Better Internet Routing through Intrinsic Support for Selfishness

Holly Esquivel, Chitra Muthukrishnan, Aditya Akella and Shuchi Chawla
University of Wisconsin, Madison

***Abstract*—The BGP routing system is inflexible and sub-optimal for both stub networks and ISPs. Stub networks are unable to obtain routes that meet their end-to-end requirements, and ISPs have limited flexibility in controlling their revenues and offering new services to a wide customer base. We present the S4R supplemental routing system to address these shortcomings. Technical soundness and economic viability are equal first class design requirements for S4R. In S4R, ISPs announce links connecting different parts of the Internet. ISPs can selfishly price their links to attract maximal amount of traffic. Stub networks can selfishly select paths that best meet their requirements at the lowest cost. We design a variety of practical algorithms for ISP and stub network response that strike a balance between accommodating the (selfish) objectives of all participants and ensuring efficient and stable operation overall. We employ large scale simulations of realistic scenarios to show that S4R operates at a close-to-optimal state and that it encourages broad participation from stubs and ISPs. We describe a prototype implementation using OpenFlow and show that it can support S4R effectively.**

I. Introduction

BGP enables stub networks, such as content providers and enterprises, to reach clients and services located throughout the Internet. ISPs employ BGP in a variety of ways to control traffic entering and leaving their networks, and in turn, to enhance their revenues. However, BGP suffers from key inflexibilities that impose constraints on both stub networks and ISPs today [4], [14]. BGP offers stub networks exactly one policy-constrained path per destination per ISP connection, with no guarantees on performance or availability. Thus, stub networks cannot flexibly meet the requirements of key network-based applications, such as satisfying the end-to-end performance constraints of real-time video or finance applications, especially during peak traffic periods [4], [18]. To overcome this, stub networks can enter into partial transit or paid peering contracts with multiple ISPs to support sensitive applications. Unfortunately, these contracts are binding and long-term in nature. Other finer-grained approaches [26], [5], [4], are either undesirable in practice or inadequate.

BGP is sub-optimal for ISPs, too [30]. ISPs have little flexibility in controlling their revenues and expanding their services to attract a larger customer base. While BGP import and export policies allow ISPs some control over their revenues, they require ISPs to rely on long-term bilateral contracts with peers and customers [23]. ISPs can offer performance guarantees for traffic within their own domain, but are at the mercy of those they contract with once traffic exits their own network. Moreover, there are no easy ways for an ISP to expand its customer base to stub networks located in places where the ISP has no "physical presence". Approaches based on tunneling (e.g., MIRO [30]) are inadequate because the tunnels must traverse multiple intermediate ISPs that may not offer the tunneled traffic the same level of high performance.

978-1-4673-0298-2/12/$31.00

Some prior efforts [12], [28], [2], [10], [9] have recognized the fundamental shortcomings of routing, namely, that it is neither aligned with important emerging stub network usage scenarios nor with ISP revenue and operational goals. However, these works focus either on (some of the) underlying implementation issues or on economic/theoretical analyses. To date, no work has both described a technical solution and evaluated its viability in practice, especially from an economic standpoint. For example, approaches such as multi-provider MPLS/VPNs [2] consider the technical issues in enabling stub networks to obtain inter-domain paths meeting their requirements, but they do not consider the crucial economic issues for both ISPs and stub networks (e.g., how to price paths to maximize revenue, how to select paths with best cost-performance trade-offs etc.), which impact whether or not such mechanisms are adopted in the first place. At the other extreme, game-theoretic models [10], [9] study selfish interactions among ISPs and stubs, and show that the result can be arbitrarily bad in some network settings; however, it is not clear if these results hold in realistic scenarios.

Our paper brings together both technical as well as economic issues to develop a compelling solution to BGP's intrinsic shortcomings. We describe the design and implementation of an economically-grounded routing system, called S4R, designed to supplement, not supplant BGP. S4R enables participating stubs and ISPs to behave *selfishly* in order to directly meet their local objectives. Thus, S4R offers its participants a great degree of flexibility, which fosters greater participation from them while not requiring any kind of global oversight. We evaluate S4R in a variety of realistic situations using metrics and models that are similar to those used in prior game-theoretical analyses and show that S4R is desirable for both stubs and ISPs. We show the S4R can be implemented effectively using the OpenFlow platform [22].

ISPs participating in S4R announce (virtual) links connecting different locations of the Internet. ISPs have the flexibility of dynamically altering the link prices so as to control the quality of their links and, more importantly, to attract arbitrary remote traffic and maximize their revenue. Stubs have the flexibility to select (or shift at any time to) paths with optimal cost-performance trade-offs for the specific application at hand. A stub will always be able to find paths that best meet its application-level requirements as long as it has the willingness to pay for it. S4R's approach to enabling selfishness of its participants directly aligns with the selfishness models studied

in prior worst-case theoretical analyses. However, we find, surprisingly, that S4R leads to robust outcomes in practice contrary to what theory suggests [10], [9].

We describe two implementations of S4R: centralized and distributed. We develop learning-based response algorithms both for ISPs to change prices to maximize revenue and for stubs to reroute traffic selfishly when prices change. Our algorithms are designed to balance the need to accommodate stub and ISP selfish objectives with ensuring stable and efficient system-wide operation. Our algorithms help S4R reach a *correlated equilibrium* where every participant is content with their own choice given others' choices.

Through an extensive evaluation using a variety of realistic and synthetic scenarios, we answer key technical and economic questions: *(i)* Does S4R reach a stable operating point? How does S4R's equilibrium performance compare to socially optimal outcomes? *(ii)* How do structural properties, including network topology, presence of monopolies, stub network value distributions, etc., impact the overall performance? *(iii)* How well does S4R support various stub network usage patterns and ISP revenue objectives? And, *(iv)* Can we overcome practical implementation challenges in supporting S4R?

We find that, in all scenarios, the net performance derived by S4R stubs (in centralized or distributed cases) is roughly 30% away from the best possible social outcome (i.e., where all ISPs are altruistic and provide globally-optimal routes). S4R can support a variety of stub usecases, which are poorly supported today, equally effectively. Furthermore, there is no significant skew in ISP revenues. Thus, S4R functions efficiently and yields attractive outcomes for all participants despite their selfish actions. We find S4R is efficient as long as there is sufficient path diversity. Furthermore, the performance of S4R is unaffected by the variance in stub networks' willingness to pay. We also find that S4R can handle a modest degree of churn in practice. Using experiments on a small-scale testbed, we find that our prototype, which uses OpenFlow switches, can effectively support S4R.

Thus, our contributions are: (1) a supplemental routing system that aligns with the economic objectives and emerging requirements of ISPs and stubs, (2) the design of algorithms that ensure robust and efficient overall operation while accommodating selfish actions of individuals, (2) an implementation of the proposal using OpenFlow-based components, (3) a thorough evaluation of economic aspects under realistic scenarios, which prove the viability of the proposal despite worst-case theoretical results to the contrary, and, (4) a thorough examination of key technical and implementation challenges.

II. Motivating Examples

To exemplify the requirements for S4R, we discuss three scenarios which are poorly supported today. In each case, a stub desires end-to-end guarantees, and these must be simultaneously supported by one or more remote ISPs.

(1.a) Expected spikes in traffic, e.g., a popular Web service experiencing increased traffic during broadcasts of major events, sales or during releases of new software (see for example [3]). To offer good performance to its clients, the Web server may wish to purchase network paths with additional end-to-end bandwidth to ISP PoPs in large cities where most clients are located only when demand is known to be high (this only addresses the network bottleneck; the Web service would have to provision its server to handle the load as well). The site may wish to revert to BGP at all other times.

(1.b) Sensitive traffic requiring a fixed amount of end-to-end point-to-point or point-to-multipoint bandwidth, e.g., real-time HD video and Telepresence. Another class includes applications requiring high availability.

(2) Bulk transfers with deadlines such as video-on-demand services, e.g., NetFlix's "On Your TV", which may want to "push" popular content to caches located closer to stub networks in order to provide instant access. Stub networks would want their end-to-end transfers to complete before a pre-set deadline, but do not care about the actual transfer rates.

Today, it is possible to obtain inter-domain paths for the above by entering into paid transit or partial peering contracts with a collection of ISPs along chosen paths. However, such contracts are long term in nature: the stub cannot switch to an alternate path should its current choice be sub-optimal, not can it opt out should traffic demand not warrant the additional capacity [23]. Another possibility is for the stub network to multihome, but it cannot hope for any concrete performance guarantees beyond the first hop. Overlays may also be employed, but such schemes have poor interactions with the objectives of ISPs [25]. Techniques such as multi-provider MPLS and VPNs [11], [2] may be applicable, but existing proposals consider the technical details of distributed provisioning of end-to-end tunnels. But it is equally important to incorporate ISP and stub economic considerations and understand the implications of various choices for inter-domain route provisioning and route selection.

Design Requirements: S4R is a routing system for providing end-to-end routes accommodating scenarios such as the above, while incorporating the economic considerations of participants. S4R's design requirements are as follows:

(R1) S4R should be useful for stubs. Specifically: (R1.1) Stubs who have the means and willingness to pay should be able to find routes matching their requirements. (R1.2) There should be no "cheaper" route that also meets the stubs' constraints at any point in time. (R1.3) S4R should support a variety of stub usage scenarios equally effectively. **(R2)** To encourage ISP participation, S4R should allow them to set prices for the routes they offer in an arbitrary selfish manner to control link quality and attract remote traffic.

Next, we describe the key components of S4R (§II). We then describe ISP pricing and stub route selection (§IV), as well as router support issues (§V). Finally, we then describe prior pessimistic theoretical studies but show using extensive simulations that S4R is effective in practice (§VI).

III. S4R Overview

Stub networks. In S4R, stubs can obtain end-to-end paths between two network locations with some associated properties. We focus mainly on *performance* guarantees, but S4R can be used for other properties, such as avoiding specific

ISPs, traffic striping, and routing through intermediaries like DDoS filters [24]. Our framework applies to two types of stubs: (1) **large stubs** such as enterprises, Web services and campuses which require guarantees for the traffic they exchange with some network destinations; (2) **small regional ISPs** with footprints limited to a specific geographic region or a state. Such networks may wish to offer their customers some performance guarantees to popular remote destinations.

Usage scenarios have not received much attention in most prior work. In S4R, stub networks can place requests of four different types that model different stub requirements:(1) *Diurnal predicted:* The stub network has a fixed required bandwidth profile (spanning 24h) for specific application traffic to a destination. (2) *Peak predicted:* This is similar to the above, except that the stub requests a certain amount of bandwidth for a specific 1 hour period corresponding to a predicted peak in the application traffic volume. (3) *Instantaneous:* Both the above guarantees are based on stubs knowing something about their traffic profile beforehand. However, traffic to certain destinations could peak unexpectedly and stay high for long periods. In such cases, based on some initial monitoring, the stub many decide to instantaneously purchase a certain amount of bandwidth for some upcoming duration of time. (4) *Elastic bulk:* This models delay-tolerant bulk transfers (e.g., prefetching VOD content, transfers of scientific datasets etc.). The stub requests to transmit some number of bytes (e.g., 500GB) within a given time interval (e.g., the next 12 hours).

S4R stubs provide a *value* associated with the traffic to a destination, which is treated as private information. This encodes the amount that the stub is willing to pay per Mbps and drives the stub's *selfish* behavior. Also, stubs are *local utility-maximizing*: Given link prices, a stub can select routes such that its *utility*—the difference between the value derived by the stub and the price it pays—is maximized on a per-destination and per-application basis.

ISPs: Each ISP offers to carry traffic across a "virtual link" between two network locations (e.g., specific PoPs) at some cost per unit bandwidth. ISPs can be large national/international-scale providers or even third party service providers. ISPs are *revenue-maximizing*, setting prices to maximize the revenue earned from the links owned. An ISP's revenue per link is the product of the stub network flow routed on the link and the link price per unit traffic. Thus, S4R enables ISPs to use pricing as a tool to attract traffic from even remote stubs and enhance their revenues, thereby freeing ISPs from the constraints imposed by rigid bilateral contracts today.

There are two possible approaches to accommodating the objectives/requirements of ISPs and stubs, discussed next.

A. Centralized S4R

In the centralized variant, stubs and ISPs communicate with a logically central, neutral *facilitator*; Similar arrangements are used for enabling partial transit and paid peering today [1]. Stubs submit their requirements and ISPs submit their virtual links. The facilitator faithfully *simulates* the selfish interactions of the stubs and ISPs, and finds a stable state, or an *equilibrium*, that is acceptable to everyone; We describe the mechanisms to emulate ISP and stub behavior in §IV.

At equilibrium, the facilitator computes: **(1) Stub routes.** For every 1hr interval, the facilitator outputs: *(a)* the S4R routes that the stub should use for the interval, and *(b)* how much traffic the stub should send across these routes. The routes and traffic splits can change on a hourly basis for diurnal-predicted and elastic-bulk demands. **(2) Link prices.** The facilitator outputs prices that ISPs should charge over time, and informs them of the amount of flow they should observe for a given stub over time. The ISP uses the latter to filter traffic from stubs who are sending more than agreed upon.

Since the participants are selfish, in order for the facilitator's solution to be acceptable to ISPs and stubs, it should ideally be a Nash equilibrium [27], where no ISP has the incentive to unilaterally deviate from the current price to boost revenue; and all stubs employ utility maximizing paths and have no incentive to route over alternate paths.

Flexible global policies: The facilitator can impose flexible policies to ensure some *global properties* are satisfied, as well. The default policy is that the net value derived across all stubs who end up using the system is maximized. While this offers global efficiency, it could result in unfairness. A different policy could ensure that a certain minimum amount of demand from each region is guaranteed routes, irrespective of the relative values of the stubs located at each location. This introduces some level of fairness into the system. Another policy could enforce both fairness and value-optimality.

B. Distributed S4R

In the distributed variant, ISPs and stubs interact directly by passing messages along, rather than through a facilitator. In this setting, two possibilities arise to communicate link prices to stubs: (a) ISPs employ link-state routing to spread link price and topology information on a regular or triggered basis. (b) ISPs register the current prices with a central database, which stubs query for prices. In both cases, stub networks react to the prices by routing their request messages along newly-found least cost paths. ISPs compute the expected traffic they will carry based on the messages that traverse their links over time and adjust their link prices accordingly. We describe algorithms for ISP price adjustment and stub routing in §IV.

The advantage of this mechanism is that it does not rely on a central point of trust. However, since player actions are completely uncoordinated, global desirable properties such as optimal net value or fairness cannot be guaranteed.

C. Support for Stub Requirements

We isolate traffic belonging to different classes of stub requests from each other because the requests have different semantics in terms of when they can be placed and by when a stub must hear an answer; e.g., diurnal requests are placed at the beginning of 24 hour time-periods, peak-predicted requests are placed at 1 hourly intervals, while instantaneous requests can be placed at any time. More specifically, we run three logically separate approaches (distributed or centralized): one

for accommodating instantaneous requests, one for accommodating peak-predicted requests, and another joint one for elastic and diurnal-predicted requests. Alongside, ISPs offer up to three logically independent sets of virtual links.

Except for instantaneous demands, we enforce that stubs employ the chosen routes and ISPs charge stubs *only after S4R has reached convergence* to ensure stable overall operation. The ISP and stub response mechanisms described next are designed so that the network will converge to a correlated equilibrium. In the centralized variant, the facilitator runs the simulation until some termination condition is reached. In the distributed variant, ISPs continue spreading topology and price information while stubs send their requests along least-cost paths for a fixed large number of iterations pass (§IV), after which actual traffic can be routed. Our evaluation (§VI) shows that both the centralized and distributed approaches reach stable operating points. To support instantaneous demands, the S4R system runs in an online fashion.

We note that three of the diurnal predicted, peak predicted and instantaneous usage scenarios can be supported in both the distributed and the centralized variants. In contrast, bulk transfers is best supported in the centralized version: the facilitator can accommodate bulk transfer requests by "spreading" their demand over time intervals when there is little demand.

IV. Design Details

The algorithms used by S4R participants have two design goals that are at odds with each other: (1) enable participants to selfishly meet their objectives; (2) ensure that the S4R system operates in a stable fashion and dynamically adjusts to varying conditions (e.g., stub requests entering and leaving, ISP links going down, etc.). To reconcile these goals, for ISPs, we borrow ideas from regret-minimization to design effective price-adjustment approaches. For stubs, we develop a novel heuristic for traffic reassignment across ISP paths.

A. Distributed S4R Design

ISPs. An ISP prices links so as to maximize its revenue given the other ISPs' and stubs' choices. Since the ISP has incomplete information about others' strategies, and moreover strategies change dynamically, the ISP faces an online optimization problem. Each ISP therefore can employ a *learning algorithm* to determine the best strategy. The learning algorithm successively tries different prices at different iterations (employing historical information) and gradually converges to an optimal price. The algorithm attempts to minimize the "regret" of the ISP, which is the difference between the optimal average revenue achievable through a single price and the average revenue obtained by the algorithm overall.

We present two iterative regret-minimizing learning algorithms: (1) The *epsilon-decreasing explore-exploit* algorithm [29], is a simple learning algorithm that is proven to minimize regret in a static environment (if strategies of other ISPs and stubs are picked from an unchanging probability distribution, and do not adapt to this ISP's strategy). (2) The *hedge-bandit* algorithm [8], provably minimizes regret even in worst-case dynamic online settings. While variants of these have been studied in theoretical settings, to the best of our knowledge we are the first to apply them to practical dynamic network pricing settings and to evaluate their performance in realistic situations (§VI).

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: $\pi_i = \alpha\pi_i + (1 - \alpha)X$.

Fig. 1. Algorithm *epsilon-decreasing explore-exploit*

In *epsilon-decreasing explore-exploit*, at every iteration the ISP either "explores" 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 exploit with probability $1 - \epsilon_t$. In the beginning, ϵ_t is set to a high value. As history is accumulated it decreases with t: $\epsilon_t = \min(N/t \log t, 1)$ for a constant N that depends on network size.

To pick the best-price-so-far in an exploit step, the ISP 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 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 (Fig 1).

Hedge-bandit also "explores" at every iteration with an ϵ_t probability. However, in this algorithm ϵ_t remains constant. The crucial difference 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 (Fig 2). This is crucial to gracefully accommodating churn (§VI).

Stubs. A stub maximizes 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 ISPs. Therefore, every stub must distribute its traffic over least cost paths to its destination. Stubs use the default BGP path for traffic with value less than the price on the least cost paths.

We now discuss a novel heuristic which stubs can employ to meet their requirements optimally given the current network state (Fig 3): each stub maintains a list of the paths that it currently uses, as well as a list of currently least-cost paths.

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.

Fig. 2. Algorithm *Hedge-Bandit*

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 $f_P = f_P + r/f_{\text{tot}} \max(f_P, \delta)$.

Fig. 3. Distributed flow-update algorithm for stubs

At every iteration the stub 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.

We also study a smoother version of this selfish flow-update algorithm to understand whether slow updates to stubs' flow lead to better convergence. Conceptually similar algorithms [6] have been employed to solve flow problems in (non-selfish) settings and have been theoretically shown to have good convergence times. In the smoother version, each stub gradually moves its traffic from paths used in the previous iteration to those that currently charge the least price. Each stub 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 stub selfishness — $\epsilon_s = 1$ corresponds to a "selfish" version where stubs always only use least-cost paths, while $\epsilon_s < 1$ corresponds to the "smooth" version where stubs 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 fluctuations in ISP prices, on the other hand, it may adversely affect performance because ISPs get slower feedback to their price changes.

B. Centralized S4R Design

The facilitator derives an equilibrium meeting some global objective using an iterative approach, simulating ISP and stub actions at each step by first picking prices for ISPs and then flow paths for stubs, until some termination condition is met.

ISP actions: The facilitator simulates a repeated game among the ISPs. 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 [15]: no player has incentive to deviate from its strategy if other players continue to follow their strategies. The facilitator simulates one of the regret-minimization algorithms described earlier.

Stub actions: At every iteration, the facilitator simulates stub behavior by routing the flow of each stub along the least cost path given link prices. Despite the constraint of following best response, the facilitator has substantial flexibility in routing flow because some source-destination pairs may have multiple least cost paths between them. Moreover, for some stubs 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 BGP path brings equal utility to them. The facilitator may use this flexibility to implement desirable social objectives; we focus on *maximizing net social value* (§II). To achieve this objective, the facilitator solves the following max-value flow problem: It first determines for every stub 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 stub along every such least cost path while honoring capacity constraints on edges and maximizing total value of the flow routed. This can be set up as a LP (omitted). If for some stub the cost of the least cost path is strictly less than the stub's value, it is in the best interest of the stub to route its entire flow, while the LP may only route a fraction of the flow to satisfy capacity constraints. To rectify this, for every stub with value strictly larger than the cost of the least cost path, the facilitator finds an arbitrary least cost path and routes the entire remaining flow along the path.

Termination condition: The facilitator could use a variety of tests to determine when to halt the simulated interaction between stubs and ISPs. One approach is to check if the total utility as perceived by both ISPs and stubs does not improve any further, i.e., checking if the difference between the 95th and the 5th percentile net utilities is less than a small fraction (say 5%) of the 95th percentile.

C. Churn in Instantaneous Requests

The issue of churn arises in the case of instantaneous demands when stub requests arrive and leave S4R unexpectedly; in all other cases, the requests are known to S4R ahead of time and churn isn't an issue. The distributed and centralized variants are designed to accommodate slow churn, where a small fraction of stubs enter and leave the system over time.

In the centralized framework, the facilitator can recompute a new equilibrium from the current state when facing churn. This happens in an offline fashion and current traffic is unaffected during the recomputation. Note however that addition of new stubs may affect whether or not some current stubs can continue routing over S4R in the new equilibrium; existing stubs, if denied, stop using the system and may submit a new request. As shown in §VI, stubs who have low willingness to pay are likely to be affected by churn in the system, and we consider this to be an acceptable outcome. In the distributed case, churn perturbs the system, forcing reconvergence to a new equilibrium within the running system itself. Any existing request can get affected during reconvergence, even though the eventual equilibrium accommodates the request. There are two reasons why a request is affected in this fashion: (1) the capacity of a link it is using overflows in some reconvergence iteration (because the link price is too low)—we describe in

§VI simple mechanisms to mitigate the impact of this in real scenarios, or (2) all paths are too expensive at the price levels in some reconvergence iteration—we find, in practice, that this typically only affects stubs with low values.

V. Using OpenFlow

We now discuss how to support S4R paths alongside traditional BGP paths atop the same underlying routing infrastructure and to configure and tear-down paths with specific requirements on the fly, where the paths can be specified at the granularity of application flows. We believe that OpenFlow provides the appropriate platform for addressing these challenges.We note that the distributed architecture can also be implemented using OpenFlow, but omit the details.

An OpenFlow network consists of two entities: NOX, a logically central controller, and dumb switches which act on the controller's instructions. Depending on specific events, the controller installs or removes flow forwarding rules on the switches. Rules are based on matching a ten-tuple for a flow that includes the source and destination IP and port and VLAN ID, among other things. Actions taken on matched flows range from permitting them (with or without mangling the flow) or denying them or forwarding to the controller. Flow rules can be stored at switches for various time periods allowing rules to be updated frequently or act as semi-permanent routing entries. We assume that OpenFlow is enabled on edge routers of stubs and on routers at either end of virtual links.

Stubs submit their requests to their respective controllers who forward them to the facilitator along the logical links shown. Once the facilitator computes routes, the NOX controllers corresponding to the on-path ISPs, as well as the source and destination controllers, are informed of the routes. The controllers then install the corresponding forwarding rules at their switches. When stubs leave, they send an explicit tear-down message. If a stub requires a single path then, simply, flow entries are installed for the corresponding connection 4-tuple at each router on the path. A stub who is offered multiple routes per destination is provided with a nonce per route. In addition, the facilitator computes a unique per-link nonce for each path. The flow tables for the stub's traffic along each path include the nonce as well, and indicate how the nonce must be rewritten for the next link. The nonce is useful to distinguish among multiple equal cost paths.

Key features in OpenFlow enable S4R to meet various application requirements. Prior work [20] has shown how to leverage QoS extensions to the OpenFlow API (i.e., per-flow rate limiters and dynamic priorities) to slice network bandwidth and dynamically assign flows to slices to meet their performance requirements. We leverage this to isolate stub requests and prevent misbehaving traffic from impacting others with hard performance constraints.

VI. S4R Viability

Some of the theoretical underpinnings of S4R have been studied in different contexts. S4R is similar to a real world marketplace where customers are willing to shop around for the best prices for sets of goods and stores try to competitively price goods to attract customers to purchase from them. Since each ISP is interested in maximizing its own revenue, the overall system performance at equilibrium may not be "socially" optimal. A natural question is how far from optimal can the system performance be. There is a rich body of recent work in algorithmic game theory that investigates such questions in the context of network pricing and network formation games [10], [9], [16], providing bounds on the "price of anarchy" (POA), namely the ratio of the worst-case system performance at equilibrium to the social optimal.

The works most relevant to S4R are [10] and [9]. These consider a general market with buyers and sellers, where each seller owns a distinct item and prices it selfishly. Each consumer buys the cheapest desirable "bundle" of products. In a network setting, the product is bandwidth; each seller is a ISP owning one virtual link, while consumers are interested in bundles of edges that form source-destination paths. For pathological instances, the price of anarchy can be unbounded, implying that system performance can be significantly far from optimal [10], [9]. The poor efficiency means that few stub networks and ISPs are likely to extract utility from S4R.

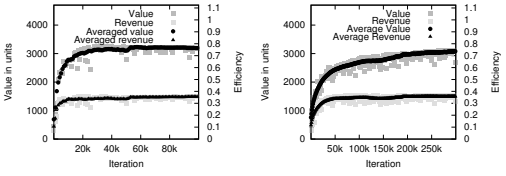
The works also find that when stub values satisfy the *monotone hazard rate (MHR)* condition [7], the worst-case performance improves significantly: it is worse than optimal by a factor no more than exponential in the number of hops between any source and the sink, and is independent of other parameters such as the values themselves, network size, available capacities, etc. The MHR condition is widely used in economics to characterize commonly occurring value distributions. Most natural distributions, e.g., uniform, normal, exponential, power-law, Laplace and chi-square, satisfy MHR [7]. This observation has implications for S4R because we can expect the distribution of per-unit-demand values to follow the stubs' (or users') wealth distribution, which is a power-law distribution with the MHR property. While this is somewhat "positive" for S4R, it still points to the fact that the outcome in practice can be quite far from the optimal, which brings S4R's viability into question.

A. S4R Evaluation

We emulate different realistic scenarios to understand to what extent the above pessimistic results hold in practice given our design choices described so far. The key questions we ask are: Does S4R's operation reach a stable point in practice? How far from optimal is S4R at this point? We study the social value derived by the system relative to the optimal social value, also called "efficiency". This measures the ability of stubs to obtain benefit while allowing the ISPs to extract maximal revenue. We also evaluate practical viability: (1) Does the lack of central control in the distributed variant set it at a disadvantage relative to the centralized variant? (2) How do various ISP and stub response algorithms described §IV perform in practice? (3) How well does S4R support different usage models and churn? Finally, we study economic viability: (4) Do stubs who value their traffic the most get to use the system? Is there a skew in how revenues are distributed across

Topology	Nodes, Edges
Rocketfuel-based	50, 100
Power law	46, 93
Random (prob. edge = 0.8)	50, 93
Regular (deg = 4)	50, 100

TABLE I

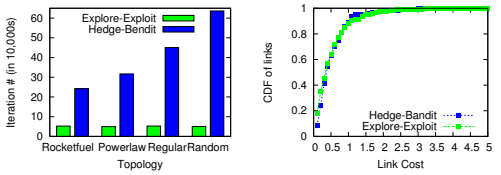


(a) EE Algorithm (b) HB Algorithm

Fig. 4. Convergence: Rocketfuel, Zipfian values.

Topology	Value Distribution	Convergence	Efficiency
Rocketfuel	Zipfian	52,000	0.78
Rocketfuel	Uniform	53,000	0.70
Rocketfuel	Bimodal	53,000	0.75

TABLE II
CENTRALIZED FRAMEWORK, WITH ISPS USING EE.



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

Fig. 5. Comparing EE and HB; Zipfian stub values

ISPs? (5) How do the underlying interconnect, distribution of stub values and the presence of monopolies impact S4R?

Topologies. We use models of ISP interconnection topologies (Table I). As a baseline, we extract a PoP-level topology spanning several ISPs ($\sim$12 in number) from Rocketfuel; the virtual link graph we use reflects 50 PoPs in highly populated cities spanning all tier-1 and some tier-2 ISPs in N. America, and the measured edges between them (100 in all).

Source stubs are located at any node in the above topology. In all, there are $10,000$ source stubs. Destinations are located at the 16 most highly populated nodes, where we assume that node population is proportional to degree. Traffic follows gravity model: the demand between a pair of nodes is proportional to population. Our baseline topology roughly reflects how S4R may be used in practice, wherein large stubs (10000 in all) wishing to reach clients in large cities purchase suitable end-to-end paths that consist of PoP-to-PoP (i.e., inter-city) virtual links in multiple ISPs. We also generated three other synthetic topologies – Power-law random graphs, Erdos-Renyi random graphs and regular graphs with roughly the same numbers of vertices and edges – to understand the impact of the ISP interconnection structure. Link capacities are 25 or 100 units. We also performed experiments using other topologies with different degrees of path diversity and path lengths.

Link costs and stub values. Each link is assigned an initial cost generated randomly between 0.1 and 5 units at a granularity of 0.1 units. ISP link costs always stay in $[0.1, 5.0]$. Each stub 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 Zipf with an exponent of $\alpha = 1$. This represents a typical income model and satisfies the MHR condition. The second set is drawn uniformly at random from the range and also satisfies MHR. The third is a bimodal distribution drawn from [0.1,1] or [4.5 and 5], with 90% of values originating from the first interval. This does not satisfy MHR. Unless otherwise specified, we assume that all stubs are employing the peak-predicted usage model. We also evaluate other usage models.

B. Centralized S4R

Convergence. Figure 4 shows the evolution of net value for stubs routing in S4R and total ISP revenue for one of our experiments. We present results for both ISP learning algorithms—EE (epsilon-decreasing explore-exploit) and HB (hedge-bandit). Both curves flatten after 50,000 iterations for EE (Fig 4(a)) and 200000 iterations for HB (Fig 4(b)). Thus S4R moves to a stable state over time. The termination condition (§IV-B) is actually met at around 52,000 and 230,000 iterations for the two algorithms, respectively. Although not shown here, we found that the system eventually converges to an equilibrium–i.e., an unchanging set of prices and routes–after some large number (>500K) of iterations across all the scenarios we studied. Compared to the status of the system at the termination condition above, the final equilibrium differs very little in terms of the per-link prices and overall value derived by stubs. Henceforth, we refer to the system at the termination condition itself as being the equilibrium state.

Efficiency. We now discuss system efficiency at equilibrium. Table II shows representative results for different topologies and stub value distributions. We assume that the EE algorithm is used to emulate ISP learning. Surprisingly, S4R arrives at a reasonably efficient outcome in most of the realistic situations (60%-80% efficiency) despite allowing selfish ISP and stub interactions, contrary to what the theory suggests. Value distributions that were shown to result in undesirable outcomes by prior analytical results, e.g. bi-modal (which is not MHR), result in efficient outcomes.

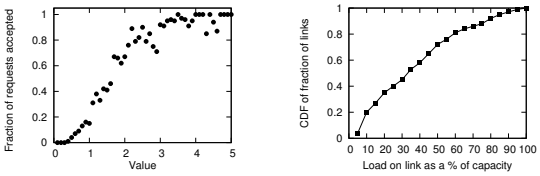
Comparing ISP Learning Algorithms. We now compare the two ISP algorithms. The results of the comparison are shown in Figure 5, where we study the time taken for S4R to converge and the link prices derived by either algorithm.

Several points are worth noting: Figure 5(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 (not shown). However, Figure 5(a) indicates that HB takes a lot longer (4-10$\times$) to converge. 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 5(a)).

HB is designed for the worst case (§IV), and hence may be more appealing under a wide spectrum of situations. However, our evaluations shows that EE works well in most practical

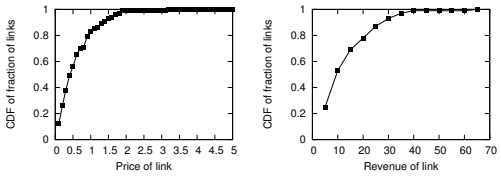
ISP Price Algorithm	ϵ_s	Efficiency	Convergence
EE	0.1	0.39	73,000
EE	1	0.69	52,000
HB	0.1	0.19	100,000
HB	1	0.61	150,000

TABLE III



(a) Stub Networks Utilizing the System (b) ISP Link Utilizations

Fig. 6. Distributed S4R at equilibrium



(a) Link Price (b) ISP Revenue

Fig. 7. Distributed S4R: ISPs at equilibrium

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

TABLE IV

situations anyway. Because of its simplicity and reasonable performance we advocate using EE in practice.

C. Distributed S4R

Vs. Centralized Variant. When there can be multiple stable states, a central facilitator can suggest the best one to all the participants, e.g., the one that maximizes social value. Such optimizations cannot be performed in the distributed framework. We examine the extent to which the distributed framework suffers due to the lack of this global optimization. Interestingly, the distributed approach performs nearly as well as the centralized one. The efficiency difference compared to centralized is just 0.07-0.12 in all scenarios (not shown).

The distributed approach is designed to run in a continuous online fashion. To understand if the system oscillates, we applied a variety of tests. We monitored the change in the net social value and net ISP revenues over time in various settings. In particular, we applied the centralized termination condition (§IV-B) to the state in various iterations of the distributed variant to examine if and when the distributed variant "terminates". We found that the distributed framework achieves convergence in all cases (omitted for brevity) and that participants in the distributed approach can keep track of the iteration number as they exchange messages and requests (§III-C), and start routing data traffic after it crosses a large value (e.g. 100,000).

Stub Response. In the distributed setting, the stub flow update algorithm (Figure 3) gradually moves an ϵ_s amount of the stubs' 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 stubs to be "fully selfish" ($\epsilon_s = 1$) leads to poor outcomes. Table III summarizes our findings. Surprisingly, we find that a large ϵ_s leads to robust performance—enabling stubs to be selfish is not bad from a global viewpoint. Constraining stubs from reacting selfishly (i.e., using a small ϵ_s) seems to lead to substantially inferior performance at equilibrium. This is because when stubs use small ϵ_s, ISPs get very slow feedback about the effects of their price changes. ISPs may then wrongly associate the price changes they made with the small increases/decreases in the amount of traffic they are carrying and the resulting revenue.

D. ISP and Stub Properties at Equilibrium

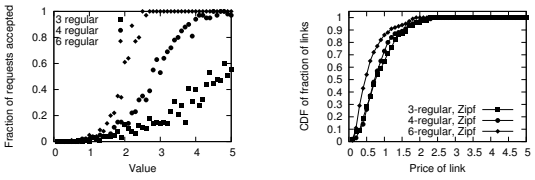
We now examine the status of stub networks and ISPs at equilibrium. We consider the distributed approach and take a snapshot of the link utilizations, stubs routing through the network, link prices and ISP revenues. In Figure 6(a), we show the fraction of stub networks that are allocated routes at each value for the distributed framework. We see that very few stub networks with low values (< 1 unit) get to route using the system. A small number of stub networks with high values are denied service—this could happen because of where they are located and how many other stubs they are competing against.

In Figure 6(b), we show the utilizations of links owned by different ISPs. A small fraction of the links (20%) have utilizations $<$10%. About 30% of the links have $\geq$ 50% utilization. This suggests that most ISPs are able to attract traffic onto their links using S4R. In Figure 7 we show the link prices and revenues of ISPs. We find that in an overwhelming fraction of the cases, the equilibrium prices are far lower than the maximum possible value of 5 (they are $\leq$ 2 units; Figure 7(a)). From Figure 7(b) we see that revenues are fairly evenly distributed across the ISPs and there is little skew. Roughly 45% of the revenues are in the 10-40 unit range. The reasonable distribution of revenues and link utilizations indicate that ISPs would be willing to take part on S4R. Since stubs with moderate to high values are able to route in S4R, it would be able to attract enough stubs as well.

E. Factors Impacting Equilibrium Properties

Theory shows that networks that contain significant disparity among stubs in terms of per-unit-demand values (i.e., non-MHR distributions) can have poor equilibria. Our results in the previous section show that value disparity still leads to reasonable outcomes in practice. Theory also showed that the price of anarchy in the market pricing game depends primarily on the existence and number of "virtual monopolies" in the network. In S4R, 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). Crucially, as the hop-lengths of paths in a network increase, this could potentially lead to a larger number of virtual monopolies for any stub network, causing system performance to worsen significantly. We now study the impact of virtual monopolies by varying the connectivity of the network.

We select the baseline regular graph topology (with per node degree of 4) and change the average degree of each node to 6 and 3 to increase/decrease, respectively, the route diversity and path lengths. We employ the distributed S4R

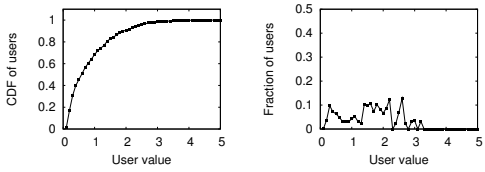


(a) Stub Networks Utilizing the System (b) Link Prices

Fig. 8. Stub Network and ISPs in regular graphs.

Iteration	Demand Increase	Reconvergence	Efficiency	% Stubs Preempted
58100	10%	6000	0.89	2%
77600	1%	6500	0.84	2%
84100	10%	6100	0.82	2%

TABLE V



(a) CDF of Preempted Stubs (b) Fraction of Stubs Preempted

Fig. 9. Preempted stubs in the instantaneous demand scenario.

variant in this analysis. As shown in Table IV, in networks where path diversity is restricted (correspondingly, there is a high incidence of virtual monopolies), S4R efficiency suffers significantly. Figure 8 shows the distribution of values of stubs who route over S4R, and the link prices selected by ISPs, in the different regular topologies. As expected, link prices increase slightly with decreased path-diversity (Figure 8(b)), because virtual monopolies charge higher equilibrium prices. More tellingly, the set of stubs using the system is very different. In topologies with larger degrees, greater fractions of stubs 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 stubs are denied. The effect of higher link prices is aggravated by longer paths, leading to a poor social outcome. Thus, when the ISP interconnect has poor path diversity and path lengths are long S4R is unlikely to be viable (i.e., few stubs are likely to use it). Luckily, S4R places a low barrier to entry: any ISP can provision a virtual link and use intelligent pricing as a tool to attract traffic to its link and enhance revenue. This will ensure a rich ISP interconnect and short paths in practice.

F. Churn: Instantaneous Demands

We study the performance of both centralized and distributed S4R when stubs enter and leave with instantaneous demands. We start the system with an initial set of stubs and let it attain equilibrium. We remove 10 random stubs and add 100 random new stubs such that the net increase in Instantaneous demand is 1-10% of current demand. Re-computation of equilibrium occurs only if these stubs can not be accommodated on the current least cost paths. We repeat this multiple times. The results of this experiment for the centralized variant are presented in Table V. An important question is how efficient the system is given that due to time constraints the solution must be found quickly by computing a new equilibrium relative to current equilibrium state. To understand this, we compute the reconvergence efficiency, which is the difference between the system's observed performance and the "ideal" performance had we let it reconverge to a new equilibrium with the current set of stubs. The reconvergence efficiency is never worse than 75%. We also note that recovergence is quick, averaging less than 6500 iterations.

We also analyze the stubs which are preempted at equilibrium due to the new stubs who are added in. Figure 9(a) shows, over the entire duration of the experiment, a CDF of the stubs who get preempted, and their values. Figure 9(b) shows the fraction of stubs preempted compared to the number of other stubs with that value in the system for one random iteration. We see that typically only stubs with lower values get preempted. Corresponding results for the distributed variant are similar and omitted for brevity.

At certain times during distributed reconvergence, links may become overloaded due to extra demand in the system, causing dissatisfaction to stubs as well as ISPs. To address this, each ISP link can have a small fraction of its capacity α_b set aside as backup. Each ISP sets or alters prices to operate below $1-\alpha_b$ of its link capacity at equilibrium. The spare α_b fraction of capacity can thus be utilized to accommodate momentary oversubscription and ensure that stubs are not impacted during churn. We found that setting $\alpha_b = 8\%$ is sufficient to ensure that no link is overwhelmed with high probability (omitted).

Our results so far have been for the peak-predicted and instantaneous demands. We also evaluated diurnal-predicted and elastic-bulk requests and found that S4R converges to 80% efficiency on average in both cases (omitted).

Summary of Findings: (1) S4R converges to a stable operating point with 65-80% efficiency, showing that S4R will be of of high overall utility. S4R is efficient even when the disparity in stub values is high in practice. Both are contrary to theoretical analysis. (2) The distributed approach converges in all situations with slightly inferior efficiency to the centralized one. The simplistic EE learning algorithm performs better in practical situations compared to HB which has proven worst-case guarantees. Selfish stub response for rerouting leads to better outcomes as it provides more up-to-date information to ISPs. (3) S4R effectively supports all the four usage models. S4R can accommodate a modest amount of churn. At equilibrium, stubs who have the highest values for their traffic always find paths, and there is no significant skew in ISP revenues. Thus, both ISPs and stubs will find S4R attractive. (4) S4R efficiency suffers under limited path diversity and/or long paths. But, as the barrier to entering S4R is low, we expect rich interconnection and path diversity.

G. Centralized Prototype

We implement the facilitator in a standard Linux desktop. It can communicate with one or more NOX controllers. We implemented a simple messaging protocol for the facilitator to inform NOX controllers of the flow state they should install. We use a separate logical facilitator and logical NOX controllers for different types of stub requests. The controllers each control distinct pre-assigned portions of switches' flow table space and of the available link capacities. Our testbed emulates the configuration shown in Figure **??** albeit over a

single 1Gbps ISP link. We have 10 stubs (VMs in a single desktop) attached to a software OpenFlow switch communicating with 10 destination VMs attached to another switch; each switch communicates with a separate controller, which are connected to the facilitator.

Communication. The messages exchanged in S4R are: (1) The sending stub sends a request to use S4R. (2) The facilitator acknowledges the request and starts route computation. (3) The facilitator notifies stubs if their request was accepted. (4) The facilitator notifies ISPs of link prices to charge. The route itself is installed directly in switches. Messages (1) (2), and (3) are minimal in size (at most 150 bytes). Messages for (4) require the most bandwidth as there is one flow table entry per flow on a link. Our analysis of the paths computed for the Rocketfuel topology shows that, $>95\%$ of the time, the paths have ≤ 5 hops. Thus, the overhead of these messages is small ($\sim$300B per request per path). With 10,000 diurnal-predicted requests, this would result in $<$190MB of control traffic.

Computation Time. With 10000 stubs, for instantaneous requests, the amount of time taken varies from a few ms to one minute. The computation time is low if the stub can be accommodated on the least-cost path at the current prices, high if accommodating the new stub results in churn in the system.

Performance guarantees. To demonstrate the ability to support stub performance guarantees, we place 10 requests where request i needs $i * 20$Mbps. We implement a toy facilitator that rejects requests with the least demands under over-subscription; thus, requests $i = 1, 2, 3$ are denied in our toy scenario. For the remaining requests, we install flow table entries at the switches, instantiate traffic at the stub (IPerf traffic generators sending as fast as possible), and measure bandwidth at the destination. We find that each stub $i > 3$ obtains roughly what it requested. The maximum deviation between the requested and obtained bandwidths was $\pm 9\%$.

VII. Related Work

S4R bears some similarity to "nanopayment" systems like Bill-Pay [12] where ISPs announce prices for various levels of service across their networks. Users employ source routing and include payments in their packets to be deducted by each ISP. Because these protocols operate at packet-level, they impose high overhead for ISPs and require significant changes to applications. Our work extends bandwidth brokering systems where the primary focus was QoS in a single ISP [32]. Some recent proposals have argued for inter-domain brokering [21], [28]. For example, the MINT proposal [28] uses a centralized mediator to run a continuous double auction to match requests with available bandwidth resources. While these proposals share the same broad goals as us, they do not take into account the selfish goals of stubs and service providers, and they don't analyze the outcome of direct interactions between them. Several routing systems enable stubs increased routing flexibility [13], [31], [30], [19], [17], [2]. However, these systems do not consider the ISP-side view, in particular, ISP pricing and the resulting interactions among ISPs/stubs.

VIII. Conclusions

S4R is an economically-grounded technical framework that supplements BGP to provide flexibility that is missing from BGP. S4R allows free form interaction between stubs and ISPs, enabling them to maximize their selfish objectives. Despite prior theoretical work showing that freeform interactions of this kind in a large marketplace can lead to arbitrarily bad outcomes, we find, surprisingly, that the performance of S4R is close to the best possible social outcome in practical scenarios. We built an OpenFlow-based prototype for S4R and showed that it imposes little overhead.

References

[1] Band-X: Bandwidth Exchange. http://www.band-x.com.

[2] BGP/MPLS IP VPNs. http://tools.ietf.org/html/draft-ietf-l3vpn-rfc2547bis-03, 1999.

[3] Apple Online Sales Huge On Black Friday. http://www.businessinsider.com/black-friday-mac-sales-down-sharply-this-year-2009-11, 2009.

[4] A. Akella et al. A Comparison of Overlay Routing and Multihoming Route Control. In *SIGCOMM*, 2004.

[5] D. Andersen, H. Balakrishnan, M. Kaashoek, and R. Morris. Resilient Overlay Networks. Banff, Canada, Oct. 2001.

[6] B. Awerbuch and R. Khandekar. Stateless distributed gradient descent for positive linear programs. In *STOC*, 2008.

[7] M. Bagnoli and T. Bergstrom. Log-concave probability and its applications. *Economic Theory*, 26(2):445–469, 08 2005.

[8] C. Burch. *Machine learning in metrical task systems and other on-line problems*. PhD thesis, Pittsburgh, PA, USA, 2000. Chair-Blum,, Avrim.

[9] S. Chawla and F. Niu. The Price of Anarchy in Bertrand Games. In *ACM Conference on Electronic Commerce*, pages 305–314, 2009.

[10] S. Chawla and T. Roughgarden. Bertrand competition in networks. In *Symposium on Algorithmic Game Theory*, pages 70–82, 2008.

[11] B. Davie and Y. Rekhter. *MPLS: technology and applications*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2000.

[12] C. Estan, A. Akella, and S. Banerjee. Achieving good end-to-end service using Bill-Pay. In *ACM HotNets-V*, Irvine, CA, Dec. 2006.

[13] B. Godfrey et al. Pathlet routing. In *ACM HotNets*, 2008.

[14] T. G. Griffin. Keynote: What can we unlearn from bgp? In *DIN05*.

[15] S. Hart and A. Mas-Colell. A simple adaptive procedure leading to correlated equilibrium. *Econometrica*, 68(5):1127–1150, 2000.

[16] A. Hayrapetyan et al. A network pricing game for selfish traffic. In *PODC*, 2005.

[17] J. He and J. Rexford. Towards internet-wide multipath routing. Technical report, Princeton Univ., 2007.

[18] J. He and J. Rexford. Toward Internet-wide Multipath Routing. 2008.

[19] U. Javed et al. Multipath protocol for delay-sensitive traffic. In *ICCSN*, 2009.

[20] W. Kim et al. Automated and Scalable QoS Control for Network Convergence. In *INM/WREN Workshop*, 2010.

[21] J. Lane et al. Path brokering for end-host path selection: Toward a path-centric billing model for a multipath internet. In *ReArch*, 2008.

[22] N. McKeown et al. Openflow: enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, 38(2):69–74, 2008.

[23] W. B. Norton. Internet service providers and peering, 2000.

[24] Prolexic Technologies Home Page. http://www.prolexic.com, 2008.

[25] L. Qiu et al. On Selfish Routing in Internet-Like Enviroments. 2003.

[26] S. Savage et al. Detour: A Case for Informed Internet Routing and Transport. 19(1):50–59, 1999.

[27] É. Tardos and V. Vazirani. Basic solution concepts and computational issues. In *Algorithmic Game Theory*, chapter 1, pages 3–28. 2007.

[28] V. Valancius et al. Mint: A market for internet transit. In *ReArch*, 2008.

[29] J. Vermorel et al. Multi-armed bandit algorithms and empirical evaluation. In *European Conf. on Machine Learning*, 2005.

[30] W. Xu and J. Rexford. Miro: multi-path interdomain routing. In *SIGCOMM*, 2006.

[31] X. Yang, D. Clark, and A. W. Berger. Nira: a new inter-domain routing architecture. *IEEE/ACM Trans. Netw.*, 15(4), 2007.

[32] Z.-L. Zhang et al. A novel bandwidth broker architecture for scalable support of guaranteed services. In *SIGCOMM*, 2000.