response behavior. After the system converges for an initial set of demands, we add new users with a total demand of 1% to 5% of the existing demand. The reconvergence times for these scenarios are presented in Table 6. We also show the efficiency of the system after reconvergence and the "restart" efficiency which reflects what would have been achieved had the entire set of users been in the system from the start.

We note that although the system converges close to the restart efficiency under extreme churn, it takes a long time to reconverge. A potential concern in this case is that at certain times links may become overloaded due to extra demand in the system, causing dissatisfaction to users as well as CSPs. On the other hand, if all link loads are always < 1 prior to reconvergence, then users with the willingness to pay still get service, and CSPs still get good revenue, and long reconvergence times are of less concern. To address this, each

Demand Increase	Reconvergence	Efficiency	Restart Efficiency	α_b needed
1%	14,000	0.60	0.62	.01
5%	28,000	0.54	0.58	.08

Table 6: Reconvergence on distributed simulations on Rocketfuel topology and Zipfian user values. We assume CSPs use the EE algorithm, which we updated to accommodate churn—in this approach, CSPs employ a random price at every iteration with a fixed small probability, similar to the HB algorithm.

k	Convergence iteration	Efficiency
3	43,000	0.26
4	52,000	0.59
6	219,000	0.70

Table 7: Efficiency in k-regular graphs

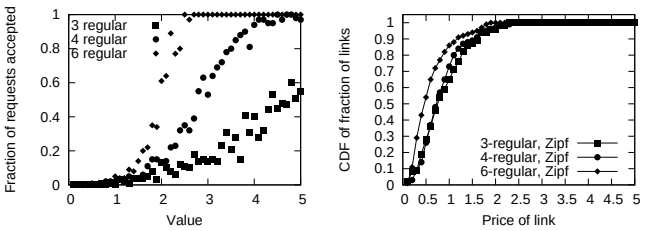
CSP link can have a small fraction of its capacity α_b set aside as backup. Each CSP sets or alters prices to operate below $1 - \alpha_b$ of its link capacity at equilibrium; if not, CSPs assume that no revenue is made on the link and change price to bring the load down. The spare α_b fraction of capacity can thus be utilized to accommodate relatively heavy churn and ensure that most users are satisfied even during churn. In our simulations using Rocketfuel topologies, we found that setting $\alpha_b = 8\%$ (conservatively) is sufficient to ensure that a link is not overwhelmed beyond its full capacity with high probability (see Table 6).

6.6 Factors Affecting the System

In §5, we outlined several factors that may affect the RouteBazaar framework, including disparity in user values, network topology, and the presence of real or virtual monopolies. Our simulations were designed to evaluate RouteBazaar to understand the impact of these factors in practice. So far we have presented results detailing the impact of the first two factors. As our results show, the practical impact of these factors seems minimal and RouteBazaar seems to offer reasonable performance under a broad range of situations.

As we show earlier, RouteBazaar presents a low barrier to entry to CSPs. Therefore we don't expect the system to contain monopolies. We conducted a brief analysis to understand the impact of virtual monopolies by varying the connectivity of the network, and thereby impacting the hop-lengths of source-destination paths. Theorems 1 and 2 suggest that as the hop-lengths of paths increase, leading to a potentially larger number of virtual monopolies for any user, the system performance relative to the optimal social value worsens. We verify this behavior through simulations.

In this set of experiments we selected the baseline regular graph topology (with per node degree of 4) and changed the average degree of each node to both increase and decrease route diversity and path lengths (use per node degrees of 3 and 6 respectively). We employed the distributed RouteBazaar framework in this analysis. Our observations are summarized in Table 7. We note that in networks where path diversity is severely restricted (correspondingly, there is a high incidence of virtual monopolies), the system-wide efficiency suffers significantly.



(a) Users Utilizing the System (b) Link Prices

Figure 10: User and CSP properties in regular graphs.

Figure 10 shows the distribution of values of users who route over RouteBazaar, and the link prices selected by CSPs, in the different regular topologies. As expected, link prices increase slightly with decreased path-diversity (Figure 10(b)), because virtual monopolies tend to charge higher prices. More tellingly, the set of users using the system is very different. In topologies with larger degree, greater fractions of users who value the routes more end up using the system. In such topologies there are both a larger number of, and shorter on average, paths available. For the degree-3 graph, significant fractions of even the high-value users are denied. In this case, the effect of higher link prices is aggravated by longer paths—longer paths mean that users' money will only get them so far, leading to poor social outcome.

7. DISCUSSION

Payments between users and CSPs. In the centralized approach, the arbiter can also play the role of mediating money exchange between each user and the set of service providers it routes over. The arbiter can tally the net amount owed by a user to each service provider and issue a monthly bill. In the distributed setting, a separate accounting framework would be necessary. The accounting mechanism could be based on the current approaches ISPs employ when charging customers on the basis of usage [25]. Using such an approach, each CSP tracks the net amount owed by individual users (note that the users may not be directly connected to the CSP links) and directly issues bills to the users. We believe that two factors will make the accounting problem relatively simple from the view-point of CSPs: (1) The users in our setting are large stub networks and small-to-medium ISPs. Thus, they are few in number and it should be possible for CSPs to track their usage without incurring too much overhead. (2) We expect the system to mostly operate at steady state with few arrivals and departures of users, and relatively slow changes to demand-value vectors (based on time of day, for example). As a result, the usage profiles (i.e. the set of users routing over each link and the amount of capacity consumed) and link prices of CSPs are likely to change slowly, if at all.

Multi-level RouteBazaar. Depending on the equilibrium prices selected by CSPs and network topology, it is possible that several users are denied service in RouteBazaar (note that the net social value of the system could still be high as users with high willingness to pay are unlikely to be denied).

To be more inclusive, a multi-level framework akin to "Paris Metro Pricing" [23] could be adopted: CSPs can offer multiple levels of service where each is priced independently and prices of different levels are significantly different. To use such a system, users would have to include the specific service-level they desire in each CSP alongside the source route. Users with low overall values now have the flexibility of (possibly) leveraging the lowest service level in each CSP and such paths are still better than routing over default BGP paths (e.g., in terms of availability). In future work we hope to evaluate such a multi-level RouteBazaar framework.

Information granularity. In RouteBazaar CSPs base their pricing decisions only on the net volume of traffic attracted by their links at a certain price setting, and how the revenue derived compares against that obtained at other price settings. Thus, the CSPs only use *coarse grained information*. They know and use very little information about users, e.g., how many there are and what their demand-value vectors are. CSPs are also assumed to be unaware of the prices selected and revenues derived by other CSPs in the system. Employing limited coarse-grained information helps keep the CSP price adjustment algorithms simple. We do note that it is possible for a given CSP to use additional information and set its prices more intelligently to further improve its revenue; both the centralized and distributed frameworks could be extended with such intelligence and this further enhances the flexibility our system offers to CSPs. Two points are worth mentioning in the context of such smarter algorithms: (1) Our analytical bounds on the price of anarchy of the system hold even with such flexibility. (2) In practice, given that CSPs already obtain reasonable revenues with the simpler approach (Section 6), more intelligent algorithms are unlikely to further boost revenues. We plan to explore the practical effects of using finer-grained information in future work.

8. CONCLUSIONS

BGP's inherent rigidity has forced users of the Internet to explore solutions like overlay routing and multihoming, which are either undesirable or inadequate. In this paper, we present an economically motivated approach called RouteBazaar for overcoming the rigidity. RouteBazaar offers flexibility and visibility to users who are willing to pay. It encourages participation and competition among service providers by allowing them to dynamically alter prices for the services they offer and giving them the ability to attract users who are not directly connected to them. Our evaluation using both analysis and simulations suggests that the framework is both efficient and stable under mild but realistic assumptions about user requirements and network topology.

Our design of the centralized and distributed versions of RouteBazaar are in no way complete or final. We expect several innovations to be possible, for instance, in service provider and user response algorithms and in accommodating high churn a lot more gracefully. Our ideas in this paper present likely starting points for these.

RouteBazaar is meant to support BGP, not replace it. As RouteBazaar adoption grows, and more scalable alternatives are introduced, we envision greater amounts of traffic using the system. At all times, however, RouteBazaar depends crucially on BGP, or some other protocol that provides reasonable bare bones reachability, as a fallback option.

9. REFERENCES

[1] D. Acemoglu and A. Ozdaglar. Competition and efficiency in congested markets. *Mathematics of Operations Research*, 32(1):1–31, 2007.
[2] A. Akella, J. Pang, S. Seshan, A. Shaikh, and B. Maggs. A Comparison of Overlay Routing and Multihoming Route Control. In *SIGCOMM*, 2004.
[3] D. Andersen, H. Balakrishnan, M. Kaashoek, and R. Morris. Resilient Overlay Networks. Banff, Canada, Oct. 2001.
[4] D. G. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. Rao. Improving Web availability for clients with MONET. In *NSDI*, 2005.
[5] E. Anshelevich, B. Shepherd, and G. Wilfong. Strategic network formation through peering and service agreements. In *FOCS*, 2006.
[6] P. Auer, N. Cesa-bianchi, Y. Freund, and R. E. Schapire. Gambling in a rigged casino: The adversarial multi-armed bandit problem. In *FOCS*, 1995.
[7] B. Awerbuch and R. Khandekar. Greedy distributed optimization of unsplittable multicommodity flows. In *PODC*, 2008.
[8] B. Awerbuch and R. Khandekar. Stateless distributed gradient descent for positive linear programs. In *STOC*, 2008.
[9] M. Bagnoli and T. Bergstrom. Log-concave probability and its applications. *Economic Theory*, 26(2):445–469, 08 2005.
[10] H. Ballani, P. Francis, and X. Zhang. A study of prefix hijacking and interception in the internet. In *SIGCOMM*, 2007.
[11] R. E. Barlow, A. W. Marshall, and F. Proschan. Properties of probability distributions with monotone hazard rate. *Ann. Math. Stat.*, 34(2):375–389, 1963.
[12] A. Blum and Y. Mansour. Learning, regret minimization, and equilibria. In *Algorithmic Game Theory*, chapter 4, pages 79–102. 2007.
[13] C. Burch. *Machine learning in metrical task systems and other on-line problems*. PhD thesis, Pittsburgh, PA, USA, 2000. Chair-Blum,, Avrim.
[14] S. Chawla and T. Roughgarden. Bertrand competition in networks. In *Symposium on Algorithmic Game Theory*, pages 70–82, 2008.
[15] C. Estan, A. Akella, and S. Banerjee. Achieving good end-to-end service using Bill-Pay. In *ACM HotNets-V*, Irvine, CA, Dec. 2006.
[16] C. Estan, A. Akella, and S. Banerjee. A la carte: An Economic Framework for Multi-ISP Service Quality. Technical Report 1591, UW-Madison, Mar. 2007.
[17] B. Godfrey, S. Shenker, and I. Stoica. Pathlet routing. In *ACM HotNets*, 2008.
[18] S. Hart and A. Mas-Colell. A simple adaptive procedure leading to correlated equilibrium. *Econometrica*, 68(5):1127–1150, 2000.
[19] A. Hayrapetyan, É. Tardos, and T. Wexler. A network pricing game for selfish traffic. In *PODC*, pages 284–291, 2005.
[20] J. R. Lane and A. Nakao. Path brokering for end-host path selection: Toward a path-centric billing model for a multipath internet. In *ReArch Workshop*, 2008.
[21] J. Mackie-Mason and H. Varian. *Public Access to the Internet*, chapter Pricing the Internet. MIT Press, 1995.
[22] R. Mahajan, D. Wetherall, and T. Anderson. Understanding bgp misconfiguration. Pittsburgh, PA, Aug. 2002. (to appear) `http://www.cs.washington.edu/homes/ratul/bgp/bgp-misconfigs.ps`.
[23] A. M. Odlyzko. Paris metro pricing for the internet. In *ACM Conference on Electronic Commerce*, 1999.
[24] A. Ozdaglar and R. Srikant. Incentives and pricing in communication networks. In *Algorithmic Game Theory*. 2007.
[25] RouteScience Technologies, Inc. Routescience PathControl. `http://www.routescience.com/products`.
[26] S. Savage, A. Collins, E. Hoffman, J. Snell, and T. Anderson. The End-to-End Effects of Internet Path Selection. Boston, MA, Sept. 1999.
[27] S. Savage et al. Detour: A Case for Informed Internet Routing and Transport. 19(1):50–59, 1999.
[28] N. Spring, R. Mahajan, and D. Wetherall. Measuring ISP Topologies with Rocketfuel. In *ACM SIGCOMM*, 2002.
[29] H. Tangmunarunkit, R. Govindan, and S. Shenker. Internet Path Inflation Due to Policy Routing. In *SPIE ITCOM*, Denver, CO, Aug. 2001.
[30] É. Tardos and V. Vazirani. Basic solution concepts and computational issues. In *Algorithmic Game Theory*, chapter 1, pages 3–28. 2007.
[31] V. Valancius, N. Feamster, R. Johari, and V. Vazirani. Mint: A market for internet transit. In *CoNEXT Workshop on Rearchitecting the Internet (ReArch)*, 2008.
[32] J. Vermorel and M. Mohri. Multi-armed bandit algorithms and empirical evaluation. In *European Conf. on Machine Learning*, pages 437–448, 2005.
[33] W. Xu and J. Rexford. Miro: multi-path interdomain routing. In *SIGCOMM*, 2006.
[34] X. Yang, D. Clark, and A. W. Berger. Nira: a new inter-domain routing architecture. *IEEE/ACM Trans. Netw.*, 15(4), 2007.
[35] Z.-L. Zhang, Z. Duan, L. Gao, and Y. T. Hou. Decoupling qos control from core routers: A novel bandwidth broker architecture for scalable support of guaranteed services. In *SIGCOMM*, 2000.